Alpine Linux： apk add mongodb-tools

数据交换：

导入 mongoimport --db users --collection contacts --file contacts.json

db.stats()

db.user.help();user为表名

db.jobdetails.count()

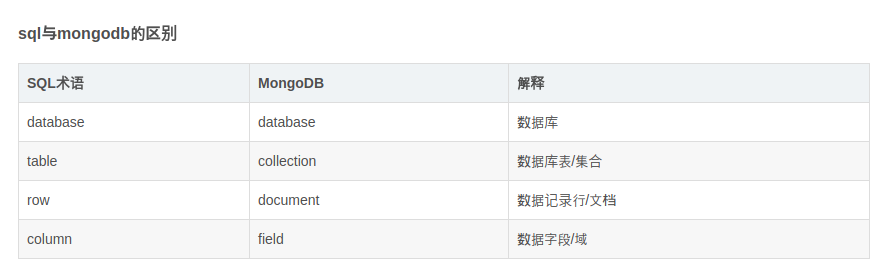
item = collection.find()

for rows in item:

print(rows.values())

ps -aux | grep mongo 通过运行时参数查看配置文件，如果没有的话，估计是默认配置文件，使用dpkg -L

是一种nosql，分布式文件存储，面向文档的数据库,存储的格式与json相似（BSON），而且同一集合不同document之间可以有不同的field。



有长连接与短连接两种池子

分片：

**① 配置服务器。**是一个独立的mongod进程，保存集群和分片的元数据，即各分片包含了哪些数据的信息。最先开始建立，启用日志功能。像启动普通的mongod一样启动配置服务器，指定**configsvr**选项。不需要太多的空间和资源，配置服务器的1KB空间相当于真是数据的200MB。保存的只是数据的分布表。当服务不可用，则变成只读，无法分块、迁移数据。

**② 路由服务器**。即mongos，起到一个路由的功能，供程序连接。本身不保存数据，在启动时从配置服务器加载集群信息，开启mongos进程需要知道配置服务器的地址，指定**configdb**选项。

**③ 分片数据服务器**。是一个独立普通的mongod进程，保存数据信息。可以是一个副本集也可以是单独的一台服务器。

mongostat 输出秒为单位的一些内容的次数、当前连接数、物理与虚拟内存使用量

mongo随着查询量的上升最后的结束会有一个拉长的小尾巴，这时候就会很影响总体的结束时间。测试的时候就测出之后需要控制的对每一个集群的查询量。内存不够的时候，会导致这段时间内查询到的量很少，波形上因此高峰滞后。

查询3天内与3天外的字段，是没有索引的。查询3天内或者是7天内，都是Collection Scan Stage，速度都是3分钟，但是如果查询3天外的，因为数据量的问题（导致cache替换、数据读取等耗时），导致查询时间需要10+分钟。

大量并发全表扫描请求的过来导致mongo的cache利用率到了瓶颈，导致用户线程也阻塞在淘汰cache上面，所以getMore整体被拖慢了。解决方法：1. 加机器，分散压力 2. 离线以较低的并发度跑出每天的数据 3. 全表扫描的模式改变，不要以三天内，三天外区分，业务以空条件扫描表，自己在业务代码里面做过滤啥的 4 迁移到mysql这种支持sql\_no\_cache的存储。

查找大量数据的时候，使用索引查找的速度没有全表扫描快

默认show users为空，创建用户（可能要先use lagou）

db.createUser({

user: "root",

pwd: "123456",

roles: [

{ role: "dbOwner", db: "lagou" },

{ role: "userAdmin", db: "lagou" }

]

}

)

取消了userAdminAnyDatabase、root 两种角色

基本操作语法：<https://blog.csdn.net/cartoonjh/article/details/79559958>

db.collection.count() use之后，被use的库就会默认代替db的位置

/\*\*  
 \* Runs a query using the following steps:  
 \* --Parsing.  
 \* --Acquire locks.  
 \* --Plan query, obtaining an executor that can run it.  
 \* --Generate the first batch.  
 \* --Save state for getMore, transferring ownership of the executor to a ClientCursor.  
 \* --Generate response to send to the client.  
 \*/

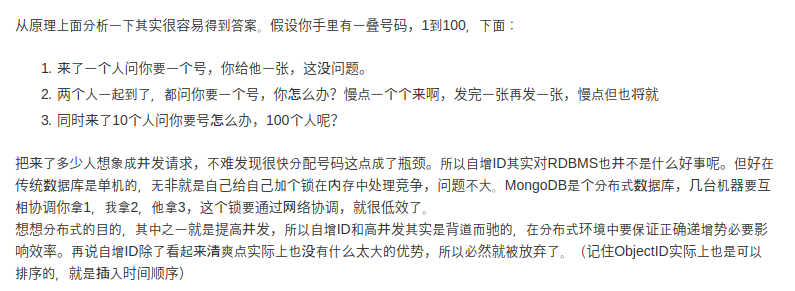
B-tree cursor 对应sort stage

全表搜索对应collection scan stage

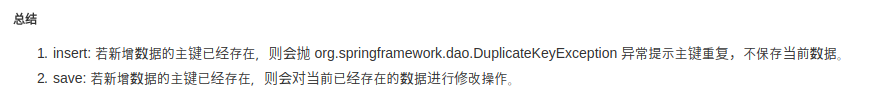
使用insert，MongoDb客户端驱动会自动为你生成一个默认ObjectId作为\_id，ObjectId是一个由8以下四部分组成的12字节对象：

4字节：UNIX时间戳（所以默认就支持按照日期来查询https://stackoverflow.com/questions/8749971/can-i-query-mongodb-objectid-by-date）  
 3字节：表示运行MongoDB的机器  
 2字节：表示生成此\_id的进程  
 3字节：由一个随机数开始的计数器生成的值

为什么不像关系型数据库一样使用自增id：（因为高并发）



insert与save的区别：



而且因为save的模式，pymong中，save是遍历数据库，导致低性能，insert直接插入数据库，其效率甚至比关系型数据库还要高。

find命令，会返回第一批满足条件的batch（默认101条记录）以及一个cursor

getMore 根据find返回的cursor继续遍历，每次遍历默认返回不超过4MB的数据

上面两条命令会导致不同的索引检索有着较大差异：<http://www.mongoing.com/archives/4913>

hint指定查找方式（使用哪一个索引，或者是不适用索引）

虽然MongoDB查询优化器一般工作的很不错，但是也可以使用hint()来强迫MongoDB使用一个特定的索引。在这种方法下某些情形下会提升性能。一个有索引的collection并且执行一个多字段的查询。传入一个制定的索引，强迫查询使用该索引。

db.users.find({"username":"user1000", "age":30}).hint({"username":1})

注意：请确定你已经创建了相应的索引。

假设在users上有个{"a": 1, "b": 1}的索引，名称是"a\_1\_b\_1"，则如下两种方式等价：

db.users.find({"a": 4, "b": 5, "c": 6}).hint({"a": 1, "b": 1})

db.users.find({"a": 4, "b": 5, "c": 6}).hint("a\_1\_b\_1")

也可以强迫查询不适用索引，做表扫描：

db.users.find().hint({"$natural":1})

mgo中对应使用iter = session.DB(db).C(collection).Find(query).Hint("$1:$natural").Iter()  ，这段代码的使用方式是我按照源码之后编写。