[基于Golang协程实现流量统计系统 2](#_Toc22106)

[概念 2](#_Toc14609)

[系统流程 3](#_Toc24808)

[并发编程 4](#_Toc7641)

[启蒙 4](#_Toc7257)

[协程 4](#_Toc18644)

[Goroutine 4](#_Toc13214)

[Golang协程特性实战 5](#_Toc10752)

[channel是具有阻塞特性 5](#_Toc19533)

[Channel是具有随机性(只要运行了go func(),执行顺序是不一定的) 5](#_Toc6139)

[带缓冲的Buffered Channel 6](#_Toc5220)

[Range 7](#_Toc23357)

[(重点)Channel的关闭必须在写的那头(即生产者)进行关闭，在读这头(即消费者)关闭容易产生死锁 7](#_Toc806)

[Select (select为非阻塞，不会影响当前的协程，包括主协程) 8](#_Toc17411)

[Select随机选择(其实也不是随机，谁先来，谁就先解决问题) 8](#_Toc3539)

[Select等待机制 (谁先来执行谁) 9](#_Toc10392)

[项目实战1 10](#_Toc22359)

[快速构建实例网站 10](#_Toc22243)

[打点原理分析 12](#_Toc23063)

[JavaScript获取客户端访问的内容 13](#_Toc7008)

[打点获取客户refer(refer来流量统计里是必须要用的,重点) 14](#_Toc17801)

[网站端的总结 16](#_Toc20720)

[Nginx打点服务器的搭建与配置 17](#_Toc21827)

[打点系统返回空白图片的意义 17](#_Toc18471)

[Nginx的ngx\_http\_empty\_gif\_module模块安装 18](#_Toc21905)

[正常配置后，再用ajax打开网址，出现405异常 18](#_Toc2558)

[nginx空gif一般用于高并发而且不需要返回的单点请求 19](#_Toc27111)

[Access\_log记录用户的访问记录 19](#_Toc22798)

[nginx简单优化技巧 20](#_Toc2976)

[做高性能的打点，一定要优化nginx 21](#_Toc11720)

[打点程序Golang的编写与处理(模拟生成测试数据) 21](#_Toc11237)

[打点程序最好添加伪静态的处理，这样统计起来比较简单 21](#_Toc3395)

[Golang--flag包(go语言通过使用标准库里的flag包来处理命令行参数)(重点，一般用于无界面程序) 21](#_Toc31162)

[Golang统计分析模块 22](#_Toc22418)

[代码编辑部分 22](#_Toc5504)

[Go get golang.org的包失败解决办法 22](#_Toc20935)

["github.com/mediocregopher/radix.v2/pool" --Redis使用库 22](#_Toc1386)

[为了防止连接池由于长时间没有工作而被断开，我们会起一个协程，定时访问redis 23](#_Toc4364)

[统计数据逐层添加，加洋葱皮的过程 23](#_Toc6838)

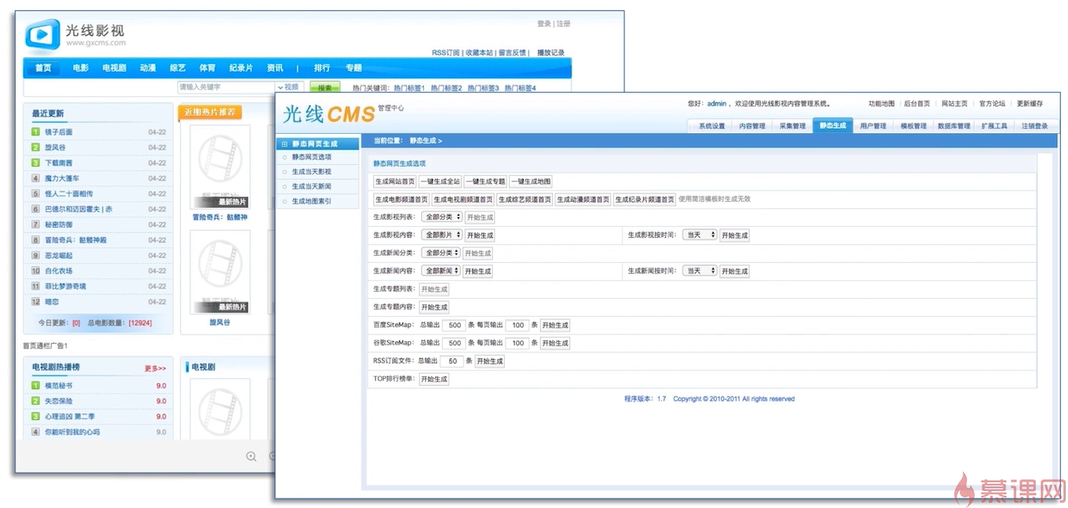
[Hase常用于大企业高并发的数据存储，Mysql不太好 23](#_Toc22389)

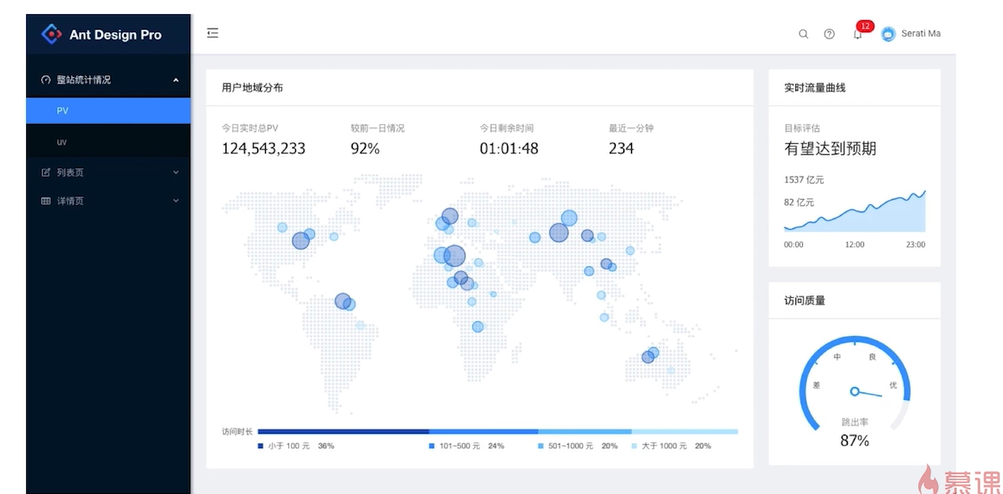
[使用蚂蚁金服的AntDesign能够快速开发中台的界面搭建 23](#_Toc13747)

[流量统计整体回顾 24](#_Toc14458)

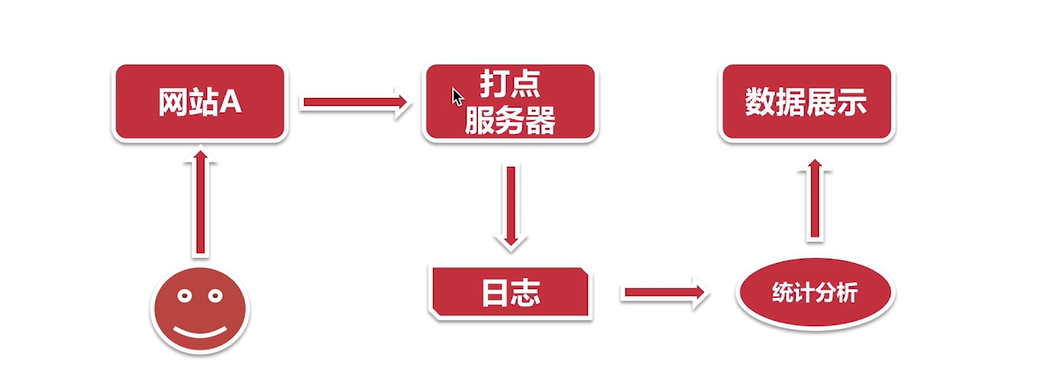
# 基于Golang协程实现流量统计系统

## 概念



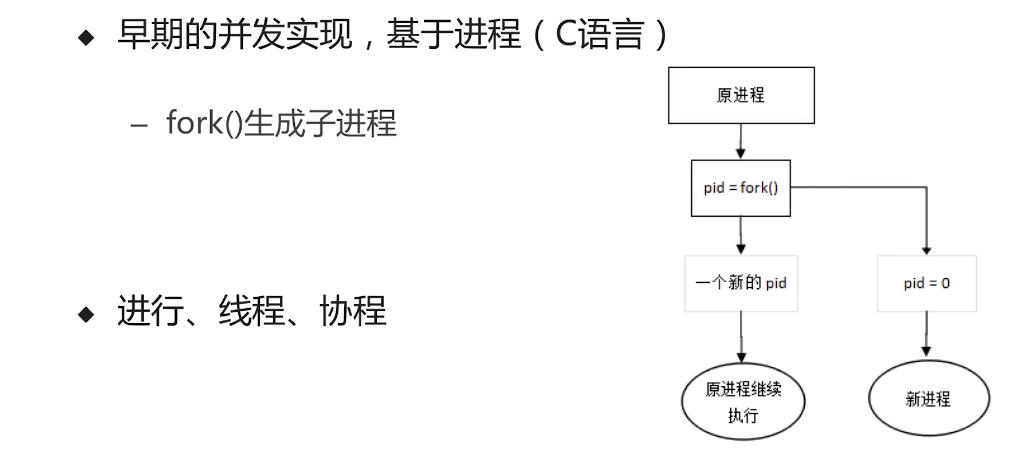


### 系统流程

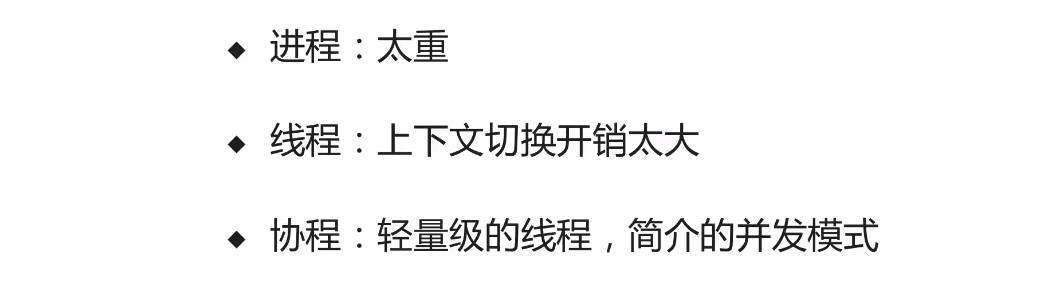


## 并发编程

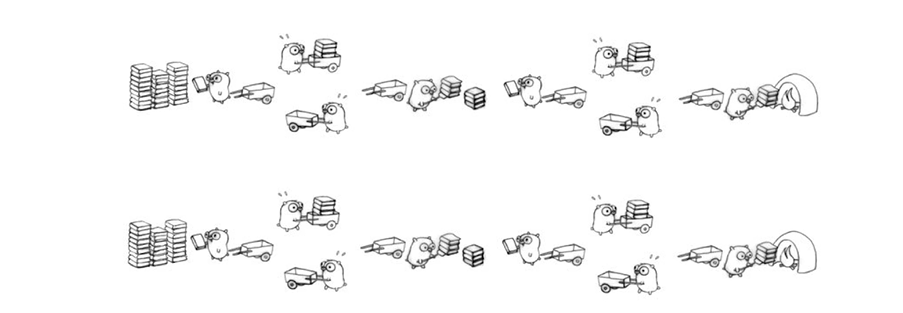
### 启蒙



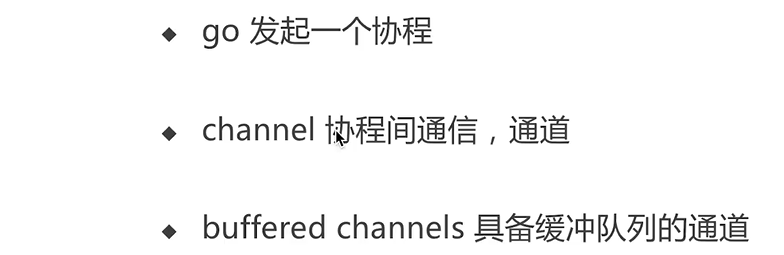
### 协程



### Goroutine



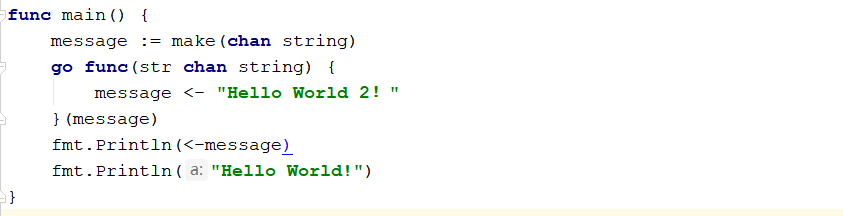
### Golang协程特性实战

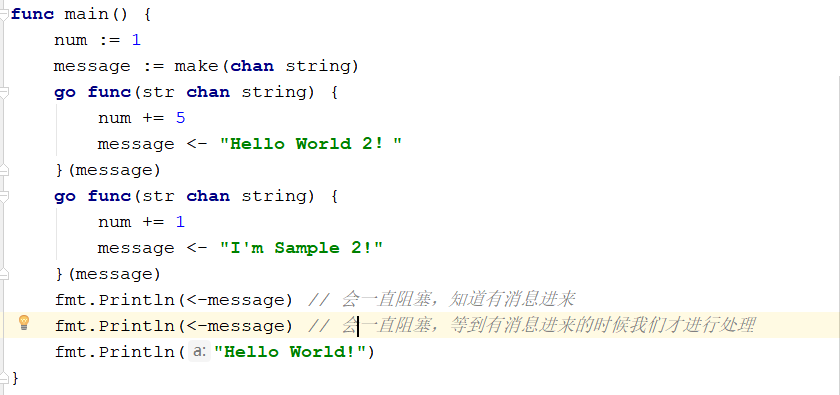


### channel是具有阻塞特性

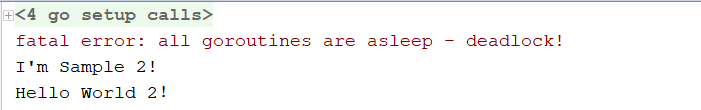
### Channel是具有随机性(只要运行了go func(),执行顺序是不一定的)

<-message会阻塞所有的后续程序，知道接收到一次channel返回的内容才会正常运行下去





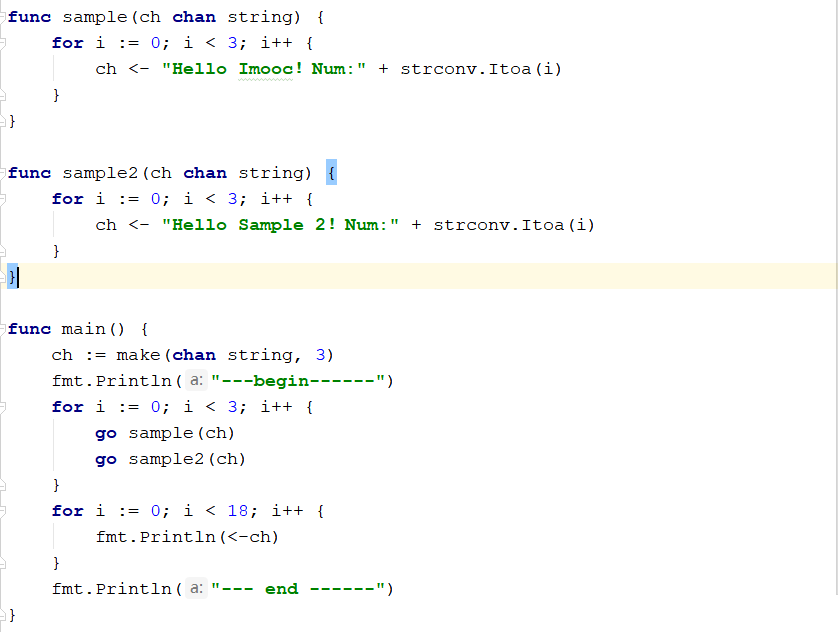
如果读取的写多了,会引发deadlock

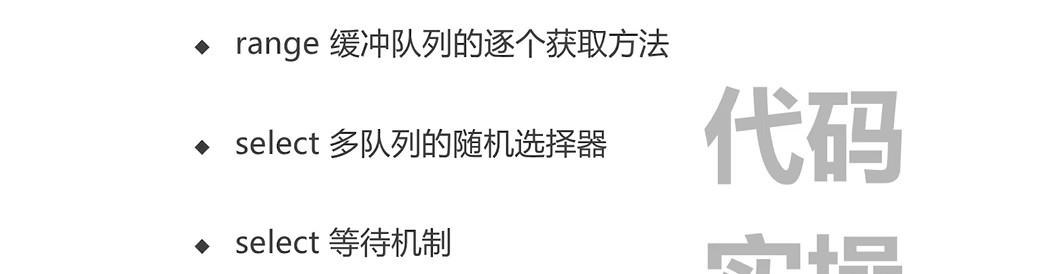


如果读取写少了，并不会出现异常反应



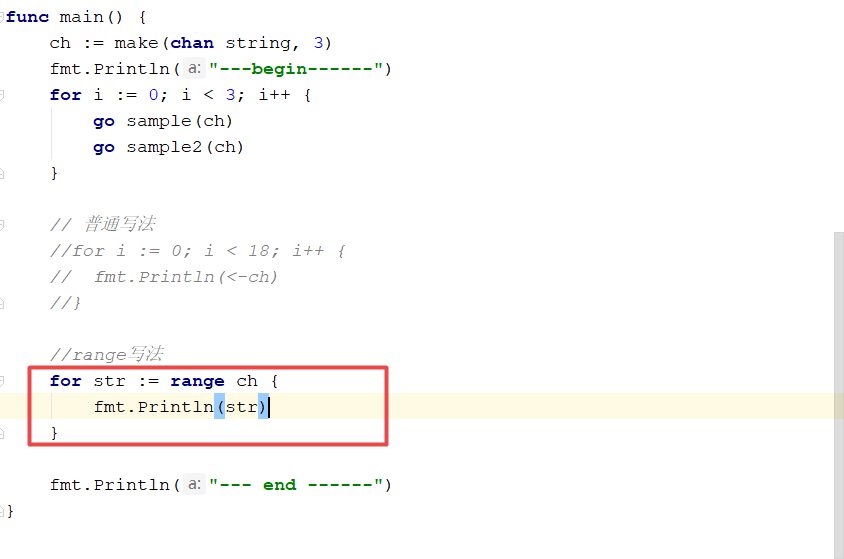
### 带缓冲的Buffered Channel



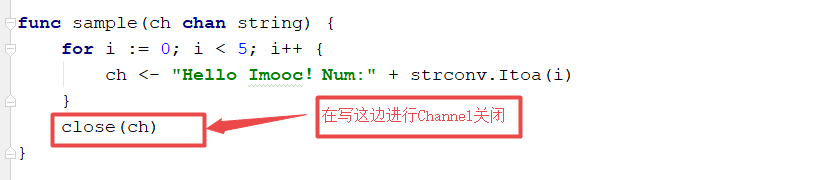


### Range

下面的这种写法会出现死锁异常，因为由于缓冲通道就产生18条的写channel信息，但是，for...range死循环会产生多过18条的读channel信息，造成死锁循环(写数量有限，但是读数量无限循环，会产生死锁)



### (重点)Channel的关闭必须在写的那头(即生产者)进行关闭，在读这头(即消费者)关闭容易产生死锁



上网搜索: 如何优雅的关闭的Channel

Golang 内建的 close 方法可以关闭 channel，如果往已经关闭的 channel 发送数据，则会报错：panic: close of closed channel.

看如下代码，在一段时间内，生产者可以不断往 channel 写入数据，消费者进行处理，一段时间后 channel 关闭了，这个时候如果还有数据往 channel 发送，程序就会报错。

如何优雅关闭 channel

那么在往通道发数据前如何判断通道是否关闭呢？

1.\_,ok := <- jobs

此时如果 channel 关闭，ok 值为 false，如果 channel 没有关闭，则会漏掉一个 jobs

2.使用 select 方式

再创建一个 channel，叫做 timeout，如果超时往这个 channel 发送 true，在生产者发送数据给 jobs 的 channel，用 select 监听 timeout，如果超时则关闭 jobs 的 channel.

### **Select** (select为非阻塞，不会影响当前的协程，包括主协程)

**select的运行每一次只会运行一次,如果像连续看到多次效果，需要添加for进行处理**

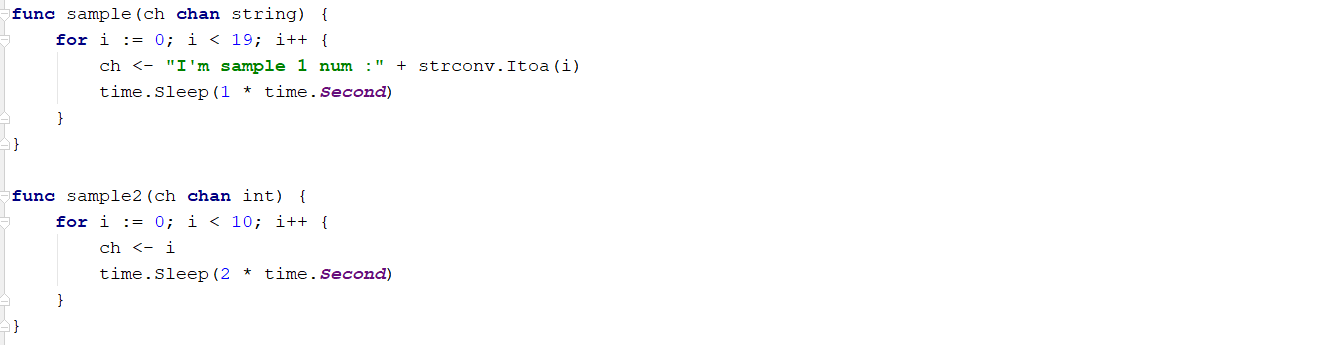
**golang 的 select 就是监听 IO 操作，当 IO 操作发生时，触发相应的动作。**

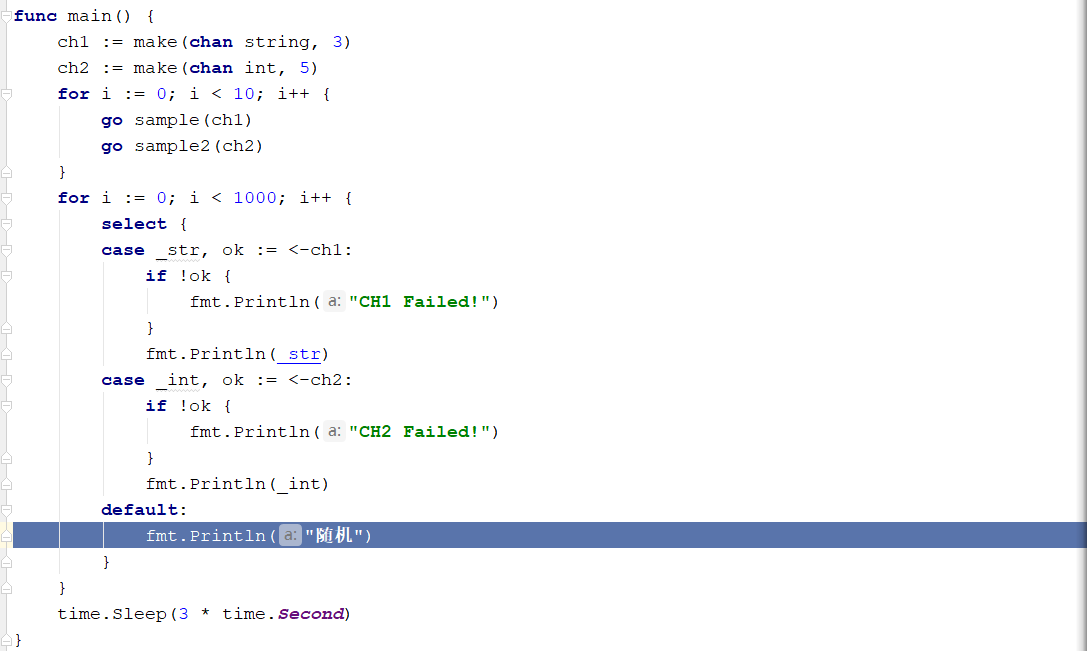
**在执行select语句的时候，运行时系统会自上而下地判断每个case中的发送或接收操作是否可以被立即执行(立即执行：意思是当前Goroutine不会因此操作而被阻塞)**

**select的用法与switch非常类似，由select开始一个新的选择块，每个选择条件由case语句来描述。与switch语句可以选择任何可使用相等比较的条件相比，select有比较多的限制，其中最大的一条限制就是每个case语句里必须是一个IO操作，确切的说，应该是一个面向channel的IO操作。**

**注意:case是阻塞的**

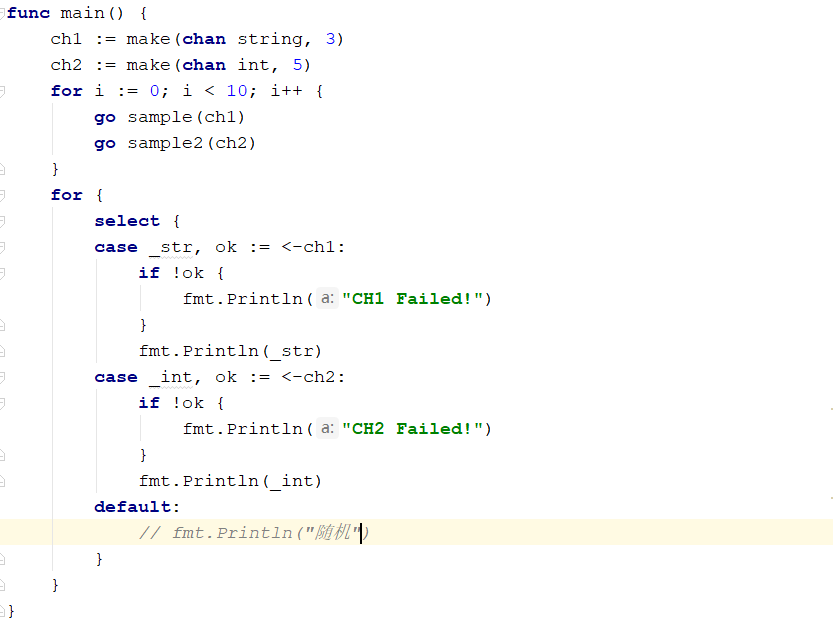
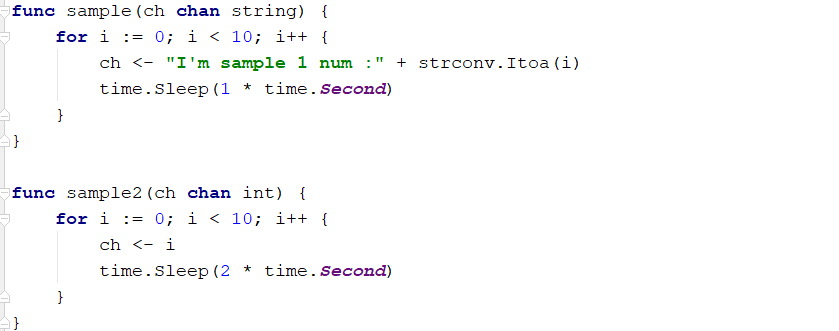
### **Select随机选择**(其实也不是随机，谁先来，谁就先解决问题)





### Select等待机制 (谁先来执行谁)

**Select是非阻塞等待，但是case是阻塞等待的，所以select加channel的模型可以形成很好的并发模型**



# 项目实战1

## 快速构建实例网站

提供了辅助包:

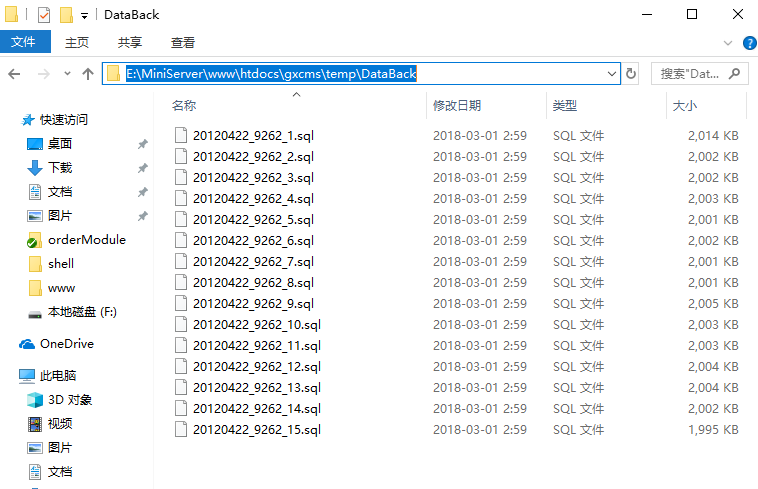


运行必须是apache大于5.1.7 小于php5.6才能运行成功，下面这个是我能正确安装的

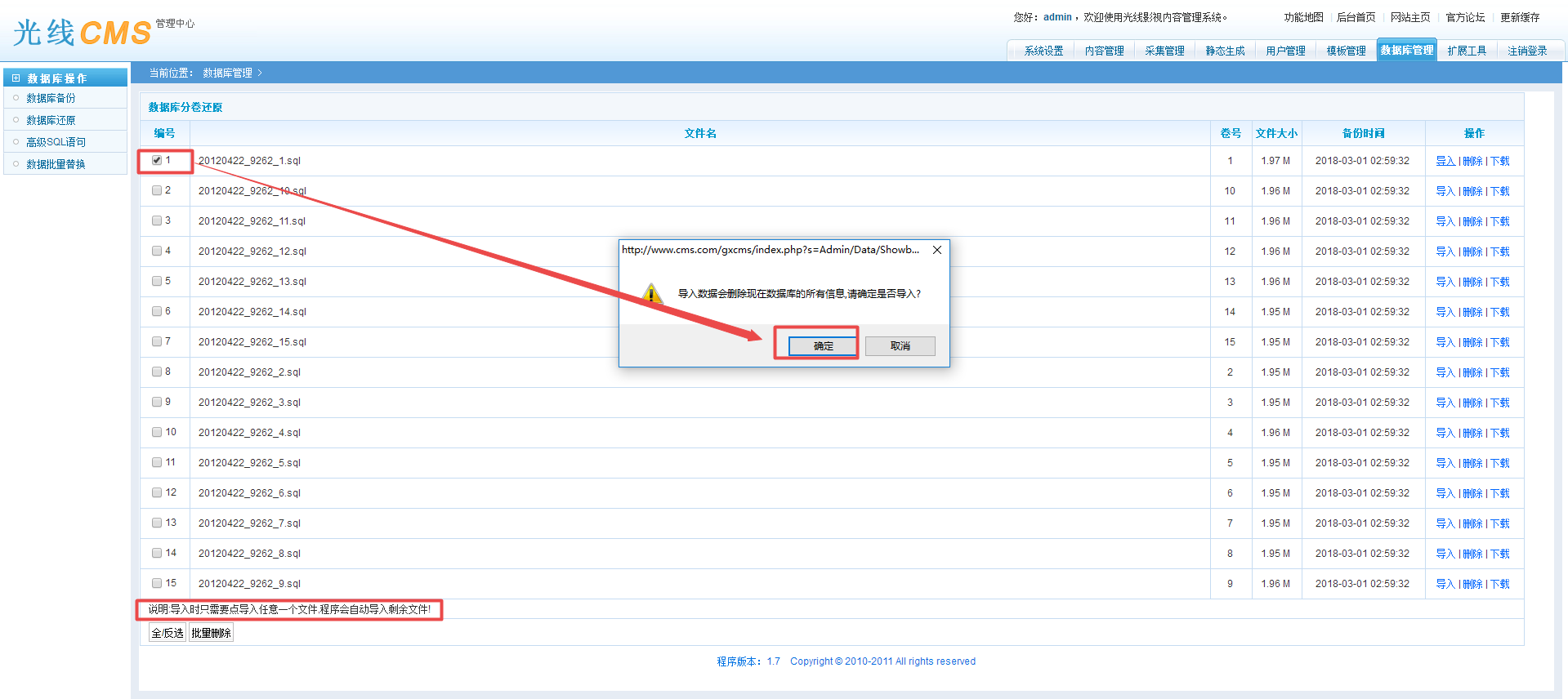


将项目代码添加到miniServer的www目录中即可

安装完成后添加测试数据,将辅助包中的文件SQLDataBack.7z解压，放到项目源码的文件夹中



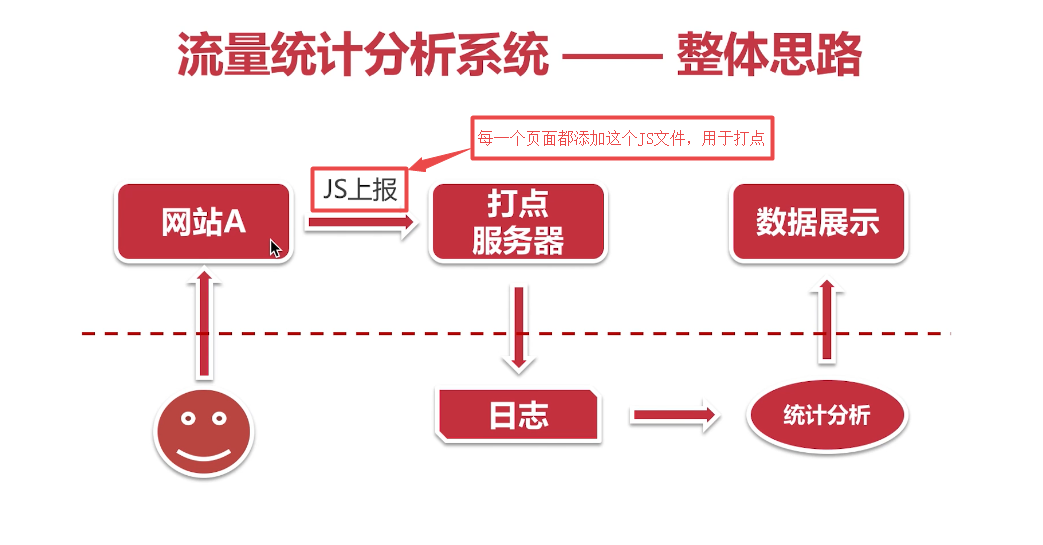
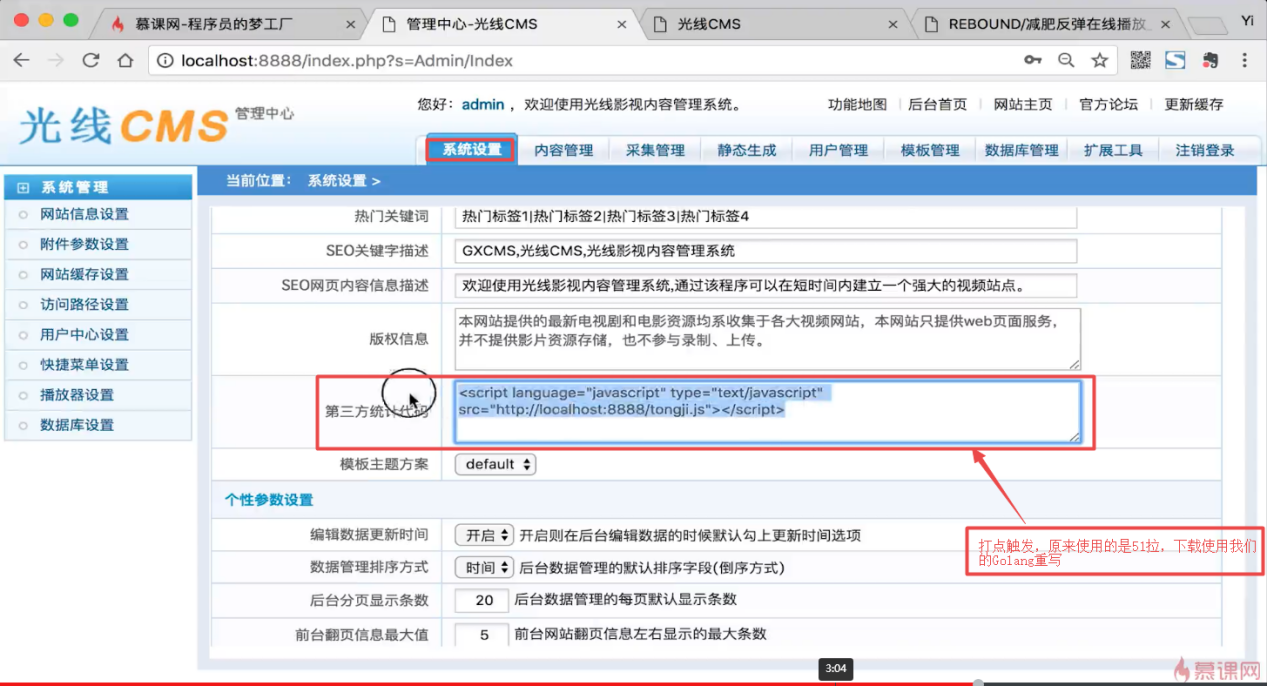
打开后台，账号/密码:admin/admin888，操作



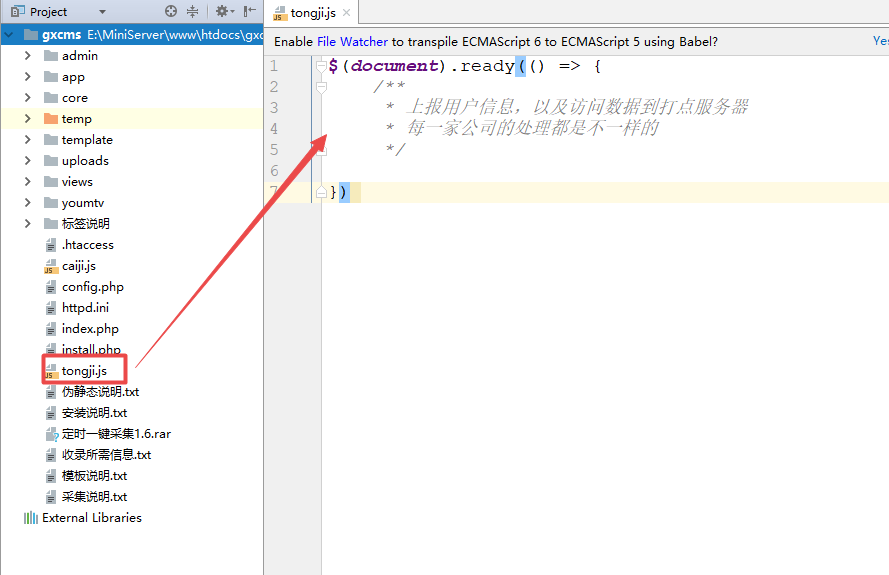


构建完成了

## 打点原理分析



部分代码



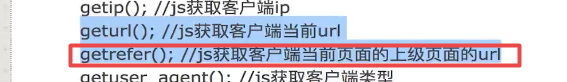
### JavaScript获取客户端访问的内容

<https://www.cnblogs.com/adolfmc/p/7698364.html>



### 打点获取客户refer(refer来流量统计里是必须要用的,重点)

可以将用户的行文串联起来

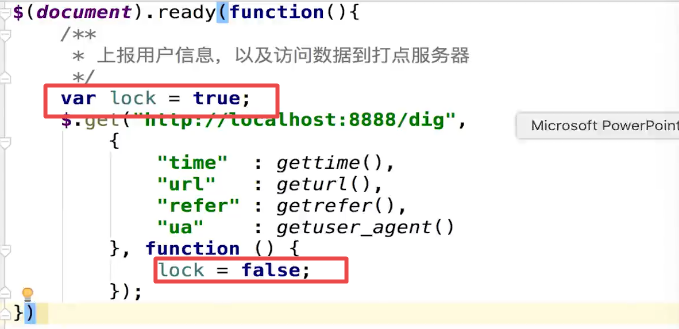


具体tongji.js的代码

AJax地址是无效的，只是用来测试一下，get是否有效



可以做一点容错处理，加锁来解决多次重复上传你的问题，当然统计学上，多一次少一次其实没有太多分别，当然精准一点肯定是好的



### 网站端的总结



## Nginx打点服务器的搭建与配置



由于打点服务器的并发量是非常高的，所以需要使用的Nginx服务器来承载请求的压力.



### 打点系统返回空白图片的意义

由于打点的并发很高，正常来说，api的返回是json等文本字符串，但是简单的json字符串很是很大的，大约800+b，但是如果采用nginx的空白1px\*1px图片大约200b，还有一个问题，由于打点其实根本不用返回内容的，但是TCP的HTTP协议必须三次握手，所以最好不要不返回东西，所以返回这个nginx空白图片更好，而且这图片一个在内存中保存的，因为模块已经打开。所以直接返回的速度很快.

### Nginx的ngx\_http\_empty\_gif\_module模块安装

windows自带了这个模块，所以不用安装

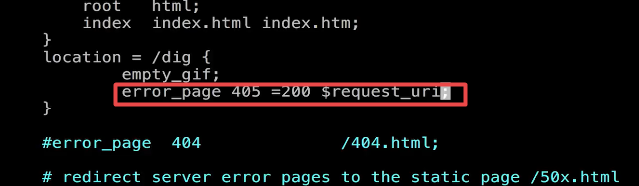
Linux暂未尝试

### 正常配置后，再用ajax打开网址，出现405异常



详细解决方案:

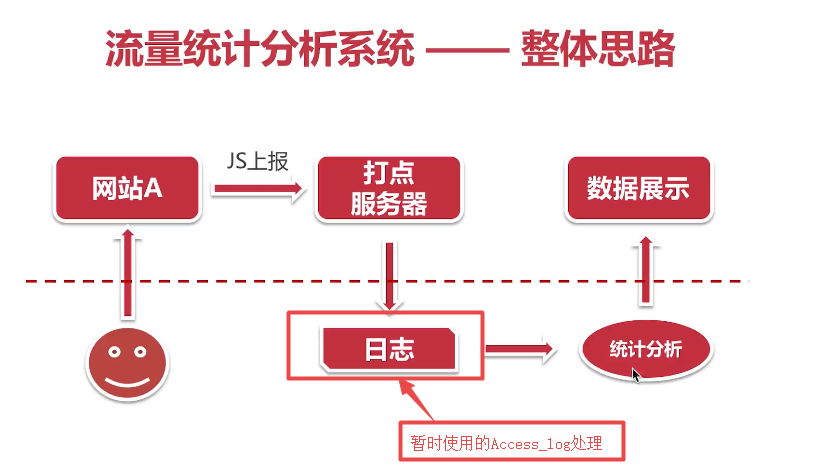
等号”=”后面没有空格(这个需要注意)



注意:更换配置后，一定要nginx.exe -s stop 不然删除了cmd还是会保留了nginx的运行，就算是新开，也不会变更执行新的配置文件.

### nginx空gif一般用于高并发而且不需要返回的单点请求

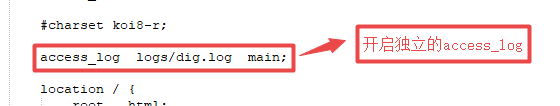
### Access\_log记录用户的访问记录



使用access\_log的原因

1. 这里的每一个环节都可以进行解耦，日志暂时使用nginx的access\_log ,如果更厉害的话，可以使用kafka,体现分层思想，不要一味吧日志也写到统计分析那里去，那里是使用go进行处理的松耦合系统.

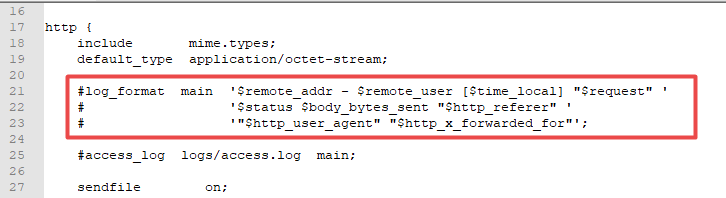
配置:



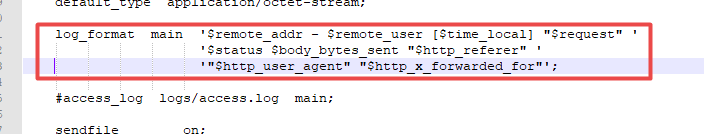
这里的main是一种日志格式

这个格式需要在http模块上定义

修改前:



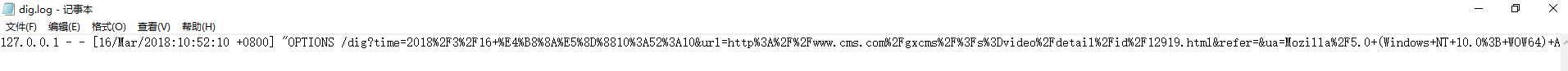
修改后:



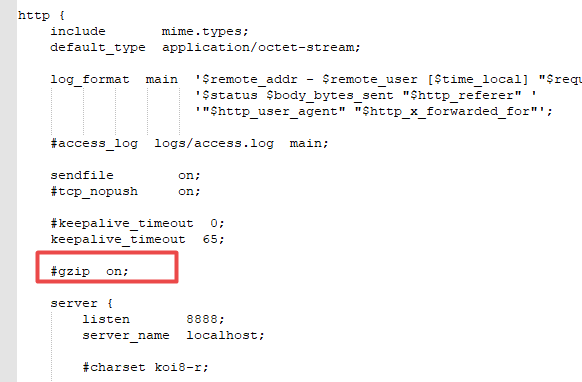
Nginx.ext -s reload重启更新配置



访问后，dig.log会出现下面的内容



### nginx简单优化技巧

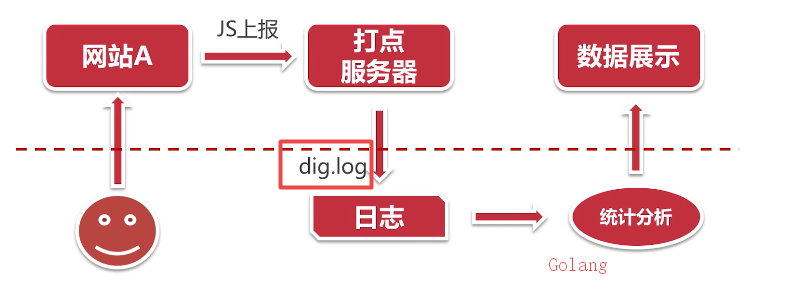


开启gzip在大一点的几百K的资源压缩有明显好处，但是小的话没什么优势.

### 做高性能的打点，一定要优化nginx

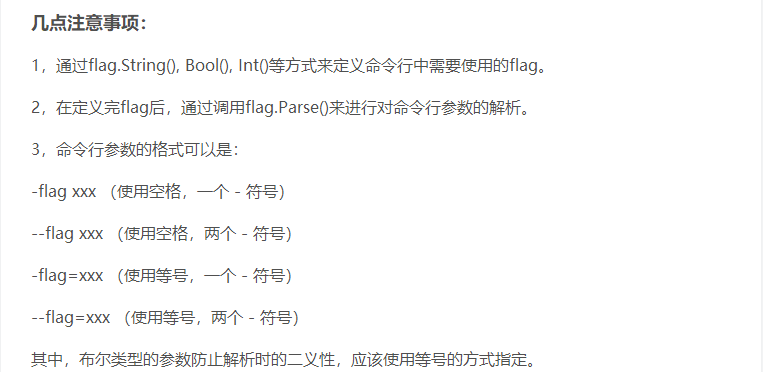
例如工作进程，gzip等优化最好开启，不调优的nginx会压力很大的.

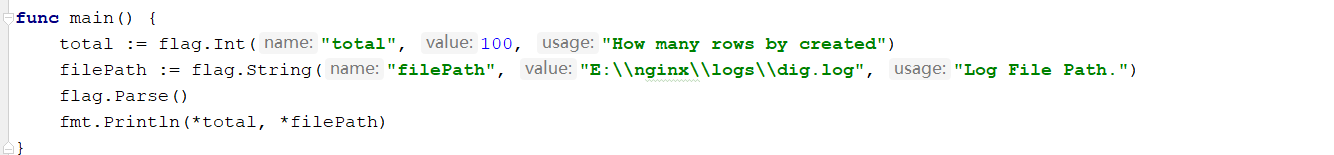
## 打点程序Golang的编写与处理(模拟生成测试数据)



### 打点程序最好添加伪静态的处理，这样统计起来比较简单

### Golang--flag包(go语言通过使用标准库里的flag包来处理命令行参数)(重点，一般用于无界面程序)

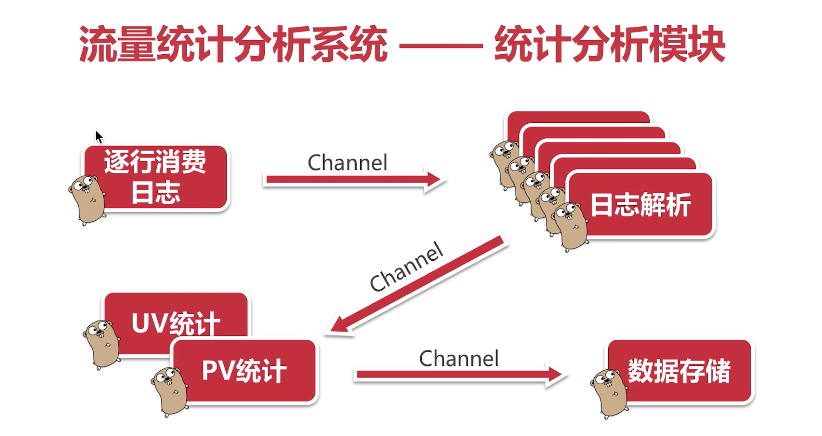




产生大量的数据的执行



## Golang统计分析模块



### 代码编辑部分

### Go get golang.org的包失败解决办法

$mkdir -p $GOPATH/src/golang.org/x/

$cd $GOPATH/src/golang.org/x/

$git clone https://github.com/golang/net.git net

包括window包,crypto包，sys包等等，都可以这样安装

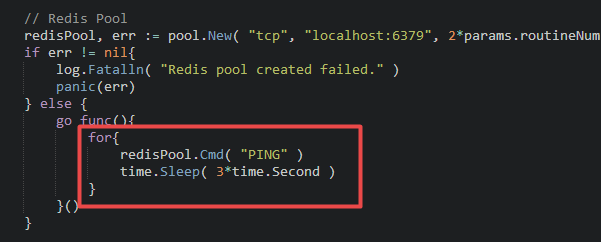
安装了这个net包，然后再去安装gin，就安装成功了。

### "github.com/mediocregopher/radix.v2/pool" --Redis使用库

Redis一定要在初始建立连接，千万不能在协程中创建连接

注意:在并发模型中执行全局变量，容易产生空指针的问题，遇到这种问题，1.资源连接的话，我们可以采用连接池的方式，2.我们可以采用在协程方法中传参的形式解决

### 为了防止连接池由于长时间没有工作而被断开，我们会起一个协程，定时访问redis

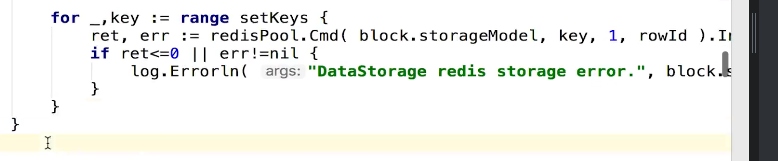


### 统计数据逐层添加，加洋葱皮的过程



这些都是需要加1的，每一个层次都是需要加1

下面的成功不做工作，出错才做工作的原因是，，这个最后的步骤数据量已经是原来的好几十倍或者上千倍，如果counter任务越多，存储压力越大，所以成功的话，我们不记录不处理



### Hase常用于大企业高并发的数据存储，Mysql不太好

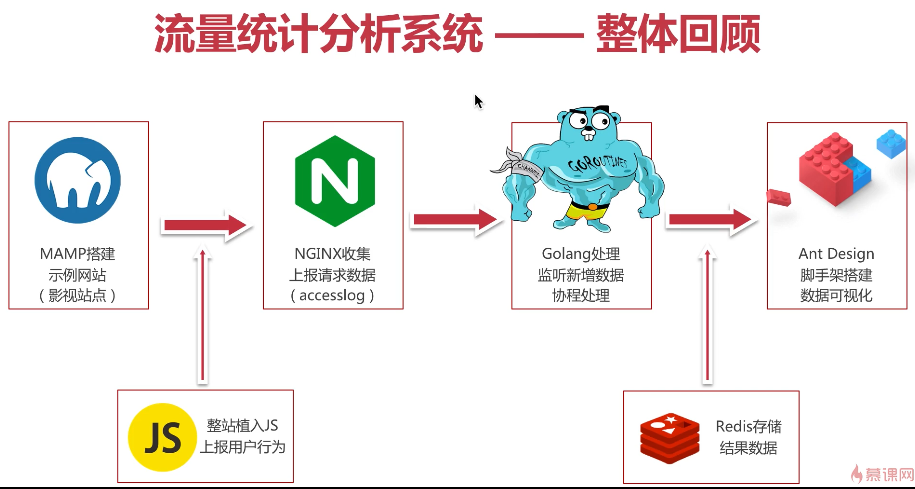
## 使用蚂蚁金服的AntDesign能够快速开发中台的界面搭建

<https://preview.pro.ant.design/#/dashboard/monitor>

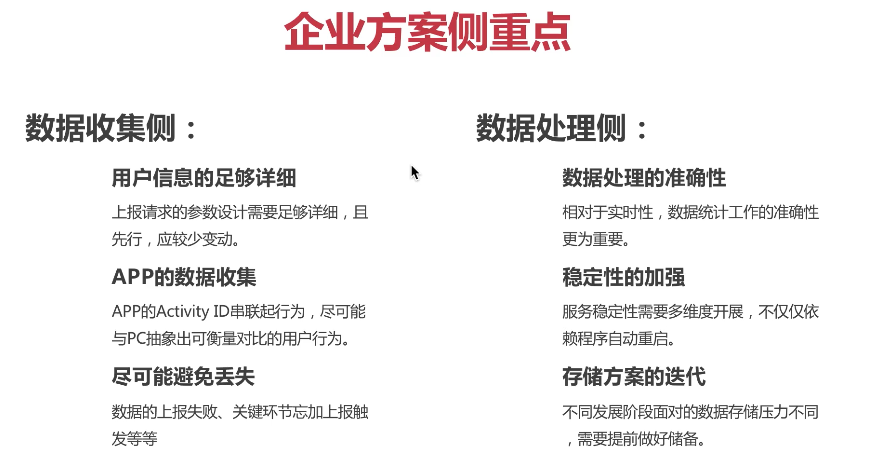
后端采用Go来采集和处理数据

使用PHP开发api来对外进行服务的输出来作为页面呈现的方法

# 流量统计整体回顾



## 企业方案的侧重点



一般手机的统计是基于Activity ID来串联起整个统计的

一般都由于操作太频繁，所以我们可能会集合成为一个本地存储的小队列，操作完10-20次左右的 操作再去上报，即本地缓存队列，这样我们的服务器记录的被访问的并发压力会少一点.

关键环节需要常常丢失，所以需要测试

小样本可以丢掉,不会影响统计，但是整个统计是准确性和稳定性是最重要的，因为只要这样才能为公司带来营收

可能统计来的数据就100条，但是通过不同为的处理就有n条(pv,uv,...还有很多其他的业务处理)，再来在存储的时候，要存储不同维度的内容，如果有M种格式，那么我么需要存储的条数可能就会到达100\*n\*m的海量数据，所以我们需要对存储做一定的优化，少一点的可以redis，但是大的很厉害的，我们采用分布式HBASE等方案来应对处理

## 课程总结

