C++操作mongodb

其实主要就是参照官方文档 http://mongoc.org/libmongoc/current/installing.html
https://docs.mongodb.com/drivers/?jump=docs
http://mongocxx.org/mongocxx-v3/installation/
https://docs.mongodb.com/drivers/c/

参考网页版本课件：
https://www.yuque.com/docs/share/2c0e2ab9-30f4-4897-b2ad-a6c854c490f4?#《C++操作mongodb》

编译平台：

- Ubuntu 16.04
- gcc 11.2
- g++ 11.2 需要>= 8.4

升级gcc g++编译器参考：https://www.yuque.com/docs/share/660d6007-2390-4aa5-8dff-23b4f882d7af?# 《QUIC开源库安装和实践》

0 背景知识

BSON(Binary Serialized Document Format) 是一种二进制形式的存储格式，采用了类似于 C 语言结构体的名称、对表示方法，支持内嵌的文档对象和数组对象，具有轻量性、可遍历性、高效性的特点，可以有效描述非结构化数据和结构化数据。

BSON是一种类json的一种二进制形式的存储格式，简称Binary JSON，它和JSON一样，支持内嵌的文档对象和数组对象，但是BSON有JSON没有的一些数据类型，如Date和BinData类型。
BSON可以做为网络数据交换的一种存储形式，这个有点类似于Google的Protocol Buffer，但是BSON是一种schema-less的存储形式，它的优点是灵活性高，但它的缺点是空间利用率不是很理想，
BSON有三个特点：轻量性、可遍历性、高效性。

1 安装驱动mongo-c-driver和mongocxx-driver

mongocxx依赖与mongo-c-driver，所以我们会先安装mongo-c-driver然后再安装mongo-cxx。

1.1 安装mongo-c-driver

下载地址：https://github.com/mongodb/mongo-c-driver/tags

我们选择较新的稳定版本1.17.5

Shell 复制代码

```shell
#下载1.17.5版本的压缩包
wget https://github.com/mongodb/mongo-c-driver/releases/download/1.17.5/mongo-c-driver-1.17.5.tar.gz
#解压压缩包
tar zxvf mongo-c-driver-1.17.5.tar.gz
#进入目录
cd mongo-c-driver-1.17.5
cd build
cmake -DENABLE_AUTOMATIC_INIT_AND_CLEANUP=OFF ..
make
sudo make install
```

lib: /usr/local/lib/
include: /usr/local/include/libmongoc-1.0

如果需要卸载：

Shell 复制代码

```shell
sudo /usr/local/share/mongo-c-driver/uninstall.sh
```

另外以上安装步骤其实在官方文档指引中都有，如下：
http://mongoc.org/libmongoc/current/tutorial.html

更多的mongo c编程参考：http://mongoc.org/libmongoc/current/tutorial.html

1.2 安装mongo-cxx

下载地址：https://github.com/mongodb/mongo-cxx-driver/tags

Shell 复制代码

```shell
#使用3.6.6的版本
wget https://github.com/mongodb/mongo-cxx-driver/releases/download/r3.6.6/mongo-cxx-driver-r3.6.6.tar.gz

#解压压缩包
tar zxvf mongo-cxx-driver-r3.6.6.tar.gz
#进入目录
cd mongo-cxx-driver-r3.6.6
cd build

cmake -DCMAKE_BUILD_TYPE=Release -DCMAKE_CXX_STANDARD=17 -DBSONCXX_POLY_USE_STD=ON -DCMAKE_INSTALL_PREFIX=/usr/local ..
# sudo make EP_mnmlstc_core  这步骤不能漏
sudo make EP_mnmlstc_core
make
sudo make install
sudo ldconfig

```

安装路径：
lib：/usr/local/lib/
include：/usr/local/include/mongocxx

2 测试mongodb

实例1,新建一个库表并插入一条数据

注：这个实例就是官方文档上的。其作用还有各种编译方法也都在指导文档中有。 更多操作参见 官方文档(是要重点研究的)

代码

C++ 复制代码

```cpp
//hello_mongoc.c

#include <mongoc/mongoc.h>
int main (int argc, char *argv[])
{
   const char *uri_string = "mongodb://localhost:27017";
   mongoc_uri_t *uri;    // url
   mongoc_client_t *client;  // 客户端
   mongoc_database_t *database;   // 数据库
   mongoc_collection_t *collection;  // 集合
   bson_t *command, reply, *insert;
   bson_error_t error;
   char *str;
   bool retval;

   /*
    * Required to initialize libmongoc's internals
    */
   mongoc_init ();

   /*
    * Optionally get MongoDB URI from command line
    */
   if (argc > 1) {
      uri_string = argv[1];
   }

   /*
    * Safely create a MongoDB URI object from the given string
    */
   uri = mongoc_uri_new_with_error (uri_string, &error);
   if (!uri) {
      fprintf (stderr,
               "failed to parse URI: %s\n"
               "error message:       %s\n",
               uri_string,
               error.message);
      return EXIT_FAILURE;
   }

   /*
    * Create a new client instance
    */
   client = mongoc_client_new_from_uri (uri);
   if (!client) {
```

```c
      return EXIT_FAILURE;
   }

   /*
    * Register the application name so we can track it in the profile logs
    * on the server. This can also be done from the URI (see other examples).
    */
   mongoc_client_set_appname (client, "connect-example");

   /*
    * Get a handle on the database "db_name" and collection "coll_name"
    */
   database = mongoc_client_get_database (client, "db_name");
   collection = mongoc_client_get_collection (client, "db_name", "coll_name");

   /*
    * Do work. This example pings the database, prints the result as JSON and
    * performs an insert
    */
   command = BCON_NEW ("ping", BCON_INT32 (1));

   retval = mongoc_client_command_simple (
      client, "admin", command, NULL, &reply, &error);

   if (!retval) {
      fprintf (stderr, "%s\n", error.message);
      return EXIT_FAILURE;
   }

   str = bson_as_json (&reply, NULL);
   printf ("%s\n", str);

   insert = BCON_NEW ("hello", BCON_UTF8 ("world"));

   if (!mongoc_collection_insert_one (collection, insert, NULL, NULL, &error)) {
      fprintf (stderr, "%s\n", error.message);
   }

   bson_destroy (insert);
   bson_destroy (&reply);
   bson_destroy (command);
   bson_free (str);

```

```
    /*
     * Release our handles and clean up libmongoc
     */
    mongoc_collection_destroy (collection);
    mongoc_database_destroy (database);
    mongoc_uri_destroy (uri);
    mongoc_client_destroy (client);
    mongoc_cleanup ();

    return EXIT_SUCCESS;
}
```

编译

例如说此处手动指定头文件及包含路径进行编译，编译语句如下。执行后即可得到可执行文件。

Shell 复制代码

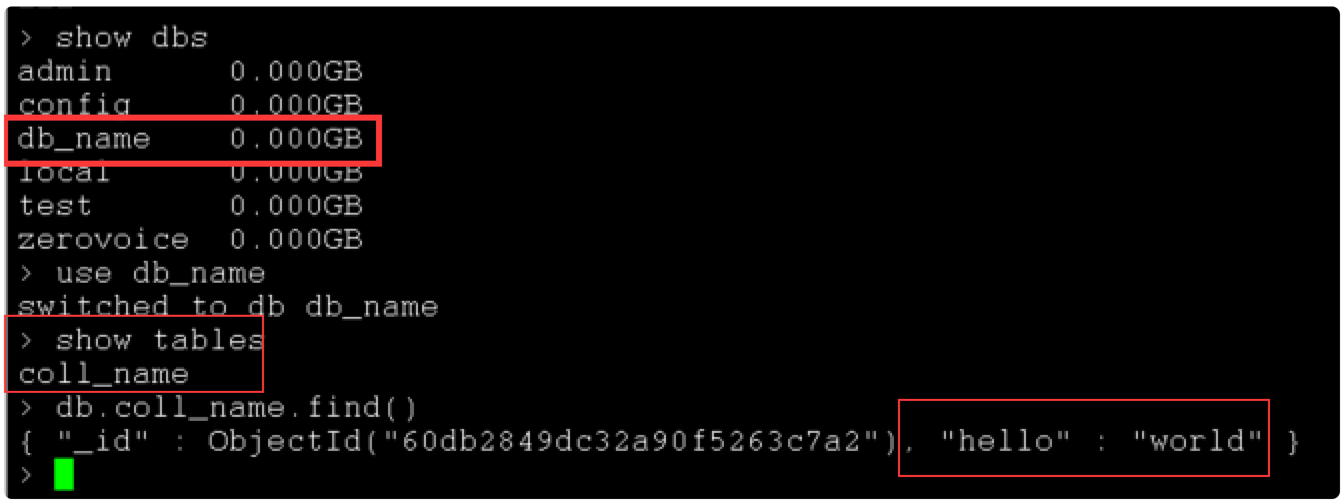
```shell
gcc -o hello_mongoc hello_mongoc.c \
    -I/usr/local/include/libbson-1.0 -I/usr/local/include/libmongoc-1.0 \
    -lmongoc-1.0 -lbson-1.0
```

执行

Shell 复制代码

```shell
./hello_mongoc
{ "ok" : 1.0 }
```

打印{ "ok" : 1.0 }，然后通过mongo查看数据库



```
> show dbs
admin        0.000GB
config       0.000GB
db_name      0.000GB
local        0.000GB
test         0.000GB
zerovoice    0.000GB
> use db_name
switched to db db_name
> show tables
coll_name
> db.coll_name.find()
{ "_id" : ObjectId("60db2849dc32a90f5263c7a2"), "hello" : "world" }
>
```

实例2,测试插入1000、10000条数据所需时间

代码

Shell 复制代码

```
// performance_mongo.cpp, 实际是根据example的create.cpp修改
#include <chrono>

#include <bsoncxx/builder/basic/array.hpp>
#include <bsoncxx/builder/basic/document.hpp>
#include <bsoncxx/builder/basic/kvp.hpp>
#include <bsoncxx/types.hpp>

#include <mongocxx/client.hpp>
#include <mongocxx/instance.hpp>
#include <mongocxx/uri.hpp>
#include <time.h>
#include <string>
#include <iostream>
using bsoncxx::builder::basic::kvp;
using bsoncxx::builder::basic::make_array;
using bsoncxx::builder::basic::make_document;

int main(int, char **)
{

    mongocxx::instance inst{};
    mongocxx::client conn{mongocxx::uri{}};

    auto db = conn["test"];

    // We choose to move in our document here, which transfers ownership to insert_one()
    clock_t startTime = clock();
    for (int i = 0; i <= 1000; ++i)
    {
        // 封装一个文档
        bsoncxx::document::value restaurant_doc = make_document(
            kvp("address",
                make_document(kvp("street", "2 Avenue"),
                              kvp("zipcode", 10075),
                              kvp("building", "1480"),
                              kvp("coord", make_array(-73.9557413, 40.7720266)))),
            kvp("borough", "Manhattan"),
            kvp("cuisine", "Italian"),
            kvp("grades",
                make_array(
                    make_document(kvp("date", bsoncxx::types::b_date{std::chrono::milliseconds{12323}}),
                                  kvp("grade", "A"),
                                  kvp("score", 11)),
                    make_document(
                        kvp("date", bsoncxx::types::b_date{std::chrono::milliseconds{121212}}),
                        kvp("grade", "B"),
                        kvp("score", 17)))),
            kvp("name", "Vella"),
            kvp("restaurant_id", std::to_string(i)));
        // 插入数据库
        auto res = db["restaurants"].insert_one(std::move(restaurant_doc));
    }
    clock_t endTime = clock();
    std::cout << " insert total time: " << double(endTime - startTime) / CLOCKS_PER_SEC << " s" << std::endl;
    // @end: cpp-insert-a-document
}
```

编译

Shell 复制代码

```shell
g++ --std=c++11 performance_mongo.cc -o performance_mongo -I/usr/local/include/mongocxx/v_noabi -I/usr/local/include/bsoncxx/v_noabi -L/usr/local/lib -lmongocxx -lbsoncxx
```

或者

Shell 复制代码

```shell
g++ --std=c++11 performance_mongo.cpp -o performance_mongo $(pkg-config --cflags --libs libmongocxx)
```

pkg-config --cflags --libs libmongocxx 能找出来对应的include路径，以及lib路径。

执行

```shell
lqf@ubuntu:/mnt/hgfs/mongo/src$ ./performance_mongo
 insert total time: 0.060378 s
```

去mongo shell查看test，可以看到插入的数据。

实例3,C连接复制集集群

参考实例1，只是url做了修改

由`const char *uri_string = "mongodb://localhost:27017";`

变成

```c
const char *uri_string = "mongodb://localhost:28017,localhost:28018,localhost:28019/?replicaSet=rs0"; //
```

主要是提供了复制集集群所有节点的ip以及对应复制集的名字。

代码

```c
//hello_mongoc_replication.c
/* gcc -o hello_mongoc_replication hello_mongoc_replication.c \
    -I/usr/local/include/libbson-1.0 -I/usr/local/include/libmongoc-1.0 \
    -lmongoc-1.0 -lbson-1.0
    */
#include <mongoc/mongoc.h>
int main(int argc, char *argv[])
{
   // https://docs.mongodb.com/manual/reference/connection-string/
   const char *uri_string = "mongodb://localhost:28017,localhost:28018,localhost:28019/?replicaSet=rs0"; // 连接默认的地址
   mongoc_uri_t *uri;                                            // url
   mongoc_client_t *client;                                      // 客户端
   mongoc_database_t *database;                                  // 数据库
   mongoc_collection_t *collection;                              // 集合
   bson_t *command, reply, *insert;
   bson_error_t error;
   char *str;
   bool retval;

   /*
    * Required to initialize libmongoc's internals 初始化内部
    */
   mongoc_init();

   /*
    * Optionally get MongoDB URI from command line
    */
   if (argc > 1)
   {
      uri_string = argv[1]; // 可以指定其他mongodb服务器地址
   }

   /*
    * Safely create a MongoDB URI object from the given string
    */
   uri = mongoc_uri_new_with_error(uri_string, &error); // 连接MongoDB服务器
   if (!uri)
   {
      fprintf(stderr,
              "failed to parse URI: %s\n"
              "error message:       %s\n",
              uri_string,
              error.message);
      return EXIT_FAILURE;
   }

   /*
    * Create a new client instance，创建客户端实例
    */
   client = mongoc_client_new_from_uri(uri);
   if (!client)
   {
      return EXIT_FAILURE;
   }

   /*
    * Register the application name so we can track it in the profile logs
    * on the server. This can also be done from the URI (see other examples).
    */
   mongoc_client_set_appname(client, "connect-example");

   /*
    * Get a handle on the database "db_name" and collection "coll_name"
    */
   database = mongoc_client_get_database(client, "db_name"); // 创建db
   collection = mongoc_client_get_collection(client, "db_name", "coll_name");

   /*
    * Do work. This example pings the database, prints the result as JSON and
    * performs an insert
    */
   command = BCON_NEW("ping", BCON_INT32(1));

   retval = mongoc_client_command_simple(
       client, "admin", command, NULL, &reply, &error);

   if (!retval)
   {
      fprintf(stderr, "%s\n", error.message);
      return EXIT_FAILURE;
   }
```

```
    str = bson_as_json(&reply, NULL);
    printf("%s\n", str);                            // { "ok" : 1.0 }
   // { "_id" : ObjectId("60db2849dc32a90f5263c7a2"), "hello" : "world" }
    insert = BCON_NEW("hello", BCON_UTF8("world")); // 组装一个json对象
                                                    // 插入到集合
    if (!mongoc_collection_insert_one(collection, insert, NULL, NULL,
&error))
    {
        fprintf(stderr, "%s\n", error.message);
    }

    bson_destroy(insert);
```

可以kill掉primary节点后再执行该程序。

核心在于url里面要包含对应节点的服务地址。

实例4,C++连接复制集集群

mongocxx::client conn{mongocxx::uri{}};   // 缺省 "mongodb://localhost:27017";
改成
ongocxx::uri url{"mongodb://localhost:28017,localhost:28018,localhost:28019/?replicaSet=rs0"};
   mongocxx::client conn{url};
即可连接。

3 更多范例

参考：mongo-cxx-driver/examples
见课程源码：mongo-src

对应的文档参考：mongo-src\mongo-driver\mongo-cxx-driver\docs\content