C/C++Linux服务器开发

高级架构师课程

三年课程沉淀 | 五次精益升级 | 十年行业积累 | 百个实战项目 | 十万内容受众

助教：秋香/2207032995

讲师介绍--专业来自专注和实力

King老师

系统架构师，曾供职著名创业公司系统架构师，微软亚洲研究院、创维集团全球研发中心。国内第一代商业Paas平台开发者。著有多个软件专利，参与多个开源软件维护。在全球化，高可用的物联网云平台架构与智能硬件设计方面有丰富的研发与实战经验。

Milo老师

曾就职于华硕集团，主导开发过全自动化测试项目，智慧养老整体解决方案，并参与多个互联网项目，并担任解决方案架构师、技术总监等职位。在自动化，互联网，移动互联网有着丰富的研发与设计经验。

Darren老师

曾供职于国内知名半导体公司（珠海扬智/深圳联发科），曾在某互联网公司担任音视频通话项目经理。主要从事音视频驱动、多媒体中间件、流媒体服务器的开发，开发过即时通讯+音视频通话的大型项目，在音视频、C/C++/GO Linux服务器领域有丰富的实战经验。

目录

CONTENTS

Mongodb体系结构

NoSQL介绍

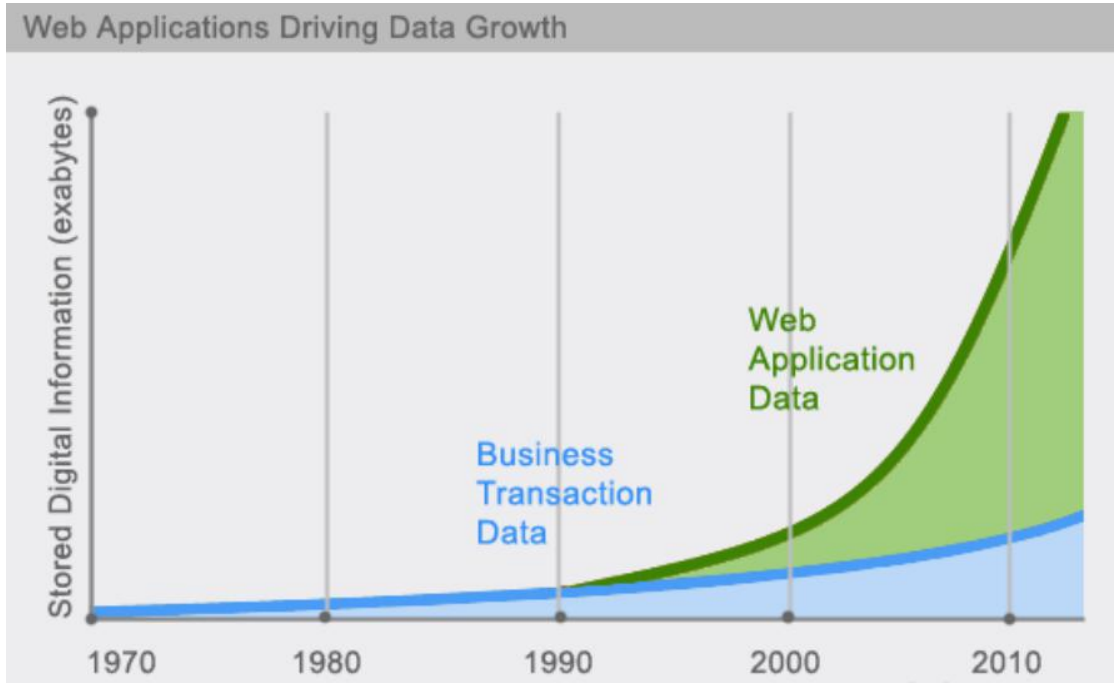
NoSQL是Not Only SQL的缩写。它指的是非关系型的数据库，是以key-value形式存储，NoSQL和传统的关系型数据库不一样，不一定遵循传统数据库的一些基本要求，比如说遵循SQL标准，ACID属性，表结构等等。

这类数据库主要有以下特点：

非关系型的，分布式的，易扩展，大数据量，灵活的数据模型。

为什么要用NoSQL

随着互联网的不断发展，各种类型的应用层出不穷，所以导致在这个云计算的时代，对技术提出了更多的需求。虽然关系型数据库已经在业界的数据存储方面占据不可动摇的地位，但是由于其天生的几个限制，使其很难满足上面这几个需求：扩展困难、读写慢、成本高、有限的支撑容量。但是 NOSQL关注的是对数据高并发地读写和对海量数据的存储等，与关系型数据库相比，它们在架构和数据模型方面做了^减法，而在扩展和并发等方面做了”加法”。



NoSQL数据库的优缺点

在优势方面，主要体现在下面这几点：

简单的扩展

快速的读写

低廉的成本

灵活的数据模型

在不足的方面，常见主要有下面这几点：

不提供对SQL的支持

支持的特性不够丰富

传统的商业智能应用

逻辑结构关系对比

关系型数据库：

MySQL数据库(database)，表(table)，记录(rows)三个层次概念组成。

非关系型数据库:

MongoDB数据库(database)，集合(collection)，文档对象(document)三个层次

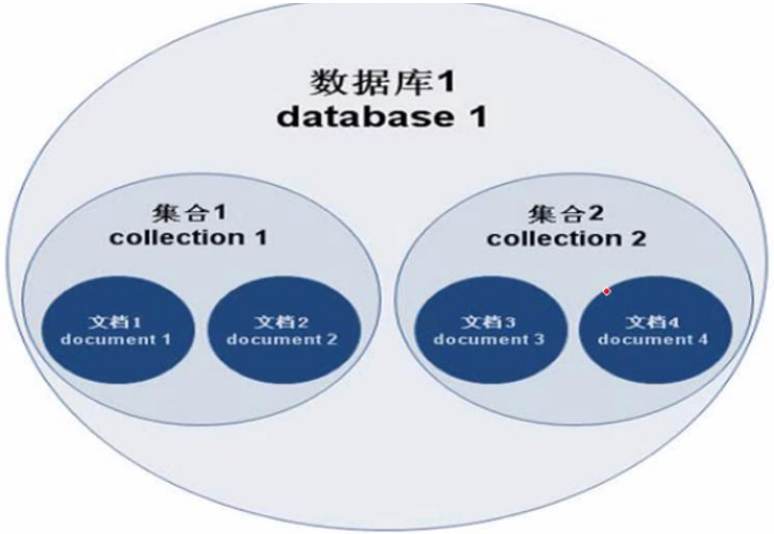
概念组成。

MongoDB里面的集合对应于关系型数据库里的表，但是集合中没有列，行和关系的

概念，

集合中只有文档，一个文档相当于一条记录。

关系数据库	MongoDB	解释对比
Database	Database	数据库/数据库
Table	Collection	数据库表/集合
Row	Document	数据库记录行/文档
Column	Field	数据列/数据字段
Index	Index	索引/索引
Table join		表关联/MongoDB不支持
Primary key	Object ID	主键/MongoDB自动将_id设为主键



数据存储结构

MySQL的数据存储结构:

MySQL的每个数据库存放在一个与数据库同名的文件夹中。

MyISAM存储引擎,数据库文件类型就包括.frm、.MYD、.MYI。

InnoDB存储引擎,数据库文件类型就包括.frm、.ibd。

/var/lib/mysql

MongoDB的数据存储结构:

MongoDB的默认数据目录是 /data/db,它负责存储所有的 MongoDB的数据文件。

/data/db目录（MMAPV1）

```
total 1.2G
-rw-r--r-- 1 root root    5 Apr 30 02:27 mongod.lock
-rw-rw-r-- 1 root root   69 Apr 30 02:33 storage.bson
-rw------- 1 root root  16M Apr 30 02:28 local.0
drwxrwxr-x 1 root root 4.0K Apr 30 02:20 journal
-rw------- 2 root root  16M Apr 30 02:33 admin.ns
-rw------- 1 root root  16M Apr 30 02:27 admin.0
-rw------- 1 root root 512M Apr 30 02:33 local.2
drwxrwxr-x 1 root root 4.0K Apr 30 02:28 diagnostic.data
drwxrwxr-x 1 root root 4.0K Apr 30 02:32 _tmp
-rw------- 1 root root  16M Apr 30 02:29 test.ns
-rw------- 2 root root  16M Apr 30 02:27 test.0
-rw------- 1 root root  32M Apr 30 02:33 test.1
-rw------- 1 root root  16M Apr 30 02:27 local.ns
-rw------- 1 root root 512M Apr 30 02:29 local.1
```

/data/db目录（WiredTiger）

```
root@ubuntu:/data/db# ll /data/db
-rw------- 1 root root 20480 Apr 30 02:27 collection-0-3422145413181698082.wt
-rw------- 1 root root 20480 Apr 30 02:33 collection-0--62885059194873530.wt
-rw------- 1 root root 36864 Apr 30 02:28 collection-2-3422145413181698082.wt
-rw------- 1 root root  4096 Apr 30 02:20 collection-4-3422145413181698082.wt
drwx------ 2 root root  4096 Apr 30 02:33 diagnostic.data/
-rw------- 1 root root 20480 Apr 30 02:27 index-1-3422145413181698082.wt
-rw------- 1 root root 20480 Apr 30 02:33 index-1--62885059194873530.wt
-rw------- 1 root root 36864 Apr 30 02:28 index-3-3422145413181698082.wt
-rw------- 1 root root  4096 Apr 30 02:32 index-5-3422145413181698082.wt
-rw------- 1 root root 12288 Apr 30 02:29 index-6-3422145413181698082.wt
drwx------ 2 root root  4096 Apr 30 02:27 journal/
-rw------- 1 root root 36864 Apr 30 02:33 _mdb_catalog.wt
-rw------- 1 root root     5 Apr 30 02:27 mongod.lock
-rw------- 1 root root 36864 Apr 30 02:29 sizeStorer.wt
-rw------- 1 root root   114 Apr 30 02:20 storage.bson
-rw------- 1 root root    46 Apr 30 02:20 WiredTiger
-rw------- 1 root root  4096 Apr 30 02:27 WiredTigerLAS.wt
-rw------- 1 root root    21 Apr 30 02:20 WiredTiger.lock
-rw------- 1 root root  1249 Apr 30 02:33 WiredTiger.turtle
-rw------- 1 root root 61440 Apr 30 02:33 WiredTiger.wt
```

Mongodb安装与配置

MongoDB安装

1.下载MongoDB

```
wget https://fastdl.mongodb.org/linux/mongodb-linux-x86_64-ubuntu1604-4.2.6.tgz
```

2.解压.压缩包

```
tar -zxvf mongodb-linux-x86_64-ubuntu1604-4.2.6.tgz
```

3.将解压包拷贝到指定目录

```
mv  mongodb-linux-x86_64-ubuntu1604-4.2.6 /opt/mongodb
```

创建数据库文件夹(默认的数据库文件的位置是/data/db,启动时会自动创建)

提示:mongodb没有具体的安装过程，解压文件包后，可以直接使用，非常高效和方便。

```
touch /opt/mongodb/logs
```

MongoDB初始设置

创建文件夹或文件

```
sudo mkdir -p /data/db                          创建目录
sudo mkdir -p /opt/mongodb/logs                 创建目录
sudo touch /opt/mongodb/logs/mongodb.log        创建空文件！ 注意不是目录而是文件
```

在mongodb安装位置/bin目录 sudo vim mongodb.conf

```
#数据文件存放目录
dbpath=/data/db
#日志文件存放目录
logpath=/opt/mongodb/logs/mongodb.log
#以守护程序的方式启用，即在后台运行
fork=true
#远程连接
bind_ip=0.0.0.0
```

MongoDB启动

启动服务端

运行mongod命令　　sudo /opt/mongodb/bin/mongod

后台启动mongodb　　sudo /opt/mongodb/bin/mongod --config /opt/mongodb/bin/mongodb.conf

启动客户端

运行mongo命令　　sudo /opt/mongodb/bin/mongo

启动命令常用参数选项说明

--dbpath	指定数据库的目录
--port	指定数据库的端口，默认是27017
--bind_ip	绑定IP
--directoryperdb	为每个db创建一个独立子目录
--logpath	指定日志存放目录
--logappend	指定日志生成方式(追加/覆盖)
--pidfilepath	指定进程文件路径,如果不指定,将不产生进程文件
--keyFile	集群模式的关键标识
--journal	启用日志
--nssize	指定.ns文件的大小,单位MB,默认是16M,最大是2GB
--maxConns	最大的并发连接数
--notablescan	不允许进行表扫描
--noprealloc	关闭数据文件的预分配功能
--fork	以后台 Daemon形式运行服务

更多的参数选项利用 mongo --help进行查看

Unix系统指令

```
milo@ubuntu:~$ ps aux |grep mongod
root        1397  0.0  0.3  52700  3872 pts/0    S+   00:14   0:00 sudo ./mongod --dbpath=/data/db --rest
root        1398  0.3  7.8 543236 78260 pts/0    Sl+  00:15   1:03 ./mongod --dbpath=/data/db --rest
milo        1824  0.0  0.0  14224   936 pts/2    S+   05:30   0:00 grep --color=auto mongod
milo@ubuntu:~$
```

注意:不要用kill -9 PID来杀死MongoDB进程，这样可能回导致MongoDB的数据损坏，
用kill -2杀死进程。

在Linux中用Kill-2和Kill-9都能够结束进程,他们的区别为:

Kill -2:功能类似于Ctrl+C是程序在结束之前,能够保存相关数据,然后再退出。

Kill -9:直接强制结束程序。

在用nohup挂起程序时,当想要结束这个程序,最好用kill -2。

可视化工具

Navicat for MongoDB	http://www.navicat.com.cn/what-is-navicat-for-mongodb
MongoDB Compass Community	https://www.mongodb.com/download-center/compass
NoSQLBooster（mongobooster）	https://nosqlbooster.com/downloads
ClusterControl	https://severalnines.com/download-clustercontrol-database-management-system
Mongo Management Studio	http://mms.litixsoft.de/index.php?lang=de/
Nosqlclient	https://github.com/nosqlclient/nosqlclient

Mongodb操作

_id和ObjectId，正则表达式

MongoDB中存储的文档必须有一个“_id”键。这个键的值可以是任何类型的，默认是ObjectId对象。

如果文档插入式没有“_id”键，系统会自动帮你创建一个。

ObjectId类型

对象id是文档中唯一的12位的ID(类似唯一主键)包含 12bytes.

0| 1| 2| 3| 4| 5| 6| 7| 8| 9| 10| 11

时间戳 | 机器 | PID | 计数器

如: ObjectId("4eae239f63520362e051e7fd")

0-3: 表示创建 unix 时间戳,格林尼治时间 UTC 时间

4-6: 机器标识码

7-8: 进程 id 组成 PID

9-11: 随机数

正则表达式

正则表示的简单示例：

^ 为匹配输入字符串的开始位置。

[0-9]+匹配多个数字，[0-9] 匹配单个数字，+匹配一个或者多个。

abc$匹配字母 abc 并以 abc 结尾，$ 为匹配输入字符串的结束位置。

MongoDB数据类型

MongoDB的文档使用BSON(Binary JSON)来组织数据, BSON类似于JSON, JSON只是一种简单的表示数据的方式,只包含了6种数据类型(null、布尔、数字、字符串、数组及对象),不能完全满足复杂业务的需要,因此,BSON还提供日期、32位数字、64位数字等类型。

null：null类型用于表示空值或不存在的字段。如: {“one” : null}

布尔类型：布尔类型有两个值, 'true'和 'false'。如: {“one” : true}

32位整数: mongoDB的控制台使用JS引擎进行输入,而JS仅支持64位浮点数,所以32位整数将会被自动转义。

64位整数: 64位整数与32位整数一样,在 MongoDB控制台使用时,会转义成64位浮点数。

64位浮点数：MongoDB控制台数字的默认类型。如: {“one” : 2.02} {“one” : 10}

字符串: UTF-8字符串都可以表示为字符串类型的数据。如: {“one” : “Hello world”}

符号：在 MongoDB控制台中不支持这种类型,将自动转义成字符串。

日期：注意:使用的时候要加上new。如:{“one” : new Date()}

代码：文档中可以包含JS代码。如: {“one” : function(){/*... ... ...*/}}

正则表达式：文档键值可以包含正则表达式,其正则表达式采用JS语法来表示。如: {“one” : ho/i}

数组：文档中键值可以表示为数组,在数组内还可以嵌套数组。如: {“x” [“a”, ”b” [“c”, ”d”]]}

内嵌文档：文档可以包含别的文档,也可以作为值嵌入到父文档中。如: {“x” : {“name” : “Tom”, “age” :20}}

创建，更新，删除，查询文档

增加、修改，删除和查询操作

1.mongodb增加操作，2.mongodb修改操作，3.mongodb删除操作，4.mongodb查询操作

高级查询操作

1.条件表达式，2.数组查询，3.子json查询，4.正则表达式，5.分页查询，6.游标使用，7.排序统计查询

高级更改操作

1.insert和save更改，2.字段更改，3.数组更改，4.字段名更改，5.位操作，6.特殊操作符$的使用

MongoDB语句

```
use zerovoice                          #创建数据库
db.createCollection("teacher")         #创建集合
#添加数据
db.teacher.insert({description:'Linux高级架构师', course:'Linux', name:'king'})
db.teacher.insertOne({description:'MongoDB', course:'Linux', name:'milo'})
db.teacher.insertMany({})
#查询
db.teacher.find()
#修改
db.teacher.update({'name':'king'},{$set:{'description':'Linux后台架构师'}})
#删除
db.teacher.remove({name:"king"})
db.teacher.deleteOne({'name':'milo'})
db.teacher.deleteMany({})
```

条件查找

MongoDB AND 条件

```
>db.col.find({key1:value1, key2:value2}).pretty()
>db.col.find({"by":"零声学院", "title":"MongoDB教程"}).pretty()
```

MongoDB OR 条件

```
>db.col.find(
   {
      $or: [{key1: value1}, {key2:value2}]
   }
).pretty()
>db.col.find({$or:[{"by":"零声学院"},{"title": "MongoDB教程"}]}).pretty()
```

AND 和 OR 联合使用

```
>db.col.find({"likes": {$gt:50}, $or: [{"by": "零声学院"},{"title": "MongoDB教程"}]}).pretty()
```

$type, limit, skip, sort, aggregate

```
db.teacher.find({"name" : {$type : 2}}) 或 db.teacher.find({"name" : {$type : 'string'}})
db.teacher.find({}).limit(1)
db.teacher.find({}).skip(1)
db.teacher.find().sort({"sex":-1})          #其中 1 为升序排列，而 -1 是用于降序排列
db.teacher.aggregate([{$group : {_id : "$name", num_tutorial : {$sum : 1}}}])
{ "_id" : "king", "num_tutorial" : 1 }
{ "_id" : "milo", "num_tutorial" : 1 }
```

Mongodb索引

索引

唯一索引（Unique）：唯一索引用于确保索引字段不存储重复的值，即强制索引字段的唯一性。

稀疏索引（Sparse）：只包含有索引字段的文档的条目，即使索引字段包含一个空值。也就是说可以跳过那些索引键不存在的文档。

TTL索引 ：

每个文件设置一个超时时间，文档到达预设的老化程序之后就会被删除。

全文本索引：

全文检索对每一个词建立一个索引，指明该词在文章中出现的次数和位置，当用户查询时，检索程序就根据事先建立的索引进行查找，并将查找的结果反馈给用户的检索方式。

地理空间索引：

2dshpere索引(用于地球表面类型的地图)

2d索引(用于平面地图和时间连续的数据)

使用GridFS存储文件：

用来存储大型二进制文件

索引操作

1、创建索引

```
db.col.createIndex({"title":1})
```

2、查看集合索引

```
db.col.getIndexes()
```

3、查看集合索引大小

```
db.col.totalIndexSize()
```

4、删除集合所有索引

```
db.col.dropIndexes()
```

5、删除集合指定索引

```
db.col.dropIndex("索引名称")
```

explain()和hint()

explain:　　提供了查询信息，使用索引及查询统计等。有利于我们对索引的优化。

queryPlanner.namespace 一个字符串，指定运行查询的命名空间（即.）。

queryPlanner.indexFilterSet boolan值，表示MongoDB 对于此query shape 是否使用了索引过滤器。

queryPlanner.winningPlan 文档类型，详细显示查询优化程序选择的查询计划。

winningPlan.stage 阶段名称。每个阶段都有每个阶段特有的信息。 例如，IXSCAN 阶段将包括索引边界以及特定于索引扫描的其他数据。 如果阶段具有子阶段或多个子阶段，则阶段将具有inputStage 或 inputStages。

winningPlan.inputStage 描述子阶段的文档。它为其父级提供文档或索引键。 如果父级只有一个子级，则该字段存在。

winningPlan.inputStages 描述子阶段的数组。子阶段为父阶段提供文档或索引键。 如果父级具有多个子节点，则该字段存在。 例如，$or 表达式或索引交集的阶段消耗来自多个源的输入。

queryPlanner.rejectedPlans 查询优化器考虑和拒绝的候选计划数组。 如果没有其他候选计划，则该数组可以为空。

```
db.teacher.find().explain()
```

hint:　　强制 MongoDB 使用一个指定的索引

```
db.users.find({gender:"M"},{user_name:1,_id:0}).hint({gender:1,user_name:1})
```

Mongodb存储引擎

存储引擎

1. MMAPV1
2. WiredTiger

关键特性	MMAPV1引擎	wiredTiger引擎
默认引擎引入	3.0版本及其之前作为MongoDB默认引擎，4.0 及其以后版本废弃	3.0版本引入，3.2版本起作为默认引擎
数据压缩	不支持	默认采用 snappy 压缩模式同时支持zlib 压缩模式
Journal日志	写操作首先在内存进行修改并将对应操作写入磁盘journal文件。如果修改在数据同步至磁盘之前MongoDB崩溃,重启之后MongoDB可以利用journal日志重新应用对应的写操作到数据文件从而保证数据一致性	journal日志保存两个检查点之间的所有修改，如果MongoDB在下一个检查点之前崩溃，引擎将会使用journal重放最后一个检查点以来的所有数据修改操作
锁和并发控制	Till 2.6, MongoDB uses a readers-writer [1] lock that allows concurrent reads access to a database but gives exclusive access to a single write operation. From 3.0, uses collection level lock	It supports document level locking.
事务	单个文档操作具备原子性	MongoDB4.0版本支持多文档事务
CPU与性能	增加CPU核数对性能的提升有限	多核系统中提升CPU核数可很大程度提升性能
数据加密	不支持	官方企业版和percona的3.6.8（PSMDB） BETA 版本支持
内存使用	自动使用机器所有空闲内存作为自己的缓存	同时使用引擎内部缓存及文件系统缓存
修改	善于处理具有大量插入、读取和就地更新的工作场景	不支持原地更新故会导致整个文档的重写
调优	可调优的点极少	允许通过不同的参数对引擎进行调优，比如：内部缓存大小(cache size),读写并发控制(read / write tickets),检查点触发时间间隔(checkpoint interval)等

Mongodb应用程序设计

应用程序设计

范式与反范式化

优化数据操作

数据库和集合的设计

一致性管理

适用场景

适用场景

持久化缓存层

高效的实时性

用于对象及JSON数据的存储

高伸缩性的场景

大尺寸，低价值的数据存储

不适用场景

要求高度事务性的系统

复杂多表查询 （关系型数据库）

真实场景

MongoDB的应用已经渗透到各个领域，比如游戏、物流、电商、社交、物联网、视频直播等。

游戏场景：使用MongoDB存储游戏用户信息，用户的装备、积分等直接以内嵌文档的形式存储，方便查询、更新。

物流场景：使用MongoDB存储订单信息，订单状态在运送过程中会不断更新，以MongoDB内嵌数组的形式来存储，一次查询就能将订单所有的变更读取出来。

社交场景：使用MongoDB存储存储用户信息，以及用户发表的朋友圈信息，通过地理位置索引实现附近的人、地点等功能

物联网场景：使用MongoDB存储所有接入的智能设备信息，以及设备汇报的日志信息，并对这些信息进行多维度的分析

视频直播：使用MongoDB存储用户信息、礼物信息等。

Mongodb工程案例开发

JSON, BSON

JSON是一种轻量级的数据交换格式。完全独立于编程语言的文本格式来存储和表示数据。支持字符串、数字、对象、数组等类型。

BSON是一种计算机数据交换格式，主要被用作MongoDB数据库中的数据存储和网络传输格式。它是一种二进制表示形式，能用来表示简单数据结构、关联数组（MongoDB中称为“对象”或“文档”）以及MongoDB中的各种数据类型。BSON之名缘于JSON，含义为Binary JSON（二进制JSON）。

开发实战

mongo-c-driver驱动安装

官方驱动网址：https://docs.mongodb.com/drivers/c/

驱动github位置：https://github.com/mongodb/mongo-c-driver

```
mkdir cmake-build
cd cmake-build
Make
Make install
```

引入头文件： #include <mongoc.h>

编译:

```
gcc -o mongo_test mongo_test.c -L/usr/local/lib -I/usr/local/include/libmongoc-1.0 -
I/usr/local/include/libbson-1.0 -lmongoc-1.0 -lbson-1.0
```

高德获取Key

1. 注册地址：https://lbs.amap.com/dev/id/choose，选择个人开发者即可

2. 获取Key：https://lbs.amap.com/api/webservice/guide/create-project/get-key
3. 服务平台：Web服务

根据经纬度保存高德地图目标位置信息

高德api：https://lbs.amap.com/api/webservice/summary/

经纬度：112.897992,28.217434　　目标位置信息：湖南省长沙市岳麓区麓谷街道雅阁国际

示例：

http://restapi.amap.com/v3/geocode/regeo?location=112.897992,28.217434&key=4d607f1ae9841d414e27f8caa0c2e55a&radius=1000&extensions=base

2020

遇见零声，遇见更好的自己

助教：秋香/2207032995