**以下是mongoDB-v3.07为例，参考文献:** **http://www.cnblogs.com/huangxincheng/archive/2012/02/18/2356595.html**

## 一、mongoDB初始化(安装完成的情况下)

cmd打开命令窗口,切换到安装目录下，如:

cd e:\database\_space\mongodb\bin

执行命令(--dbpath参数是设置db的数据存放路径):

mongod --dbpath=e:\database\_space\mongodb\db

查看打印的信息，其中有:

mongoDB starting:pid=xxx port=27017 …则说明配置成功

## 二、mongoDB连接

切换到mongoDB安装目录:

cd e:\database\_space\mongodb\bin

**执行命令（不连接带参数）:**

mongo

则就会连接到mongodb数据库，默认连接到test数据库

**执行命令（带连接参数，指定连接的IP和端口）：**

mongo 127.0.0.1:27017

**执行命令（指定IP端口和要连接的数据库）:**

mongo 127.0.0.1:27017/admin (注：连接到admin集合（数据库)

## 三、mongoDB使用(完全支持JS)

简介：db是全局对象,管理整个Mongodb数据库,默认为每一条记录增加\_id字段

### 3.1、insert增(其中user为集合，相当于表,如果第一次使用不存在user,则会自动增加user)

**单条数据插入:**

db.user.insert({“id”:0,“name”:”test”,”age”:99});

**批量插入(这里用js的for循环实现):**

var myuser={}

for(var i=1;i<10;i++){

user.id=i

user.name=’name\_’+i

user.age=i

db.user.insert(myuser)

};

### **3.2 remove删**

**不带参数的remove，会删除集合中的所有数据**

db.user.remove({});//注：如果是db.user.remove();会出错：remove needs a query

**带参数的remove,如删除name=name\_1的数据**

db.user.remove({“name”:”name\_1”});

### 3.3 update 改

update有2个参数，第一个参数为条件，第二个为新值

**整体跟新:跟新后新数据为{“name”:”name\_1\_1”}，没有了id,age**

db.user.update({“id”:1},{“name”:”name\_1\_1”});

局部更新,mongodb提供了2个修改器$inc(increase缩写)和$set,$inc:每次修改都会在原基础上增加$inc指定的值,如果没有此key则会自动创建此key.

$inc: 将id为1的age增加30，不影响其他key:

db.user.update({“id”:1},{$inc:{“age”:30}});

$set: 将id为1的age改为30,不影响其他key:

db.user.update({“id”:1},{$set:{“age”:30}});

**upsert操作（指如果根据条件在数据库中查询到记录，则更新记录，如果没有查询到，则新增一条数据，避免了在数据库里面判断是update还是add操作，有点像oracle中的merge语句）,具体使用起来只需要在设置update的第三个参数为true即是upsert操作:**

(修改或新增id等于11的数据):

db.user.update({“id”:11},{“id”:11,”name”:”name\_11”,”address”:”address\_11”,”phone”:”phone\_11”,”photoaddr”:”photoaddr\_11”},true);

**mongoDB中的批量更新,在mongoDB的更新中，如果条件匹配出多条记录，则只会更新第一条，如果想要实现批量更新，只需要设置第四个参数为true即可**:

db.user.update({“id”:{$gt:9}},{$set:{“photoaddr”:””}},true,true);

### 3.4 find 查

**查询集合中所有数据**：

db.user.find();

**条件查询,与常用的逻辑一样:**

1. >,>=,<,<=,!=
2. And ,or ,in,notIn

在mongoDB中都已经封装好了：

1. $gt,$gte,$lt,$lte,$ne
2. $or,$in,$nin

**$gt,(id大于8的)：**

db.user.find({“id”:{$gt:8}});

**$gte,(id大于等于8的)：**

db.user.find({“id”:{$gte:8}});

**$le,(id小于8的)：**

db.user.find({“id”:{$lt:8}});

**$lte,(id小于等于8的)：**

db.user.find({“id”:{$lte:8}});

**and(与逻辑,注:无关键字),(id等于8并且name等于name\_8的)：**

db.user.find({“id”:8,”name”:”name\_8”});

**$or(或逻辑),(id等于8或id等于9的)：**

db.user.find({$or:[{“id”:8},{“id”:9}]});

**$in(包含逻辑):**

(id包含在8,9,10之中的)：

db.user.find({“id”:{$in:[8,9,10]}});

(name包含在name\_8,name\_9,name\_10之中的)：

db.user.find({"name":{$in:["name\_8","name\_9","name\_10"]}});

**$nin(不包含逻辑)，(id不存在于8,9,10之中的)：**

db.user.find({“id”:{$nin:[8,9,10]}});

**正则表达式法,(name以name开头且以10结尾的)：**

db.user.find({“name”:/^name/,”name”:/10$/});

**$where关键字(可以使用js)：**

(匹配name等于name\_10的)：

db.user.find({$where:function(){return this.name==’name\_10’}});

(匹配name等于name\_9或等于name\_10的)：

db.user.find({$where:function(){

return (this.name==”name\_9” || this.name==”name\_10”)

}});

### 3.5聚合函数(count,distinct,group,mapReduce)

#### **3.5.1 count(数量):**

**查询集合中的总数:**

db.user.count();

**带参数的count(查询name以name开头的数量):**

db.user.count({“name”:/name/});

**(查询name在name\_10,name\_11之中的数量)：**

db.user.count(“name”:{$in:[“name\_10”,”name\_11”]});

**(查询name等于name\_10或id=11的数量)：**

db.user.count({$or:[{“name”:”name\_10”},{“id”:11}]});

#### **3.5.2 distinct(去重，指定的字段的值不能重复)：**

db.user.distinct(“name”);

#### **3.5.3 group(分组):**

group有一个参数，参数类型json对象,此json对象中共有5个键,分别是key(必选),initial(必选),$reduce(必选),finalize(可选),condition(可选).

**key:**

用于定义要分组的字段，如:”key”:{“age”:true},意思为以age分组。

**initial:**

每一组都分享的“初始化函数”，比如age=20的value的集合list分享了一个initial函数，同样，age=22的value的集合list也分享一个initial函数。如：”initial”:{“names”:[]}，意思是分组后的值存放到names的数组里。

**$reduce:**

函数，其第一个参数是当前文档对象（相当于当前行数据），第二个参数是上一次function操作的累计对象,第一次时为initial中的{“names”:[]}对象,有多少个文档，$reduce就会执行多少次，比如说:按age分组，age=20的一组有2个，则会调用2次，用法:”$reduce”:function(currRow,prevObj){prevObj.names.push(currRow.name)};

**finalize:**

函数，此函数有一个参数，是每一组要输出的对象，每一组文档执行完后，都会触发此方法，那么可以用此函数来增加额外字段的显示，如增加count属性，用法:

“finalize”:function(outObj){

// 动态根据names的长度来添加count属性

outObj.count= outObj.names.length;

}

**condition:**

条件语句,在分组的时候可以用此参数来限制数据的条件语句，用法和平时条件一样，如:”condition”:{“age”:{$gte:10}};//即取age大于等于10的数据进行分组。

**group总体用例：**

将所有数据按age分组:

db.user.group({

“key”:{“age”:true},

“initial”:{“names”:[]},

“$reduce”:function(row,out){

out.names.push(row.name);

}

});

将id大于等于10的数据分组且在每一组中增加count信息:

db.user.group({

“key”:{“age”:true},

“initial”:{“names”:[]},

“$reduce”:function(row,out){

out.names.push(row.name);

},

“finalize”:function(out){

out.count=out.names.length;

},

“condition”:{“id”:{$gte:10}}

});

### 3.6 mapReduce

mapReduce其实是一种编程模型，用在分布式计算中，其中有一个”map”函数，一个”reduce”函数。

**map函数:**

这个称为映射函数，里面会调用emit(key,value)，集合会按照你指定的key进行映射分组。

**reduce函数:**

这个称为简化函数，会对map分组后的数据进行分组简化，注意：在reduce(key,value)中的key就是emit中的key，vlaue为emit分组后的emit(value)的集合，这里也就是很多{"count":1}的数组。

**mapReduce：**

这个就是最后执行的函数了，参数为map，reduce和一些可选参数。具体看图可知。

**mapReduce函数的用法(这个还不太清楚)**

创建map 函数:

function map(){

emit(this.age,{coutn:1});//按age分组

};

创建reduce函数:

function reduce(key,value){

var r = {count:0};

for(var i=0;i<value.length;i++){

r.count+=value[i].count;

}

return r;

};

调用mapReduce：

db.user.mapReduce(map,reduce,{“out”:”myResult”});

mapReduce执行结果的解析:

result:存放结果的集合名

input:传入文档的个数

emit:emit函数调用的次数

reduce:reduce函数调用的次数

output:返回文档的个数

ok:执行状态,1:成功，0:失败

从mapReduce执行后的结果集合中取数据:

db.myResult.find();

### **3.7 mongoDB中的游标**

定义一个游标:

var cur = db.user.find({“id”:{$gte:10});

迭代游标获取数据:

cur.forEach(function(row){

print(“id=”+row.id+”;name=”+row.name);

});

### 3.8 mongoDB中的分页

mongoDB中的分页用skip(num)和limit(num)来组合实现,skip(num)指跳过num条记录，limit(num)指限制在num条记录。

创建一个测试分页的函数:

function paging(curPage,pageSize){

// 计算要跳过多少条记录

var skip = (curPage-1)\*pageSize;

// 定义游标

var cur = db.user.find().skip(skip).limit(pageSize);

// 打印游标信息

cur.forEach(function(row){

print(“id=”+row.id+”;name=”+row.name+”;age=”+row.age);

});

}

调用分页测试函数测试：

paging(1,5);//第一页，每页5条

paging(2,5);//第二页，每页5条

…

### 3.9 mongoDB中的排序

排序用sort函数,参数为对象，key是要排序的字段，value的值为1或-1,1为升序，-1为降序，如:

按age升序排序:

db.user.find().sort({“age”:1});

按age降序排序:

db.user.find().sort({“age”:-1});

按age降序，name升序排序:

db.user.find().sort({“age”:-1,”name”:1});

### 3.10 索引操作

为了测试大数据的执行性能，新建user\_big集合，并插入10w个文档（即10w条数据）:

for(var i=1;i<=100000;i++){

db.user\_big.insert({“id”:i,”name”:”name\_”+i});

}

**查看执行计划(性能分析函数):mongoDB中执行计划查看用explain()/explain(true)函数。**

查询id等于100000的数据并查看那执行计划:

db.user\_big.find({“id”:100000}).explain(true);

从执行计划中可以看出执行时间等等信息。

再没有使用索引是，会从10w个文档中以“全表扫面”的方式找到id=100000的数据，这是非常耗时的。

#### 3.10.1 索引的创建(ensureIndex)

这里在id上建立索引,1表示按id升序，-1表示按id降序：

db.user\_big.ensureIndex({“id”:1});

创建索引后，再执行上面的查询：

db.user\_big.find({“id”:100000}).explain(true);

可以从执行计划中看出性能大幅提高。

#### 3.10.2 索引的分类

mongoDB中的索引分为：唯一索引、组合索引、删除索引。

##### 3.10.2.1 唯一索引

建立唯一索引后，一样的键值就不能重复插入了,

在user\_big中的id上创建唯一索引：

db.user\_big.ensureIndex({“id”:1},{“unique”:true});

那么user\_big中id的值是不能重复。

##### 3**.10.2.2 组合索引**

有时候在查询时，查询的条件并不是单一的，可能是多个条件的，如，查询name为”XXX”且address为”yyy”的用户的信息，可以使用name和address的**联合索引**加速查询效率。

**创建联合索引:**

如创建name与address的联合索引：

db.user.ensureIndex({“name”:1,”address”:1});

或者

db.user.ensureIndex({“address”:1,”name”:1});

或者

db.user.ensureIndex({“name”:-1,”address”:1});

或者

db.user.ensureIndex({“address”:-1,”name”:-1});

….等等

可以看到，由于组合方式的不同，位置、升序、降序的不同都是产生的不同的索引，可以用**getIndexes()**来查看所建立的索引。

#### 3.10.3 索引的查看

mongoDB中提供了查看集合中已创建的索引。

如查看user\_big集合中所有的索引：

db.user\_big.getIndexes();

#### 3.10.4 索引的删除(dropIndex,dropIndexes)

随着业务的发展，有些索引会用不到，这个时候，最好删除不需要的索引，因为索引需要实时维护，会降低CUD的操作性能，所以再应用索引的时候，需谨慎使用。

**删除索引：dropIndex函数**,如:

删除在user\_big集合中id上的索引：

db.user\_big.dropIndex({“id”:1});

**删除所有的索引：dropIndexes函数(将会把user\_big集合中的索引都删除掉，除了系统自建的名字为”\_id\_”的索引外)：**

db.user\_big.dropIndexes();

### 3.11 mongoDB主从复制、读写分离、集群部署

mongoDB可以实现双机热备份、读写分离、集群部署，这样的话，如果碰到数据库宕机或者被毁灭性破坏时，减少损害。

#### 3.11.1 主从复制

主从复制模型图，如：

主数据库

从数据库-3

从数据库-2

从数据库-1

创建多个数据库服务器，由于创建虚拟机费时费事，下面就在同一台机器上创建多个数据库当作是多个服务器中的多个数据库了。

**我的第一个mongodb（主数据库）放在E:\database\_space\mongodb 下。**

###### **3.11.1.1 创建从数据库方式一:立即生效模式（mongodb-slave-1）：**

这种创建从数据库的方式，就是在启动从数据库的时就指定了source(主数据库)，就意味着这个从数据库启动后，就会同步主数据库的数据。

第一步：创建从数据库mongodb-slave-1

在E:\database\_space\下创建mongodb-slave-1目录，将mongodb的文件（除了db存放数据的目录不拷贝）拷贝到此目录下，这样就创建了一个从数据库mongodb-slave-1了。

第二步：启动mongodb数据库，并指定它为主数据库

打开window命令窗口，切换到E:\database\_space\mongo\bin下，执行启动命令：

mongod --dbpath=E:\database\_space\mongodb\db --master

注：上面命令中，参数dbpath指数据库的数据存放路径，参数master指定该数据库为 主数据库。

第三部：启动mongodb-slave-1从数据库

切换到mongodb-slave-1\bin目录下，执行启动命令：

mongod --dbpath=E:\database\_space\mongodb-slave-1\db --port=27018 --slave --source=127.0.0.1:27017

参数解释：

dbpath:数据库数据存放位置

port:指定数据库所用的端口,这里都是在本地，所以用127.0.0.1

slave:指定此数据库为从数据库

source:指定此从数据库的主数据库的IP地址与端口

第四步：连接主数据库(mongodb)和从数据库(mongodb-slave-1)

连接主数据库:mongo 127.0.0.1:27017

连接从数据库:mongo 127.0.0.1:27018

第五步：测试主数据库(mongodb)的数据是否会同步复制到从数据库(mongodb-slave-1)

在两个数据库中执行同样的查询语句后，从结果可以看出，主数据库中的数据已经完全复制到了从数据库了。通过从数据库(mongodb-slave-1)的连接窗口的日志信息中可以看出，从数据库每10s都会从主数据库中同步数据，日志大致是：[replsslave] repl:syncing from host:127.0.0.1:27017。当在主数据库中插入新数据库时，经测试发现从数据库都会实时的同步过来。

###### 3.11.1.2 创建从数据库方式二:后期生效模式(mongodb-slave-2)

这种方式创建从数据库，就是在从数据库启动的时候不指定source（即主数据库），这样的话，就算这个从数据库启动了，也不会同步数据库到此数据库中，如果哪天有需要，可以设置运行设置(source)的命令就可以同步主数据库中的数据库。

当没有指定source（主数据库）启动后，从日志信息中可以看到信息：

no source given,add a mater to local.resources to start replication.

第一步：创建mongodb-slave-2从数据库

在E:\database\_space\下创建mongodb-slave-2目录，将mongodb的文件拷贝到此目录下。

第二步：启动从数据库mongodb-slave-2

打开windows命令窗口，切换到目录E:\database\_space\mongodb-slave-2\bin下，执行启动命令：

mongod –dbpath=E:\database\_space\mongodb-slave-2\db –port=27019 –slave

第三步：连接从数据库mongodb-slave-2

打开windows命令窗口，切换到E:\database\_space\mongodb-slave-2\bin下，执行连接数据库命令：

mongo 127.0.0.1:27019

第四步：测试从数据库mongodb-slave-2是否会同步数据

由于没有指定source（主数据库），所以mongodb-slave-2中并没有同步任何数据。

第五步：动态指定source（即主数据库）

1.在从数据库mongodb-slave-2的连接窗口下执行命令：use local（即切换到local数据库，注：local数据库时mongodb默认创建的数据库，在这个local数据库下有个默认的集合叫sources，我们连接数据库时是默认连接到test数据库）。

2.在local数据库下的sources中插入数据，执行语句：

db.sources.insert({“host”:”127.0.0.1:27017”});

意思即是指定127.0.0.1:27017作为从数据库mongodb-slave-2的主数据库源。

3.指定完主数据库后，切换到test数据库:user test,在执行查询语句，此时会发现主数据 库中的数据已经被同步到mongodb-slave-2从数据库中了。

#### 3.11.2 读写分离

读写分离，顾名思义就是读数据的数据库和写数据库的数据库不是同一个数据库，在mongodb中也能实现读写分离。在默认的情况下，从数据库不支持数据的读取的，但是在驱动中给我们提供了一个叫“slaveOkay”来让我们可以显示的读取从数据库中的数据，以减轻主数据库的性能压力，可以参考相关信息。

### 3**.12 mongoDB副本集集群**

这种也是很好的主从数据库集群，想对于3.11所述的集群方式有两个区别：

1、该集群没有特定的主数据库（在3.11的集群方式是通过master参数指定了主数据库的）。

2、如果哪个主数据库宕机了，集群中就会推选出一个从数据库来作为主数据库，这样就具备了故障自动恢复能力。

从上面的两方面的对比可以看出，副本集方式的集群具有更多优势，那么怎样建立副本集数据库集群呢？

#### 3.12.1 副本集方式建立数据库集群

第一步：集群名称命名

既然要建立集群，就应该为集群取个名字，我这里取名为：ltl。

第二步：启动mongodb-slave-1数据库服务

cmd命令打开windows命令窗口，切换到E:\database\_space\mongodb-slave-1/bin下，执行启动命令：

mongod –dbpath=E:\database\_space\mongodb-slave-1\db --port=27018 --replSet ltl/127.0.0.1:27019

参数描述：  
 dbpath:上面解释过了

port:数据库所在端口

replSet ltl/127.0.0.1:27019:表示让服务器知道ltl中还有其他数据库,并指定127.0.0.1:27019是ltl集群下的另一个服务器

第三步：启动mongodb-slave-2数据库服务

cmd命令打开windows命令窗口，切换到E:\database\_space\mongodb-slave-2/bin下，执行启动命令：

mongod –dbpath=E:\database\_space\mongodb-slave-2\db --port=27019 --replSet ltl/127.0.0.1:27018

参数描述同上第二步。

第四步：初始化“副本集”

1.连接到任意的数据库，并且一定要进入admin集合，如下以连接到mongodb-slave-2(127.0.0.1:27019)为例，命令：

mongo 127.0.0.1:27019/admin

2.进入admin集合后，执行“副本集群初”始化命令:

db.runCommand({“replSetInitiate”:{

“\_id”:”ltl”,

“members”:[

{“\_id”:1,”host”:”127.0.0.1:27018”},

{“\_id”:2,”host”:”127.0.0.1:27019”}

]

}});

第五步：确定哪个是主数据库（加入第四步成功）

当建立好副本式数据库集群后，获取哪个数据库时主数据库的方式有两种：

方法一（不直观）：查看每一个数据库启动的命令窗口中的日志，如果日志中有如下类 似信息：

replSet syncing to:IP:PORT

则可以判断to后面的IP和端口上的数据库就被选为了主数据库。

方法二（较好）：连接没一个数据库，如果连接后出现的命令输入行前缀为 集群 名:PRIMARY，就可以确定当前连接的这个数据库被选作为了主数据库，如：

连接：mongo 127.0.0.1:27019,出现命令行前缀是：ltl:PRIMARY>，就可知127.0.0.1:27019 上的数据库时主数据库。

第六步：测试mongodb-slave-1和mongodb-slave-2是否已经组成集群

经测试，在更新主数据库中的数据后，再从数据库中查询，可以发现数据已经被同步到了从数据库中了。

注：在从数据库中查询数据时可能会出现:”not mater and slaveOk=false….”类似的错误，解决方式就是在主数据库和从数据中执行命令：

db.getMongo().setSlaveOk();

意义就是授权给从数据库，允许在从数据库中显示的查询数据。

#### 3.12.2 在副本集集群中建立仲裁服务器

仲裁服务器，就是指只参与投票选举。仲裁服务器中需要集群中的任何一个服务器。

第一步：创建冲裁服务器mongodb-slave-3

在E：\database\_sapce\下新建mongodb-slave-3目录，将mongodb文件拷贝到该目 录下。

第二步：启动仲裁服务器mongodb-slave-3

切换的E:\database\_sapce\mongodb-slave-3\bin下，运行启动命令：

mongod –dbpath=E:\database\_space\mongodb-slave-3\db –port=27020 –replSet ltl/127.0.0.1:27018

参数说明如3.12.1中所述。

第三步：将仲裁服务器添加到集群中

3.12.1中，已经建立好了集群，并已经知道127.0.0.1:27019（即mongodb-slave-2） 是主数据库服务器，所以需要连接到mongodb-slave-2数据库的admin集合，执行添加 节点命令：

rs.addArb(“127.0.0.1:27020”);

#### 3.12.3 查看集群中所有节点、状态等信息(rs.status())

连接任何集群中的数据库服务器后，执行命令：

rs.status();

就可以查看到集群中的信息。

信息描述：

set:集群名

date:创建日期

members:集群中的成员

members信息中的描述：

name:节点名

stateStr:节点类型（PRIMARY:主数据库，SECONDARY:从数据库，ARBITER:仲裁数据 库）

#### 3.12.4 集群中的自动故障恢复

副本式集群中的故障自动恢复，如果集群中的主数据库服务器发生意外故障，则集群中就会推选出一从数据库服务器来顶替作为主数据库。

在3.12.1,3.12.2,3.12.3中建立好了数据库服务器集群,名称为ltl（数据库服务器数量为3），其中一个仲裁数据库服务器：mongodb-slave-3(27019),一个从数据库服务器：mongodb-salve-1(27018),一个主数据库服务器：mongodb-salve-2(27020)。

测试自动故障恢复：

第一步：找到主数据库服务器（即mongodb-slave-2：27019）,将其连断开后，再到连接的从数据库执行查询语句，会发现命令行前缀的变化：

从 ltl:SECONDARY 变成了 ltl:PRIMARY，由此可知，集群选取了mongodb-slave-1:27018来代替作为主数据库。

### 3.13 mongoDB中集合切换

执行切换命令：

use admin;//切换到admin集合

use test;//切换到test集合

### 3.14 mongoDB分片式集群

当数据量达到T级别以上是，磁盘、内存就比较难应付，曾对大数据存储的解决方案，有了分片技术。

#### 3.14.1 分片技术

在mongoDB中有一种集群，叫分片技术。分片技术采用将集合进行分拆，然后将拆分的数据均摊到几个片上的技术。如下图所示：

分片数据库2

分片数据库1

配置(Config)数据库

路由数据库

客户端

**分片数据库：**指正真存储数据的数据库集群

**路由数据库：**管理分片数据库，会根据管理员设置的“片键”将数据分摊到每个分片数据库中。

**配置(config)数据库：**保存数据和片的对应关系以及相应的配置信息。

#### 3.14.2 mongoDB分片技术应用

首先，准备四个数据库，分别用于两个分片数据库、一个路由数据库、一个配置数据库。

第一步：启动配置(Config)数据库(27017)（数据库之间的配置信息都在这里，所以需最先启动）

cmd命令切换到E:\database\_space\mongodb\bin下，执行启动命令：

mongod --dbpath=E:\database\_space\mongodb\db --port=27017

第二步：启动mongos路由数据库(27018)（注：这里启动的是mongos,而不是mongod了）

cmd命令切换到E:\database\_space\mongodb-slave-1\bin下，执行启动命令：

**mongos** --port=27018 --configdb=127.0.0.1:27017

参数说明：

port:数据库端口

configdb:指定此路由数据库所用的配置(config)数据库地址

第三步：分别启动各个分片数据库(27019,27020)

cmd命令分别切换到各个分片数据库bin目录下，执行启动命令：

mongod –dbpath=E:\database\_space\mongodb-slave-2\db –port=27019

mongod –dbpath=E:\database\_space\mongodb-slave-3\db –port=27020

第四步：在路由数据库中添加分片（将分片数据库交给路由数据库管理）

1. 连接路由数据库（27018），且连接到admin集合,如下：

mongo 127.0.0.1:27018/admin

1. 添加分片数据库（27019,27020），用addShard(),如下所示：

db.runCommand({“addShard”:”127.0.0.1:27019”,”allowLocal”:true});

db.runCommand({“addShard”:”127.0.0.1:27020”,”allowLocal”:true});

第五步：设置片键（让数据库知道怎么切分数据）

连接到路由数据库中

1. 开启数据库分片功能(enablesharding()),这里以开启test数据库为例,如下：

db.runCommand(“enablesharding”:”test”);

1. 指定集合中分片的片键,以下以user集合的name为例，如下：

db.runCommand({“shardcollection”:”test.user”,”key”:{“name”:1}});

#### 3.14.3 mongoDB数据分片信息的查看与测试

(1)在user集合中插入10w条数据，用于测试，如下：

for(var=1;i<=100000;i++){

db.user.insert({“id”:i,”name”:”name\_”+i,”age”:i});

};

(2)查看mongodb中的数据分片情况

执行printShardingStatus();即可查看分片情况,如下：

db.printShardingStatus();

分片信息解析：

shards:当前集群中的分片

databases:有个partitiones:true表示已分区，false表示没有分区

chunks:这里发现集合被分成了几个段，具体可查询相关资料。

### 3.15 mongoDB运维技术

作为mongoDB管理员，常用的主要有4个方面知识：安装部署、状态监控、安全认证、数据备份和恢复。

#### 3.15.1 安装部署

在以前的章节中都是采用console程序来承载的，这种方式再生产环境时并不是最佳的，在mongodb中提供了一个叫“服务寄宿“的模式。

##### 3.15.1.1 “服务寄宿“的开启

第一步：服务安装

以管理员身份打开cmd命令，切换到mongodb的文件目录下，如：

E：\database\_space\mongodb\bin

运行安装命令：

mongod –dbpath=E:\database\_space\mongodb\db – logpath=E:\database\mongodb\log\log.txt –port=27017 –install

参数说明：

logpath:用来记录日志，指定后不会用console来承载日志

install:开启安装服务寄宿（大大简化了操作）

第二步：查看是否安装成功

在windows的服务列表中，查看是否有名字为“MongoDB“的服务，如果有则安装成功，否则安装不成功

第三步：如果上一步没有问题的话

继续执行命令：

net start MongoDB

如果命令行提示“MongoDB服务已经成功启动“，则启动成功

**注：命令参数比较多也比较复杂，可以用 –help来查看帮助，如：**

**mongod --help,mongo –help等等**

#### 3.15.2 状态监控

状态监控可以让我们实时的了解数据库的运行状态、健康状况以及性能调优，在mongodb中提供过了三种状态监控方式：http监视器、serverStatus()、mongostat。

##### 3.15.2.1 http监视器

前面已介绍过，可参考相关资料。

##### 3.15.2.2 serverStatus(函数) 监视

serverStatus()函数可以获取mongodb的服务器统计信息，包括：全局锁、索引、用户操作行为等信息，对于管理员来说很重要，可参考：

<http://www.cnblogs.com/xuegang/archive/2011/10/13/2210339.html>

如：

db.serverStatus();

##### 3.15.2.3 mongostat(命令)监视

前两个监视器是静态统计，不能实时自动查看数据的变化，而mongostat则是动态统计，它会每秒刷新。mongostat是命令，可以在mongodb的bin目录下找到mongostat.exe。

用法如下：

cmd命令切换到mongodb的bin目录下，运行mongostat命令：

mongostat –port 27018

即可实时查看服务器状态信息。

#### 3.15.3 安全认证

作为数据库软件，mongodb也可以像oracle等数据库一样采用用户验证的方法。

（1）、添加用户

mongodb中提供了addUser方法来添加用户。

在admin数据库中添加用户：将会被视为超级管理员，如：

在普通数据库中添加用户：根据参数情况，用户行为、权限不一样。

添加用户，如：

（2）、启动mongodb是必须有 –auth参数（开启auth验证模式登录），如：

mongod –dbpath=E:\database\_space\mongo-slave-1\db –port=27018 –auth

#### 3.15.4 备份和恢复

mongodb中常用的数据备份和恢复的方式有三种：直接复制、mongodump和mongorestore、主从复制。

##### 3.15.4.1 直接复制

这种方式算是最简单的，但是在服务器还在运行的情况下直接复制是由危险的，可复制的出来时，数据一遭到破坏，唯一能保证的就是先关闭数据库服务器，再复制数据，复制完后再重启服务器。

##### 3.15.4.2 mongodump和mongorestore

这种方式是mongodb提供的内置工具，很好用，能保证在不停止服务的情况下直接进行数据复制，mongodumo和mongorestore都可以在bin目录下看到。

**（1）mongodump数据导出**

cmd命令窗口切换到mongodb的bin目录下，执行数据库导出命令：

mongodump –port=27018 –d test –o D:\test\mongodb\backup

上述命令意思为：将端口27018上test数据库中的数据导出到D:\test\mongodb\backup 目录下。

**（2）mongorestor数据导入**

cmd命令窗口切换到mongodb的bin目录下，执行数据导入命令：

mongoresotre –port=27018 –d test –drop D:\test\mongodb\backup\test

上述命令意思为：将D:\test\mongodb\backup\test下的数据导入到端口为27018,数据库 名称为test的数据库中，其中drop参数指：将test数据恢复之前先删除原有数据库里 面的数据，具体可用—help查看帮助。

##### 3.15.4.3 主从复制

主从复制已经在3.11章节中介绍过了，这里不在阐述。

##### 3.15.4.4 小结

上述三种数据复制的方法都不能保证获取数据的实时性，在备份的时候可能还有数据在内存中还没有被写到磁盘上，这时就会出现数据不准确了。那么怎么解决这个问题呢，mongodb允许暴力地将数据刷到磁盘上。mongodb提供了fsync+lock机制来实现，fsync+lock首先会把缓冲区的数据暴力刷入到磁盘，然后在数据库加个锁，其他的所有写操作都全部阻塞，知道fsync+lock被释放为止。

使用方法：

Step1:暴力刷数据且加锁

db.runCommand({“fsync”:1,”lock”:1});

step2：备份数据

mongodump …..

step3:释放锁

db.#cmd.unlock.findOne();