1.新KV存储系统在intel非易失性内存和Intel ssd性能对比测试(benchmark

./memtier\_benchmark -s 172.17.233.55 -p 8003 -t 20 -c 1 --command="get \_\_key\_\_" --command-key-pattern=R --command-ratio=1 --key-prefix=intel\_string\_00 --key-minimum=1000000000 --key-maximum=4500000000 --test-time=30

)

2.集群控制平台后端2.0的开发：pika-stark、stark-admin部分的单元测试、mysql数据库之间交互（使用sqlx）

3.集群控制平台由gin框架开发，采用swaggo从而自动化生成TESTful api接口文档

4.grafana新的监控页面和设计指标，grafana页面的手动注册、自动注册（json、返回string、http.defaultclient.do把req变成resp）

gin包rest api(representation state transfer，用url表示资源，通过http方法表示对这些资源的操作)

创建router对象(gin.Default())；通过router创建监听。设置监听回调；router.run()

gin进行了封装，把request和response都封装到 gin.Context的上下文环境

sqlx包与mysql进行数据交流

交互命令**User:Pwd@tcp(Host:Port)**

const dsn = "root:123456@tcp(127.0.0.1:3306)/database"/?charset=utf8mb4

db, err = sqlx.Connect("mysql", dsn)

func Connect(driverName, dataSourceName string)

sql := `update user set name = ?, age = ? where id = ?`创表也是这个函数

res, err := DB.Exec(sql, "LGX", 28, 20)MustExec会返回panic

查询一行数据

var u user

if err := db.Get(&u, sqlStr, 1); err != nil {

查询多行数据

var users []user

if err := db.Select(&users, sqlStr, 0); err != nil {

Mysql时间戳用的timestamp，1970-01-01 00:00:01.000000到2038。有时区的自动转换。非小数部分4字节，datetime5字节，5.6.4之前是8字节。小数部分都是0到3字节。On update current\_timestamp()



http请求报文

1.请求方法（主要就是post(非覆盖，用于new新增)和get(覆盖，用于update资源)、delete、put。Head和trace、options）

2.url地址和报文头的host属性相加为完整的url

3.协议名称/协议版本

4.报文头

Accept: 告诉server，client接受什么类型的响应 application/json

Referer：req是从哪个url来的

Cache-control: 对缓存控制

Accept-encoding：告诉server，接受什么编码格式

Host：要请求resource所在的主机和端口

Authorization：当客户端接收到来自WEB服务器的 WWW-Authenticate 响应时，用该头部来回应自己的身份验证信息给WEB服务器

1. 报文体: kv组成的，多个请求参数的数据

http响应报文

1. 报文协议/版本
2. 状态码/状态描述

303重定向到其他页面304资源未修改

1. 响应报文头

Date 设置消息发送的日期和时间

1. 响应报文体

写完import它是红色的，然后在那个有go.mod的目录上go mod tidy一下就好了。

如果想把包导入到本项目，再go mod vendor一下

文件夹及子文件夹权限可以用  chmod -R 777 /var/home/userid/cc