**部署mongodb服务**

介绍

介于关系数据库和非关系数据库之间的产品

一款基于分布式文件存储的数据库，旨在为web应用提供可扩展的高性能数据存储解决方案

将数据存储为一个文档（类似于JSON对象，mysql里的记录），数据结构由键值（key=>values）对组成

支持丰富的查询表达，可以设置任何属性的索引

支持副本集（相当于mysql里面的主从同步），分片

直接存储到硬盘

相关概念

db（库，对应mysql里面的database）

collection/tables（集合，对应mysql里的表）

document（文档，对应mysql里的记录/行）

field（字段，对应mysql里的列）

index（对应mysql里的index）

embedding&linkding（对应mysql里的join）

shard（对应mysql里的partition）

sharding key（对应mysql里的partition key）

搭建MDB服务器

装包（免安装，解压后即可使用）

mkdir /usr/local/mongodb

tar -xf mongodb-linux-x86\_64-rhel70-3.63.tgz

cp -r mongodb0linux0x86\_64-rhel70-3.63/bin /usr/local/mongodb

cd /usr/local/mongodb

mkdir -p etc log data/db

创建配置文件

手动创建服务主配置文件

vim /usr/local/mongodb/etc/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

bind\_ip=192.168.1.79//指定监听的物理接口地址，如果不指定，默认为127.0.0.1

port=27050//指定服务的端口，如果不指定为默认的27017

添加命令目录路径到系统路径中

vim /etc/profile

export PATH=/usr/local/mongodb/bin:$PATH

source /etc/profile

启动服务

./bin/mogod -f /usr/local/mongodb/etc/mongodb.conf

ps -C mongod

netstat -tunlp | grep 27017

连接服务，本机连接，默认没有密码

/usr/local/mongodb/bin/mongo

停止服务

killall mongod

mongod --shutdown -f /usr/local/mongodb/etc/mongodb.conf

指定服务的IP地址和端口号

停止服务

修改配置文件（添加bind和port两项）

启动服务

连接

mongo --host 192.168.1.79 --port 27050

**mongodb基本使用（严格分区大小写）**

常用管理命令

数据库管理

查看

show dbs//查看已有库

db//显示当前库

创建、切换

use 库名//切换库，若库不存在延时创建

查看集合

show collections或show tables//查看库下已有集合

删除库

db.dropdatabase()//删除当前库

数据库命名规范

不能使用空字符串（“”）

不能含有’ '（空格）、.、$、/、\和\0（空字符）

应全部小写

最多64个字节

集合管理（类似于mysql的表）

查看

show collections或show tables//查看集合

删除

db.集合名.drop()//删除集合

创建

db.集合名.save({})//创建集合，集合不存在时，创建并添加文档

集合命名规范

不能是空字符串””

不能含有\0（空字符），此字符表示集合的结尾

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（类似于mysql表记录）

查看

db.集合名.find()

db.集合名.find({查询条件})

db.集合名.findOne()//返回第一条文档

统计

db.集合名.count()//统计有多少文档

添加

db.集合名.insert({“name”:”jim”})

删除

db.集合名.remove({})//删除所有文档

db.集合名.remove({条件})//删除所有的匹配文档

基本数据类型

字符/布尔/空

字符串string

UTF-8字符串都可以表示为字符串类型的数据

{name:”aaa”}

布尔值bool

布尔类型有两个值true和false,{x:true}

空null

用于表示控制或者不存在的字段,{x:null}

数值/数组array

数值

shell默认使用64为浮点型数值。{x:3.14}或{x:3}

numberint（4字节整数）{x:numberint(3)}

numberlong（8字节整数）{x:numberlong(3)}

数组array

数据列表或数据集可以表示为数组

{x:[“a”,”b”,”c”]}

代码/日期/对象

代码

查询和文档中可以包含任何javascript代码

{x:function(){/\*代码\*/}}

日期

日期被存储为自新纪元以来经过的毫秒数，不含时区，{x:new date()}

对象（如果不指定，默认生成，相当于mysql主键）

对象id是一个12字节字符串，是文档的唯一标识{x:ObjectId()}

内嵌/正则表达式

内嵌

文档可以嵌套其他文档，被嵌套的文档作为值来处理

{aaa:

{address:”beijing”,trl:”8888888”,person:”hansy”}}

正则表达式

查询时，使用正则表达式作为限定条件{x:/正则表达式/}

数据导入导出

csv适合导出指定列的，json适合导出所有列的

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

语法格式2：

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

目录必须创建，文件如果存在则会覆盖

数据导入（事先可以没有库和集合，会延时穿件，json格式的重复导入记录会报错，csv格式的会追加导入）

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 [--hearderline] [--drop] 目录名/文件名.csv

必须指定列名否则会报错，默认会把标题也当数据导入

1. 导入数据时，若库和集合不存在，则会创建库和集合后再导入数据
2. 若库和集合已经存在，则以追加的方式导入数据到集合里；
3. 使用--drop选项可以删除原数据后导入新数据,--headerl;ine忽略标题

数据的备份恢复

数据备份

备份数据所有库到当前目录底下的dump目录下

mongodump [--host ip地址 --port 端口]

备份时指定备份的库和备份的目录

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

查看bson文件内容

bsondump ./dump/bbs/t1.json

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] 备份的目录名

**mongodb副本集（也称为mongodb主从复制）**

指在多个服务器上存储数据副本，并实现数据同步

提高数据可用性、安全性，方便数据故障恢复

工作原理：

至少需要两个节点。其中一个主节点，负责处理客户端的请求，其余是从节点，负责复制主节点数据

常见搭配方式：一主一从、一主多从

主节点记录所有操作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

replica sets副本集

从1.6版本开始支持此配置

支持故障自动切换、自动修复成员节点，将宕机期间的数据同步过来，降低运维成本

类似于高可用集群

配置replica sets

运行mongodb服务

启动服务时，指定主机所在副本集名称

--replSet rs1//指定副本集名称（临时）

vim /usr/local/mongodb/etc/mongodb.conf

replSet=rs1（永久，所有主机都加，并重启服务）

配置节点信息（在谁上创建，谁就是主，优先获取得到IP的就是主）

在任意一台主机连接mongodb服务，执行如下操作

config = [\_id:”rs1”,

members:[

{\_id:0,host:”192.168.4.51:27051”,priority:10},

{\_id:1,host:”192.168.4.52:27052”,priority:9},

{\_id:2,host:”192.168.4.53:27053”,priority:9}]};

priority可选，写上的话，数值大的那个是主

初始化replica sets环境

>rs.initiate(config)

查看副本集信息

查看状态信息rs.status()

查看是否为master库rs,isMaster()

验证副本集配置

同步数据数据验证，允许从库查看数据，默认从库不允许进行操作

db.getMongo().setSlaveOk()

自动切换主库验证

rs,isMaster()

恢复独立数据库服务器

配置文件注释并重新启动

#replSet=rs1

**mongodb文档管理**

插入文档

save()

格式

db.集合名.save({key:”值”,key:”值”})

注意

集合不存在时创建集合，然后再插入记录

\_id字段已存在时，修改文档字段值

\_id字段不存在时，插入文档

一次只能写一条文档

insert()

格式

db.集合.insert({key:”值”,key:”值”})

注意

集合不存在时创建集合，然后再插入记录

\_id字段值已存在时，放弃插入

\_id字段值不存在时，插入文档

一次只能写一条文档

插入多条记录

db.集合名.insertMany([

{name:”xi”,age:11},

{name:”a”,age:1}

])

查询文档

查询语法

显示所有行（默认输出20行，输入it可显示后续行）

>db.集合名.find()

显示第一行

>db.集合名.findOne()

指定查询条件并显示指定字段

>db.集合名.find({条件},{定义显示的字段})

>db.user.find({\_id:0,name:1,shell1})//0不显示，1显示

行数显示限制

limit(数字)//显示前几行

>db.集合名.find().limit(3)

skip(数字)//跳过前几行

>db.集合名.find.skip(2)

sort(字段名)//1升序，-1降序

>db.集合名.find().sort(age:1|-1)

匹配条件

简单条件

>db.集合名.find({key:”值”})

>db.集合名.find({key:”值”,key:”值”})//两个条件同时成立时显示

范围比较

$in 在...里

>db.user.find({uid:{$in:[1,6,9]}})

$nin 不在...里

>db.user.find({uid:{$nin:[1,6,9]}})

$or 或

>db.user.find({$or:[{name:”root”,{uid:1}] })

数值比较

$lt $lte $gt $gte $ne

< <= > >= !=

>db.user.find({uid:{$gte:10,$lte40}})

正则匹配

db.user.find({name:/^a/})

匹配空，也可以匹配没有的字符串

db.user.find({name:null,uid:null})

更新文档（修改列的值）

update()

语法格式

>db.集合名.update({条件},{修改的字段})

默认的时候，当条件匹配，会去更新第一个值，并且会删除其他列

多文档更新

语法格式：默认是更新与条件匹配的第1行

>db.user.update({条件},{修改的字段},false,true)

$set/$unset

$set条件匹配时，修改制定字段的值

>db.user.update({条件},{$set:{修改的字段}})

$unset删除与条件匹配文档的字段，删除文档制定的列

>db.集合名.update({条件},{unset:{key:values}})

$inc

$inc条件匹配时，字段值自加或自减

>db.集合名.update({条件},{$inc:{字段名:数字}})

数字为正，自增，为负，自减

$push/$addToSet

$push向数组中添加新元素（可重复添加）

>db.集合名.update({条件},{$push:{数组名:”值”}})

$addToSet避免重复添加

>db.集合名.update({条件},{$addToSet:{数组名:”值”}})

$pop/$pull

$pop从数组头部删除一个元素

>db.集合名.update({条件},{$pop:数字})

1删除数组尾部元素，-1删除数组头部元素

$pull删除数组指定元素

>db.集合名.update({条件},{$pull:{数组名:值}})

删除集合

$drop/$remove

$drop删除集合的同时删除索引

>db.集合名.drop()

$remove删除文档时不删除索引

>db.集合名.remove({条件})//删除与条件匹配的文档

>db.集合名.remove({})//删除所有文档