Nosql

主流的 RDBMS 软件 – Oracle – DB2 – WS -sqlserver – MySQL

主流NoSQL 服务软件 – Redis – MongoDB – Memcached – CouchDB – Neo4j – FlockDB

Redis 介绍

– Remode DIctionary Server( 远程字典服务器 )

– 使用 C 语言编写的,遵守 BSD 的开源软件

– 是一款高性能的 (Key/Values) 分布式内存数据库

– 并支持数据持久化的 NoSQL 数据库服务软件

常用操作指令

– Set keyname keyvalue // 存储

– get keyname // 获取

– Select 数据库编号 0-15 // 切换库

– Keys \* // 打印所以变量

– Keys a? // 打印指定变量

– Exits keyname // 测试是否存在

// 查看生存时间

– ttl keyname

– type keyname // 查看类型

– Set keyname keyvalue // 存储

– get keyname // 获取

– Select 数据库编号 0-15 // 切换库

– Keys \* // 打印所以变量

– Keys a? // 打印指定变量

– Exits keyname // 测试是否存在

// 查看生存时间

– ttl keyname

– type keyname // 查看类型

– move keyname dbname // 移动变量

– expire keyname 10 // 设置有效时间

// 删除变量

– del keyname

– flushall // 删除所有变量

– save // 保存变量

– shutdown // 关闭服务

配置文件常用配置选项

– port 6379 // 端口

– bind 127.0.0.1 //IP 地址

– tcp-backlog 511 //tcp 连接总数

– timeout 0 // 连接超时时间

– tcp-keepalive 300 // 长连接时间

– daemonize yes // 守护进程方式运行

– databases 16 // 数据库个数

– logfile /var/log/redis\_6379.log //pid 文件

– maxclients 10000 // 并发连接数量

– dir /var/lib/redis/6379 // 数据库目录

设置连接密码

• 设置密码

– grep -n requirepass /etc/redis/6379.conf

501:requirepass 123456

– [root@localhost redis-4.0.8]# redis-cli

– 127.0.0.1:6379> ping

– (error) NOAUTH Authentication required.

– 127.0.0.1:6379> auth 123456 // 输入密码

– OK

– 127.0.0.1:6379> ping

– PONG

– 127.0.0.1:6379>

部署 LNMP+Redis

安装软件包

• 安装源码 nginx 和 PHP

]# yum -y install gcc gcc-c++ pcre-devel zlib-devel

]# tar -zxf nginx-1.12.2.tar.gz

]# cd nginx-1.12.2

]# ./configure --prefix=/usr/local/nginx

]# make

]# make install

[root@bogon ~ ]# yum -y install php-common

[root@bogon ~ ]# rpm -ivh php-fpm-5.4.16-42.el7.x86\_64.rpm

修改配置文件

• 修改 nginx 服务主配置文件

\[root@bogon ~]# vim /usr/local/nginx/conf/nginx.conf

location ~ \.php$ {

root

html;

fastcgi\_pass 127.0.0.1:9000;

fastcgi\_index index.php;

fastcgi\_param SCRIPT\_FILENAME $document\_root/$fastcgi\_script\_name;

include

fastcgi\_params;

}

:wq

启动服务

• 启动 nginx 服务

[root@bogon nginx-1.12.2]# /usr/local/nginx/sbin/nginx

[root@bogon nginx-1.12.2]# netstat -utnalp | grep :80

• 启动 php-fpm 服务

[root@bogon lnmp]# systemctl start php-fpm

[root@bogon lnmp]# netstat -utnalp | grep :9000

测试配置

• 测试文件

[root@bogon lnmp]# vim /usr/local/nginx/html/test.php

<?php

echo "hello world!!!";

?>

:wq

[root@bogon lnmp]#

• 访问 nginx 服务

[root@bogon lnmp]# elinks --dump http://localhost/test.php

hello world!!!

运行 Redis 服务

• 装包并启动服务

# tar -xzf redis-4.0.8.tar.gz

# cd redis-4.0.8

#make

#make install

#./utls/install\_server.sh // 初始化

# /etc/init.d/redis\_6379 start

配置 php 支持 Redis

• 安装 php 扩展

#yum -y install autoconf

#yum -y install automake

#rpm -ivh php-devel-5.4.16-42.el7.x86\_64.rpm

# tar -zxf php-redis-2.2.4.tar.gz

# cd phpredis-2.2.4/

[root@bogon phpredis-2.2.4]# /usr/bin/phpize

Configuring for:

PHP Api Version:

20100412

Zend Module Api No:

20100525

Zend Extension Api No: 220100525

# ./configure --with-php-config=/usr/bin/php-config

[root@bogon phpredis-2.2.4]# make

[root@bogon phpredis-2.2.4]# make install

[root@bogon ~]# vim /etc/php.ini

extension\_dir = "./"

extension = "redis.so"

测试配置

• 查看是否支持模块

[root@bogon ~]# php -m | grep -i redis

redis

• 编写测试文件

[root@bogon bin]# cat /usr/local/nginx/html/redis.php

<?php

$redis = new redis();

$redis->connect('127.0.0.1',6379);

$redis->set('redistest','666666');

echo $redis->get('redistest');

?>

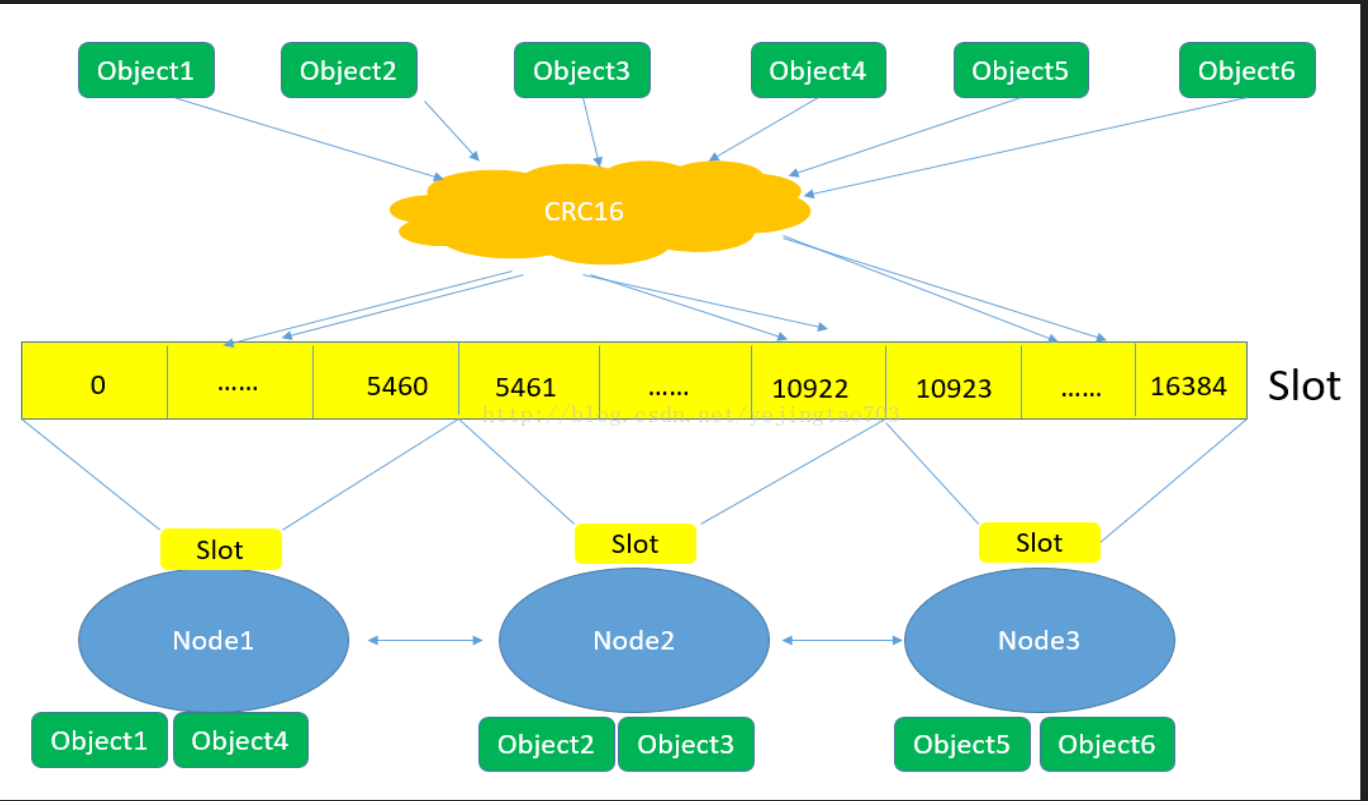
• 访问 nginx 服务

[root@bogon bin]# elinks --dump http://localhost/redis.php

第二天

Redis集群

六台 Redis 服务器



装包

• 在 6 台主机上做如下配置,并运行 redis 服务

#yum -y install gcc gcc-c++

#tar -zxvf redis-4.0.8.tar.gz

#cd redis-4.0.8/

#make

#make install

#./utils/install\_server.sh

• 修改配置文件

#vim /etc/redis/redis.conf

bind IP 地址

// 只写物理接口 IP 地址

daemonize yes

// 守护进程方式运行

port xxxx

// 端口号不要使用默认的 6379

cluster-enabled yes

// 开启集群

cluster-config-file nodes.conf

// 集群的配置文件不要使用默认的名称

cluster-node-timeout 5000

// 请求超时 5 秒

创建集群

• 在任意一台 redis 服务器上,执行创建集群的脚本都可以

– 不是 ruby 脚本运行环境

– 创建集群

#yum -y install ruby rubygems

#rpm -ivh --nodeps ruby-devel-2.0.0.648-30.el7.x86\_64.rpm

#gem install redis-3.2.1.gem

#cd redis-3.2.0/src/

#./redis-trib.rb create --replicas 1 \

192.168.4.51:6351 \ --replicas 1 表示 自动为每一个 master 节点分配一个 slave 节点

192.168.4.52:6352 \

192.168.4.53:6353 \

192.168.4.54:6354 \

192.168.4.55:6355 \

192.168.4.56:6356

查看集群信息

• 任意一台主机访问本机的 redis 服务,查看即可

– redis-cli -c -h IP 地址 -p 端口

> cluster nodes # 查看本机信息

> cluster info # 查看集群信息

测试集群

• 在其中任意一台上存储数据 ,在其他任意一台上都

可以获取数据。

存数据 set name jerry

取数据 get name

管理命令

redis-cli 命令

• 查看命令帮助

– redis-cli -h

• 常用选项

– -h IP 地址

– -p 端口

– -c 访问集群

redis-trib.rb 脚本

• 语法格式

– Redis-trib.rb

选项 参数

• 选项

– add-node 添加新节点

– check 对节点主机做检查

– reshard 对节点主机重新分片

– add-node --slave 添加从节点主机

– del-node 删除节点主机

集群节点选举测试

• 把是 master 角色主机上的 Redis 服务 停止

#redis-cli -h ip -p 端口 shutdown

• 对应是 slave 角色主机 是否能自动升级为 master

> cluster status

注意: 原先是 master 服务启动后 身份是当前 master 的 slave

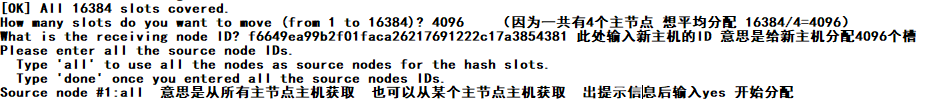
添加新节点

将新主机添加到集群中

– # ./redis-trib.rb add-node 新主机 Ip: 端口 192.168.4.51:6351

• 手动对集群进行重新分片

– src]# ./redis-trib.rb reshard 192.168.4.51:6351



• 查看分配的槽位

– src]# ./redis-trib.rb check 192.168.4.51:6351

• 对添加为从节点的主机做如下配置

– 装包 修改配置文件 启动 redis 服务

• 添加从节点

# ./redis-trib.rb add-node --slave --master-id id 值 从节点 ip: 端口 192.168.4.51:6351

↑如果不指定主节点的 id 的话,会把新节点 随机添加为 从节点 最少的主的从。

移除从节点

• 从节点主机没有槽位范围,直接执行移除命令即可

– redis-trib.rb del-node 192.168.4.51:6351 被移除主机的 ID

第三天

Redis 主从复制

主从复制工作原理

• 工作原理

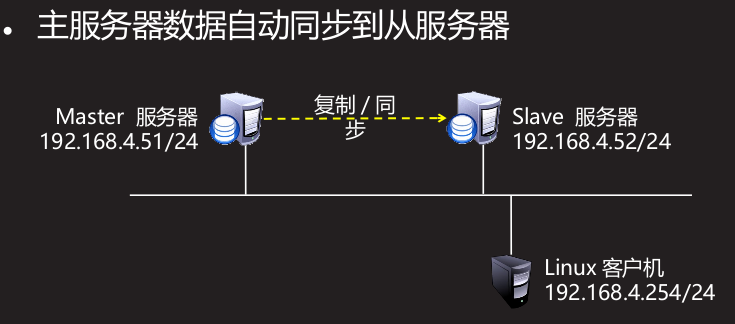
– Slave 向 maste 发送 sync 命令

– Master 启动后台存盘进程,同时收集所有修改数据命令

– Master 执行完后台存盘进程后,传送整个数据文件到slave 。

– Slave 接收数据文件后,将其存盘并加载到内存中完成首次完全同步

– 后续有新数据产生时, master 继续将新的所以收集到的修改命令依次传给 slave ,完成同步。



配置从库

• 配置从库 192.168.4.52/24

– redis 服务运行后,默认都是 master 服务器

– 修改服务使用的 IP 地址 bind 192.168.4.X

[root@redis52 ~]# redis-cli -h 192.168.4.52

192.168.4.52:6379> info replication // 查看主从配置信息

# Replication

role:master

connected\_slaves:0

......

192.168.4.52:6379> SLAVEOF 192.168.4.51 6379 命令行指定主库SLAVEOF 主库 IP 地址 端口号

OK

192.168.4.52:6379> info replication

# Replication

role:slave

master\_host:192.168.4.51

master\_port:6379

反客为主

• 反客为主

– 主库宕机后,手动将从库设置为主库

[root@redis52 ~]# redis-cli -h 192.168.4.52

192.168.4.52:6379> SLAVEOF no one // 设置为主库

OK

192.168.4.52:6379> info replication

# Replication

role:master

哨兵模式

• 哨兵模式

– 主库宕机后,从库自动升级为主库

– 在 slave 主机编辑 sentinel.conf 文件

– 在 slave 主机运行哨兵程序

[root@redis52 ~]# vim /etc/sentinel.conf

sentinel monitor redis51 192.168.4.51 6379 1

:wq

sentinel monitor 主机名 ip 地址 端口 票数

主机名:自定义

IP 地址: master 主机的 IP 地址

端 口: master 主机 redis 服务使用的端口

票 数:主库宕机后, 票数大于 1 的主机被升级为主库

[root@redis52 ~]# redis-sentinel /etc/sentinel.conf

配置带验证的主从复制

• 配置 master 主机

– 设置连接密码 ,启动服务,连接服务

[root@redis51 ~]# sed -n '70p;501p' /etc/redis/6379.conf

bind 192.168.4.51

requirepass 123456 // 密码

[root@redis51 ~]#

[root@redis51 ~]# /etc/init.d/redis\_6379 start

Starting Redis server...

[root@redis51 ~]# redis-cli -h 192.168.1.111 -a 123456 -p 6379

192.168.4.51:6379>

• 配置 slave 主机

– 指定主库 IP ,设置连接密码,启动服务

[root@redis52 ~]# sed -n '70p;282p;289p' /etc/redis/6379.conf

bind 192.168.4.52

slaveof 192.168.4.51 6379 // 主库 IP 与端口

masterauth 123456 // 主库密码

[root@redis52 ~]#

[root@redis52 ~]# /etc/init.d/redis\_6379 start

Starting Redis server...

[root@redis52 ~]# redis-cli -h 192.168.4.52

192.168.4.52:6379> INFO replication

# Replication

role:slave

master\_host:192.168.4.51

master\_port:6379

RDB 介绍

• 全称 Reids DataBase

– 数据持久化方式之一

– 在指定时间间隔内,将内存中的数据集快照写入硬盘

。

– 术语叫 Snapshot 快照。

– 恢复时,将快照文件直接读到内存里。

相关配置参数

• 文件名

– dbfilename “dump.rdb” // 文件名

– save “” // 禁用 RDB

• 数据从内存保存到硬盘的频率

– save 900 1 // 900 秒内且有 1 次修改存盘

– save 300 10//300 秒内且有 10 次修改存盘

– save 60 10000//60 秒内且有 10000 修改存盘

• 手动立刻存盘

// 阻塞写存盘

– > save

– > bgsave // 不阻塞写存盘• 压缩

– rdbcompression yes | no

• 在存储快照后,使用 crc16 算法做数据校验

– rdbchecksum yes|no

• bgsave 出错停止写操作 , 对数据一致性要求不高设置为 n o

– stop-writes-on-bgsave-error yes|no

使用 RDB 文件恢复数据

• 备份数据

– 备份 dump.rdb 文件到其他位置

– ~]# cp 数据库目录 /dump.rdb

备份目录

• 恢复数据

– 把备份的 dump.rdb 文件拷贝回数据库目录 , 重启 redis 服务

– cp 备份目录 /dump.rdb 数据库目录 /

– /etc/redid/redis\_ 端口 start

RDB 优点 / 缺点

• RDB 优点

– 持久化时, Redis 服务会创建一个子进程来进行持久化,会先将数据写入到一个临时文件中,待持久化过

程都结束了,再用这个临时文件替换上次持久化好的文件;整个过程中主进程不做任何 IO 操作,这就确保

了极高的性能。

– 如果要进程大规模数据恢复,且对数据完整行要求不是非常高,使用 RDB 比 AOF 更高效。

• RDB 的缺点

– 意外宕机,最后一次持久化的数据会丢失。

第四天

MongoDB

软件介绍

• 介于关系数据库和非关系数据库之间的产品

– 一个基于分布式文件存储的数据库。

– 由 C++ 语言编写。旨在为 WEB 应用提供可扩展的高性能数据存储解决方案。

– MongoDB 将数据存储为一个文档,数据结构由键值(key=>value) 对组成。

– MongoDB 文档类似于 JSON 对象。字段值可以包含其他文档,数组及文档数组。

装包

• 免安装,解压后即可使用

[root@bogon ~]# mkdir /usr/local/mongodb

[root@bogon ~]# tar -zxf mongodb-linux-x86\_64-rhel70-3.6.3.tgz

[root@bogon ~]#

[root@bogon ~]# cp -r

mongodb-linux-x86\_64-rhel70-3.6.3/bin /usr/local/mongodb/

[root@bogon ~]# cd /usr/local/mongodb/

[root@bogon mongodb]# mkdir etc

[root@bogon mongodb]# mkdir log

[root@bogon mongodb]# mkdir -p data/db

创建配置文件

• 手动创建服务主配置文件

[root@bogon ~]# vim mongodb.conf

logpath=/usr/local/mongodb/log/mongodb.log

logappend=true# 追加的方式记录日志信息

dbpath=/usr/local/mongodb/data/db # 数据库目录

fork=true # 守护进程方式运行

启动服务

• 启动服务

– ]# ./bin/mongod -f /usr/local/mongodb/etc/mongodb.conf

• 查看进程

– ]# ps -C mongod

• 查看端口

– ]# netstat -utnlp | grep :27017

连接服务

• 本地连接,默认没有密码

[root@bogon ~]# /usr/local/mongodb/bin/mongo

MongoDB shell version v3.6.3

connecting to: mongodb://127.0.0.1:27017

MongoDB server version: 3.6.3

......

> show dbs // 显示已有的库

admin 0.000GB

config 0.000GB

local 0.000GB

> exit # 断开连接

bye

[root@bogon ~]#

数据库管理

• 库管理命令: 查看 创建 切换 删除

– show dbs 查看已有的库

显示当前所在的库

– db

– use 库名 切换库,若库不存在的话 自动延时创建库

– show collections 或 show tables 查看库下已有的集合

– db.dropDatabase()

删除当前所在的库

• 数据库名可以是满足以下条件的任意 UTF-8 字符串。

– 不能是空字符串( "") 。

– 不得含有 ' ' (空格 ) 、 . 、 $ 、 / 、 \ 和 \0 ( 空字符 )。

– 应全部小写。

– 最多 64 字节。

集合管理

• 集合管理命令:查看 创建 删除

– show collections 或 show tables # 查看集合

# 删除集合

– db. 集合名 .drop()

– db. 集合名 .save({'',''})

在时,创建并添加文档

> db.user.save({'name':'bob','age':'21'})

WriteResult({ "nInserted" : 1 })

• 合法的集合名

– 集合名不能是空字符串 "" 。

– 集合名不能含有 \0 字符(空字符 ) ,这个字符表示集

合名的结尾。

– 集合名不能以 "system." 开头,这是为系统集合保留的

前缀。

– 用户创建的集合名字不能含有保留字符。

文档基本管理 ( 续 1)

• 文档管理命令: 查看 统计 添加 删除

– db. 集合名 .find()

– db. 集合名 .count()

– db. 集合名 .insert({“name”:”jim”})

– db. 集合名 .find({ 条件 })

– db. 集合名 .findOne() # 返回一条文档

– db. 集合名 .remove({}) # 删除所有文档

– db. 集合名 .remove({ 条件 }) # 删除与条件匹配的所有文档

• 插入记录

>db.col.insert(

{ title: 'MongoDB 教程 ',

description: 'MongoDB 是一个 Nosql 数据库 ',

by: 'MongoDB 中文网 ',

url: 'http://www.mongodb.org.cn',

tags: ['mongodb', 'database', 'NoSQL'],

likes: 100

}

)

>db.col.remove({‘ttle’:‘MongoDB 教程’ }) 删除记录

数据导入导出

数据导出

• 语法格式 1

– #mongoexport [--host IP 地址 --port 端口 ]

-d 库名 -c 集合名 -f 字段名 1, 字段名 2

--type=csv > 目录名 / 文件名 .csv

• 语法格式 2

– #mongoexport --host IP 地址 --port 端口

- 库名 -c 集合名 -q ‘{ 条件 }’ -f 字段名 1 ,字段名 2

--type=csv > 目录名 / 文件名 .csv

注意:导出为 csv 格式必须使用 -f 指定字段名列表 !!!

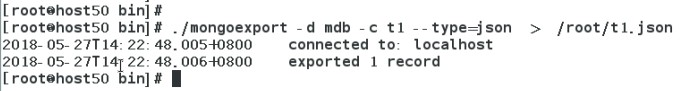
数据导出(续 1 )

• 语法格式 3

#mongoexport [ --host IP 地址 --port 端口 ]

-d 库名 -c 集合名 [ -q ‘{ 条件 }’ –f 字段列表]

--type=json> 目录名 / 文件名 .json





数据导入

• 语法格式 1

– #mongoimport –host IP 地址 – port 端口

-d 库名 – c 集合名

--type=json目录名 / 文件名 .json

• 语法格式 2

– #mongoimport –host IP 地址 – port 端口

-d 库名 – c 集合名

--type=csv --headerline [--drop] 目录名 / 文件名 .csv

注意:导入数据时库和集合不存在时,会创建库和集合后导入数据反之以追加的方式导入数据到集合里,使用— drop 选项可以 删除原有数据后导入新数据 --headerline 忽略标题

数据备份恢复

数据备份

• 备份数据所有库到当前目录下的 dump 目录下

# mongodump [ --host ip 地址 --port 端口 ]

• 备份时指定备份的库和备份目录

# mongodump [ --host ip 地址 --port 端口 ] -d 数据库名 -c 集合名 -o 目录

目录无需事先创建 备份时指定即可!!!

• 查看 bson 文件内容

#bsondump ./dump/bbs/t1.bson数据恢复

• 语法格式

– mongorestore --host IP 地址 --port 端口 -d 数据库名 [ -c 集合名 ] 备份目录名

第五天

副本集介绍

• 副本集是什么

– MongoDB 复制是将数据同步在多个服务器的过程。

– 复制提供了数据的冗余备份,并在多个服务器上存储数据副本,提高了数据的可用性, 并可以保证数据的

安全性。

– 复制还允许您从硬件故障和服务中断中恢复数据

• 副本集工作过程

– mongodb 的复制至少需要两个节点。其中一个是主节点,负责处理客户端请求,其余的都是从节点,负

责复制主节点上的数据。

– mongodb 各个节点常见的搭配方式为:一主一从、一主多从。

– 主节点记录在其上的所有操作 oplog ,从节点定期轮询主节点获取这些操作,然后对自己的数据副本执行

这些操作,从而保证从节点的数据与主节点一致。

副本集实现方式

• Master-Slave 主从复制

– 实现数据同步只需要在某一台服务器启动时加上 "-master" 参数,以指明此服务器的角色是 primary ;另一

台服务器加上 "-slave" 和 "-source" 参数,以指明此服务器的角色是 slave 。

• 主从复制的优点如下:

– 从服务器可以执行查询工作,降低主服务器访问压力。

– 在从服务器执行备份,避免备份期间锁定主服务器的数据。

– 当主服务器出现故障时,可以快速切换到从服务器,减少当机时间。

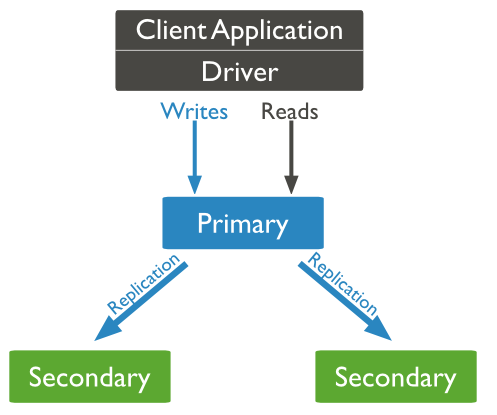
• Replica Sets 复制集

– MongoDB 在 1.6 版本对开发了新功能 replica set,这比之前的 replication 功能要强大一 些,增加了

故障自动切换和自动修复成员节点,各个 DB 之间数据完全一致,大大降低了维 护成功。使用 replica set

故障切换完全自动。

– Replica Sets 的结构类似一个集群,完全可以把它当成一个集群,因为它确实与集群实现的作用是一样的:如果其中一个节点出现故障,其他节点马上会将业务接管过来而无须停机操作



运行服务

• 启动服务时,指定主机所在副本集名称

– 副本集成员间使用相同的副本集名称

– --replSet rs1 // 指定副本集名称

[root@server0 ~]#mkdir /data/db

[root@server0 ~]#./mongod --bind\_ip 192.168.4.61 \

--logpath=/var/log/mongod.log --replSet rs1 &

[root@server0 ~]# jobs

Running .........

配置节点信息

• 在任意一台主机连接 mongod 服务,执行如下操作

[root@server0 ~]# ./mongo --host 192.168.4.61

config = {

\_id:"rs1",

members:[

{\_id:0,host:“IP 地址 : 端口 "},

{\_id:1,host:“IP 地址 : 端口 "},

{\_id:2,host:“IP 地址 : 端口 "}

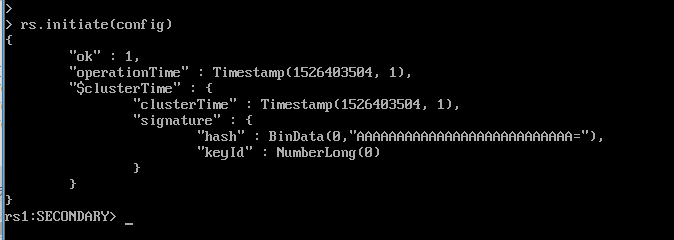
]

};

初始化 Replica Sets 环境

• 执行如下命令

– >rs.initiate(config)



查看副本集信息

• 查看状态信息

– > rs.status( )

• 查看是否是 master 库

– > rs .isMaster( )

验证副本集配置

• 同步数据验证

– >db.getMongo( ).setSlaveOk( ) 允许从库查看数据

• 自动切换主库验证

– > rs.isMaster( ) 查看是否是主库

关于文档的操作就不再赘述了。