**Redis,Memcached, MongoDB区别**

**Redis是一个开源（BSD许可），**内存存储的数据结构服务器，可用作数据库，高速缓存和消息队列代理。

支持多种数据结构、支持持久化到磁盘操作（RDB/AOF）、支持主从可同步复制增量复制，支持事务但比较弱

**应用：**适用于对读写效率要求都很高，数据处理业务复杂和对安全性要求较高的系统

**Memcached是一个开源的，**高性能，分布式内存对象缓存系统。

有多核优势、支持的数据类型不多只支持key/value、无法持久化、不能备份数据、只能用于缓存

**应用：**动态系统中减轻数据库负载，提升性能；做缓存，适合多读少写，大数据量的情况

**MongoDB是一个开源的基于分布·**式文件存储的数据库,文档型的非关系型数据库，与上面两者不同。

吃内存较大、不支持事务、内置置了数据分析的功能(mapreduce)其他不支持、1.8开始支持binlog方式持久化

**应用：**主要解决海量数据的访问效率问题。

**1.性能上：**

性能上都很出色，具体到细节，由于Redis只使用单核，而Memcached可以使用多核，所以平均每一个核上Redis在存储小数据时比Memcached性能更高。而在100k以上的数据中，Memcached性能要高于Redis，虽然Redis最近也在存储大数据的性能上进行优化，但是比起 Memcached，还是稍有逊色。

前两者性能差不多，大于 Mongodb。

**2.内存空间和数据量大小：**

MemCached可以修改最大内存，采用LRU算法。Redis增加了VM的特性，突破了物理内存的限制。

　 MongoDB 适合大数据量的存储，依赖操作系统 VM 做内存管理，吃内存也比较厉害，服务不要和别的服务在一起

**3.操作便利上：**

MemCached数据结构单一，仅用来缓存数据，而Redis支持更加丰富的数据类型，也可以在服务器端直接对数据进行丰富的操作,这样可以减少网络IO次数和数据体积。

　 Mongodb 支持丰富的数据表达，索引，最类似关系型数据库，支持的查询语言非常丰富。

**4.可靠性上：**

MemCached不支持数据持久化，断电或重启后数据消失，但其稳定性是有保证的。Redis支持数据持久化和数据恢复，允许单点故障，但是同时也会付出性能的代价。

　 MongoDB 从 1.8 版本开始采用 binlog 方式支持持久化的可靠性。

**5.应用场景：**

Memcached：动态系统中减轻数据库负载，提升性能；做缓存，适合多读少写，大数据量的情况（如人人网大量查询用户信息、好友信息、文章信息等）。

Redis：适用于对读写效率要求都很高，数据处理业务复杂和对安全性要求较高的系统（如新浪微博的计数和微博发布部分系统，对数据安全性、读写要求都很高）。

　 MongoDB:主要解决海量数据的访问效率问题。

**需要慎重考虑的部分：**

1.Memcached单个key-value大小有限，一个value最大只支持1MB，而Redis最大支持512MB

2.Memcached只是个内存缓存，对可靠性无要求；而Redis更倾向于内存数据库，因此对对可靠性方面要求比较高

3.从本质上讲，Memcached只是一个单一key-value内存Cache；而Redis则是一个数据结构内存数据库，支持五种数据类型，因此Redis除单纯缓存作用外，还可以处理一些简单的逻辑运算，Redis不仅可以缓存，而且还可以作为数据库用

4. Redis的（3.0）是指集群分布式，也就是说集群本身均衡客户端请求，各个节点可以交流，可拓展行、可维护性更强大。

5. MongoDB不支持事务。

**Redis安装(默认端口：6379)**

]# rpm -q gcc || yum -y install gcc ]# cd redis-4.0.8/ ]# make && make install

]# cd utils #提供了初始化脚本在此目录 utils]# ./install\_server.sh #运行初始化脚本

]# vim /etc/redis/6379.conf #配置文件6379.conf（配置登录密码等等）

]# redis-cli #进入数据库 ]# /etc/init.d/redis\_6379 stop #停止 ]# /etc/init.d/redis\_6379 start #启动

Set keyname keyvalue #定义数据给变量 get keyname #输出获取变量数据

Select #数据库编号0-15，切换库 Keys \* #查看所有变量

flushdb #仅清除当前库的数据

save #保存所有变量(正常默认是5分钟自动保存到硬盘,执行后立即保存)

Shutdown #关闭redis 服务 #停止服务时,自动将数据存入dump.rdb文件中

**创建redis集群：**集群必须至少要有三台主库,

创建方式1 --replicate 1 定义每个主库有1个从库,3台主库,3台从库,总共6台

创建方式2 --replicate 2 定义每个主库有2个从库,3台主库,6台从库,总共9台 以此类推

2.1修改redis配置文件

(以51为例,52-55同样)

[root@sql51 ~]# /etc/init.d/redis\_6379 stop

[root@sql51 ~]# ls /var/lib/redis/6379/

dump.rdb #redis数据文件,要删掉,不删掉无法创建集群 # rm -rf /var/lib/redis/6379/\*

[root@sql51 ~]# vim /etc/redis/6379.conf

70 bind 192.168.4.51 #不用回环地址

93 port 6351 #修改端口

137 daemonize yes #守护进程方式运行

501 #requirepass 123456 #501 注释掉密码

815 cluster-enabled yes #815 行开启集群

823 cluster-config-file nodes-6351.conf #指定集群信息文件

829 cluster-node-timeout 5000 #829当前主机与其他主机通讯超时时间5s

[root@sql51 ~]# vim /etc/init.d/redis\_6379 #修改redis脚本文件第43行

43 $CLIEXEC -h 192.168.4.51 -p 6351 shutdown

[root@sql51 ~]# /etc/init.d/redis\_6379 start #检测启动正常

[root@sql51 ~]# netstat -nultp | grep redis #查看端口

[root@sql51 ~]# ls /var/lib/redis/6379/ #查看重新生成的文件

dump.rdb nodes-6351.conf

[root@sql52 ~]# /etc/init.d/redis\_6379 stop #检测关闭正常

[root@sql52 ~]# /etc/init.d/redis\_6379 start

[root@sql52 ~]# vim /etc/init.d/redis\_6379

[root@sql52 ~]# redis-cli -h 192.168.4.52 -p 6352 #登录正常

192.168.4.52:6352> keys \* #查看所有值无任何数据

192.168.4.51:6351> cluster info #查看集群状态

cluster\_state:fail #集群状态失败

192.168.4.51:6351> cluster nodes #查看集群节点

412a79829c7d0eed812b24959ca123b58990d60e :6351@16351 myself,master - 0 0 0 connected

部署管理主机：ql51 RedisA 192.168.4.51 6351 同时兼任:管理主机

root@sql51 ~]# cd redis-cluster/

[root@sql51 redis-cluster]# ls

redis-3.2.1.gem ruby-devel-2.0.0.648-30.el7.x86\_64.rpm

[root@sql51 redis-cluster]# yum -y install ruby rubygems #脚本文件是ruby语言编写的,所以要安装解释器

[root@sql51 redis-cluster]# yum -y install ruby-devel-2.0.0.648-30.el7.x86\_64.rpm

[root@sql51 redis-cluster]# gem install redis-3.2.1.gem

[root@sql51 src]# cp redis-trib.rb /root/bin/ #创建命令工具

[root@sql51 src]# chmod +x /root/bin/redis-trib.rb

[root@sql51 src]# redis-trib.rb create --replicas 1 192.168.4.51:6351 192.168.4.52:6352 192.168.4.53:6353 2.168.4.54:6354 \

> 192.168.4.55:6355 192.168.4.56:6356 #创建集群--replicas 1方式：主主主 从从从

]# redis-cli -h 192.168.4.51 -p 6351 -a 123456 #进入51节点

192.168.4.51:6351> cluster info #查看集群状态

cluster\_state:ok #集群状态ok

[root@sql51 src]# redis-cli -c -h 192.168.4.51 -p 6351 #登录集群-c

192.168.4.51:6351> keys \*

**集群添加master 角色：默认添加的服务器就是主，从库需要特殊指定**

1 添加redis服务器：[root@sql51 ~]# redis-trib.rb del-node 要添加的主机ip:端口 任意主机:端口

[root@sql51 ~]# redis-trib.rb add-node 192.168.4.58:6358 192.168.4.51:6351

[root@sql51 ~]# redis-trib.rb info 192.168.4.51:6351 #查看查看集群主库信息

192.168.4.51:6351 (412a7982...) -> 2 keys | 5461 slots | 1 slaves.

192.168.4.58:6358 (c2e039c3...) -> 0 keys | 0 slots | 0 slaves. #存有0个变量

192.168.4.53:6353 (c269f171...) -> 1 keys | 5461 slots | 1 slaves.

。。。。。。。

2 分配hash slot(哈希槽)

[root@sql51 ~]# redis-trib.rb reshard 192.168.4.58:6358 #任意集群中ip

**集群添加slave 角色：**Add-node --slave [--master-id id值] ip地址:端口 192.168.4.51:6351

没有指定[--master-id id值],默认谁的主库最少,自动添加为谁的从库

[root@sql51 ~]# redis-trib.rb add-node --slave 192.168.4.57:6357 192.168.4.51:6351

>>> Send CLUSTER MEET to node 192.168.4.57:6357 to make it join the cluster.

**集群移除slave角色(57)**

[root@sql51 ~]# redis-trib.rb del-node 任意主机:端口 被移除的主机id

[root@sql51 ~]# redis-trib.rb del-node 192.168.4.51:6351 f5251bebf750c5beaa48fbbed03e880889e07fcd #57的id

[root@sql51 ~]# redis-trib.rb info 192.168.4.51:6351 #查看集群

**集群移除master角色**

1重新分片:释放占用的hash槽

[root@sql51 ~]#redis-trib.rb reshard 192.168.4.51:6351

[OK] All 16384 slots covered.

How many slots do you want to move (from 1 to 16384)? 4096

What is the receiving node ID? 412a79829c7d0eed812b24959ca123b58990d60e #接受哈希槽的主机id(51)

Please enter all the source node IDs.

Type 'all' to use all the nodes as source nodes for the hash slots.

Type 'done' once you entered all the source nodes IDs.

Source node #1:c2e039c3a388e62ec6c02a8af3b5a8bd53040a65 #要移除的主机id(58)

Source node #2:done #只移除一个主机的最后写done

**2**再移除主机

sql51 ~]# redis-trib.rb del-node 任意主机:端口 被移除的主机id

sql51~]# redis-trib.rb del-node 192.168.4.51:6351 c2e039c3a388e62ec6c02a8af3b5a8bd53040a65 #被移除的主机id58

**Redis 主从复制工作原理**

1. Slave向master发送sync命令

2. Master启动后台存盘进程,并收集所有修改数据命令

3. Master完成后台存盘后,传送整个数据文件到slave

4. Slave接收数据文件(dump.rdb文件).加载到内存中完成首次完全同步

5. 后续有新数据产生,master继续将新的数据收集到的修改命令传送给slave,slave运行命令后完成同步

主从注意事项:启动服务默认为主库、从库需要手动配置、从库默认是只读的、命令操作都是临时,

192.168.4.51:6351> info replication #查看主从配置信息

# Replication

role:master #主库 开启服务默认是主库

connected\_slaves:0 #从库个数为”0”

[root@sql52 ~]# redis-cli -h 192.168.4.52 -p 6352 #临时手动设置为从库，重启服务失效

192.168.4.52:6352> slaveof 192.168.4.51 6351

OK

192.168.4.52:6352> info replication

# Replication

role:slave #从库

master\_host:192.168.4.51 #主库地址

master\_port:6351 #主库端口

[root@sql52 ~]# vim /etc/redis/6379.conf #永久配置从库:修改配置文件

282 # slaveof <masterip> <masterport> #修改前

282 slaveof 192.168.4.51 6351 #修改后

[root@sql52 ~]# /etc/init.d/redis\_6379 stop #停止服务

[root@sql52 ~]# /etc/init.d/redis\_6379 start #开启服务

192.168.4.52:6352> slaveof no one #临时手动设置为主库，重启失效,恢复从库

OK

192.168.4.52:6352> info replication

# Replication

role:master

connected\_slaves:0

282# slaveof 192.168.4.51 6351 #注释这行，永久配置主库，注释配置文件中

[root@sql52 ~]# vim /etc/redis/6379.conf #主库设置连接密码的主从配置

501 requirepass 123456 设置密码为123456

[root@sql52 ~]# vim /etc/redis/6379.conf #从库密码设置

289 # masterauth 123456 #设置密码为123456

**哨兵模式**

 监视当前主库,当主库宕机后,从对应的从库里选举新的主库是一种服务,可以在主库上运行.也可以单独拿一台设备运行使用哨兵模式,如果主服务器设置了密码.各从服务器配置文件中设置了 masterauth 123456(指定连接主服务器密码),因为从服务器没有设置相同连接密码(或者没设置连接密码)的主服务器会漂移,但是所有的从服务器状态都会是down。

**需要安装redis软件包：**创建服务主配置：文件有配置文件模板

[root@sql51 ~]# ls /root/redis-4.0.8/sentinel.conf

/root/redis-4.0.8/sentinel.conf #模板

[root@sql55 ~]# vim /etc/sentinel.conf #直接编辑空白文件,添加内容也可

bind 0.0.0.0 #实验中未配置

sentinel monitor sql51 192.168.4.51 6351 1 #添加当前主库

sentinel auth-pass 123456 #设置主库连接密码

格式：#sentinel monitor master主机名 masterip地址 master端口 票数 #sentinel auth-pass 连接主库密码

#票数:有几台哨兵主机连接不上主库时切换主库,需要这里只有一台哨兵服务器主机

[root@sql55 ~]# redis-sentinel /etc/sentinel.conf #启动服务

**redis数据持久化(RDB/AOF)：**服务在运行的过程中,自动把内存里的数据存储到物理硬盘

持久化（Persistence）即把数据（如内存中的对象）保存到可永久保存的存储设备中（如磁盘）。

 持久化的主要应用是将内存中的对象存储在数据库中，或者存储在磁盘文件中、XML数据文件中等等。

 持久化是将程序数据在持久状态和瞬时状态间转换的机制。

 JDBC就是一种持久化机制。文件IO也是一种持久化机制。

日常持久化的方法，例如：将鲜肉冷藏，吃的时候再解冻的方法也是。将水果做成罐头的方法也是。

**持久化之RDB介绍：**Redis数据库文件,全称Redis DataBase

 数据持久化方式之一按照指定时间间隔,

将内存中的数据集快照写入硬盘，术语叫Snapshot快照

恢复时,将快照文件直接读入内存

**RBD优点：**高性能的持久化实现--创建一个子进程来执行持久化.先将数据写入临时文件.持久化过程结束后,再用这个临时文件替换上次持久化好的文件;过程中主进程不做任何IO操作，比较适合大规模数据恢复,且对数据完整性要求不是非常高的场合 **RDB缺点：**意外宕机时,最后一次持久化的数据会丢失

[root@sql57 ~]# vim /etc/redis/6379.conf **#配置文件中启用RDB**

save "" #如果注释则禁用RDB

Save 900 1 #900秒内且至少有1次修改,进行存盘

Save 300 10 #300 秒内且至少有10次修改,则进行存盘

Save 60 10000 #60 秒内且至少有10000次修改则进行存盘

[root@sql57 ~]# cp dump.rdb /var/lib/redis/6379/ #以ROB文件恢复数据

[root@sql57 ~]# /etc/init.d/redis\_6379 start

Starting Redis server...

**持久化之AOF：**为了弥补RDB的缺点,出现AOF

只做追加操作的文件.Append Only File,只记录redis服务所有写的操作,默认未启用

 不断的将新的写操作追加到文件的末尾,就是一个文本文件,使用cat命令可以查看文件内容

[root@sql56 ~]# vim /etc/redis/6379.conf **#配置文件中启用AOF**

673 appendonly yes #启用aof ,默认no

677 appendfilename "appendonly.aof" #指定文件名

以下为AOF文件记录写操作的方式

702 # appendfsync always #有新写操作立即记录,记录命令同时把数据写入dump.rdb

703 appendfsync everysec #3选一 #每秒记录一次,记录命令同时把数据写入dump.rdb

704 # appendfsync no #从不记录,

[root@sql57 ~]# rm -rf /var/lib/redis/6379/\* #以下为AOF文件恢复数据

[root@sql57 ~]# cp appendonly.aof /var/lib/redis/6379/ #appendonly.aof从别的机器上拷贝过来的

[root@sql57 ~]# /etc/init.d/redis\_6379 start

Starting Redis server...