## 网站用户行为日志采集和后台日志服务器搭建

### 1.简介

网站流量数据统计分析，可以帮助网站管理员、运营人员、推广人员等实时获取网站流量信息，并从流量来源、网站内容、网站访客特性等多方面提供网站分析的数据依据。从而帮助提高网站流量，提升网站用户体验，让访客更多的沉淀下来变成会员或客户，通过更少的投入获取最大化的收入。

访问日志指用户访问网站时的所有访问、浏览、点击行为数据。比如点击了哪一个链接，打开了哪一个页面，采用了哪个搜索项、总体会话时间等。而所有这些信息都可通过网站日志保存下来。通过分析这些数据，可以获知许多对网站运营至关重要的信息。采集的数据越全面，分析就能越精准。

日志的生成渠道分为以下两种:

web服务器软件(httpd、nginx、tomcat)自带的日志记录功能，如Nginx的access.log日志;

自定义采集用户行为数据，通过在页面嵌入自定义的 javascript 代码来获取用户的访问行为(比如鼠标悬停的位置，点击的页面组件等)，然后通过 ajax 请求到后台记录日志，这种方式所能采集的信息会更加全面。

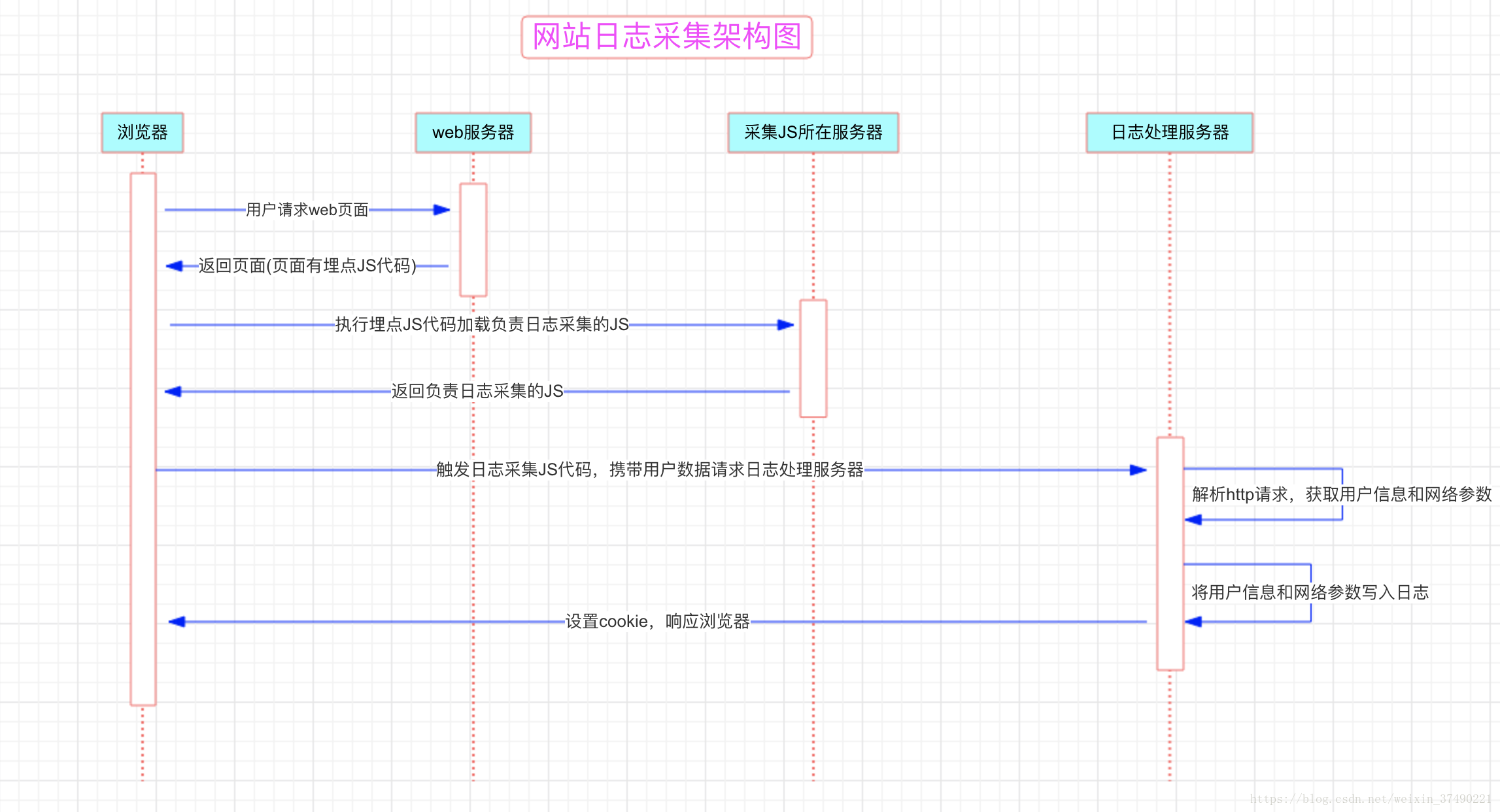
在实际操作中，有以下几个方面的数据可以自定义的采集:

|  |  |
| --- | --- |
| **类型** | **列举** |
| 系统特征 | 比如所采用的操作系统、浏览器、域名和访问速度等 |
| 访问特征 | 包括停留时间、点击的URL、所点击的“页面标签”及标签的属性等 |
| 来源特征 | 包括来访 URL，来访 IP 等 |
| 来源特征 | 包括所访问的产品编号、产品类别、产品颜色、产品价格、产品 利润、产品数量和特价等级等 |

下面是博主访问京东首页，其自定义采集的数据日志格式的截取部分:

https://mercury.jd.com/log.gif?t=www.100000&m=UA-J2011-1&pin=zaomianbao&uid=1522992113160771150291522&sid=15229913133260771150291522|42&v=je=0$sc=24-bit$sr=1680x1050$ul=en-us$cs=UTF-8$dt=京东(JD.COM)-正品低价、品质保障、配送及时、轻松购物！$hn=www.jd.com$fl=-$os=mac$br=chrome

### 2.设计架构



**埋点是指:**在网页中预先加入小段 javascript 代码，这个代码片段一般会动态创建一个 script标签，并将 src 属性指向一个单独的 js 文件，此时这个单独的 js 文件会被浏览器请求到并执行，这个 js 往往就是真正的数据收集脚本。

数据收集完成后，js 会请求一个后端的数据收集脚本，这个脚本一般是一个伪装成图片的动态脚本程序，js 会将收集到的数据通过 http 参数的方式传递给后端脚本，后端脚本解析参数并按固定格式记录到访问日志，同时可能会在 http 响应中给客户端种植一些用于追踪的 cookie。

番外：为什么使用请求图片的形式收集?

这里收集的形式其实有很多种，比如ajax请求，但是在实际生产环境中服务一般都是分布式部署，ajax请求就会涉及到跨域的问题(具体详见:同源策略与通过跨域资源共享实现ajax跨域访问)，而请求图片地址的形式则不在同源策略的限制范围内，即请求图片没有跨域限制。另一方面，请求一个静态资源的速率旺旺比请求一个动态接口要快得多。

### 3.设计实现

步骤：

1. 确定想要收集的信息
2. 确定埋点代码
3. 编写JS前端日志收集脚本
4. 确定日志格式
5. 编写后台脚本
6. 日志切分
7. 在网站页面进行埋点
8. 检测日志数据

选型：这里我们选用Apache作为Web服务器，日志后台服务器使用Nginx处理日志请求，同时将JS前端日志收集脚本直接放置在Nginx服务器内(架构中将其单独部署一个服务器)。

#### 3.1确定收集的信息

|  |  |  |
| --- | --- | --- |
| 名称 | 途径 | 备注 |
| 访问时间 | web server | Nginx $msec |
| IP | web server | Nginx $remote\_addr |
| 域名 | JavaScript | document.domain |
| URL | JavaScript | document.URL |
| 页面标题 | JavaScript | document.title |
| 分辨率 | JavaScript | window.screen.height & width |
| 颜色深度 | JavaScript | window.screen.colorDepth |
| Referrer | JavaScript | document.referrer |
| 浏览客户端 | web server | Nginx $http\_user\_agent |
| 客户端语言 | javascript | navigator.language |
| 访客标识 | cookie | Nginx $http\_cookie |
| 网站标识 | javascript | 自定义对象 |
| 状态码 | web server | Nginx $status |
| 发送内容量 | web server | Nginx $body\_bytes\_sent |

#### 3.2埋点代码

埋点，是网站分析的一种常用的数据采集方法。核心就是在需要进行数据采集的关键点植入统计代码，进行数据的采集。比如以谷歌分析原型来说，需要在页面中插入一段它提供的 javascript 片段，这个片段往往被称为埋点。

<script type="text/javascript">

var \_maq = \_maq || [];

\_maq.push(['\_setAccount', 'zaomianbao']);

(function() {

var ma = document.createElement('script');

ma.type = 'text/javascript';

ma.async = true;

ma.src = 'http://192.168.214.152/ma.js';

var s = document.getElementsByTagName('script')[0];

s.parentNode.insertBefore(ma, s);

})();

</script>

其中\_maq 是全局数组，用于放置各种配置，其中每一条配置的格式为:

\_maq.push([‘Action’, ‘param1’, ‘param2’, …]);

\_maq 的机制不是重点，重点是后面匿名函数的代码，这段代码的主要目的

就是引入一个外部的 js 文件(ma.js)，方式是通过 document.createElement 方法 创建一个 script 并根据协议(http 或 https)将 src 指向对应的 ma.js，最后将这个元素插入页面的 dom 树上。

注意 ma.async = true 的意思是异步调用外部 js 文件，即不阻塞浏览器的解 析，待外部 js 下载完成后异步执行。这个属性是 HTML5 新引入的。

番外：js 自调用匿名函数

格式: (function(){})();

第一对括号向脚本返回未命名的函数;后一对空括号立即执行返回的未命名函数，括号内为匿名函数的参数。

自调用匿名函数的好处是，避免重名，自调用匿名函数只会在运行时执行一 次，一般用于初始化。

#### 3.3前端数据收集脚本

数据收集脚本(ma.js)被请求后会被执行，一般要做如下几件事:

1、通过浏览器内置 javascript 对象收集信息，如页面 title(通过 document.title)、 referrer(上一跳 url，通过 document.referrer)、用户显示器分辨率(通过 windows.screen)、cookie 信息(通过 document.cookie)等等一些信息。

2、解析\_maq 数组，收集配置信息。这里面可能会包括用户自定义的事件跟 踪、业务数据(如电子商务网站的商品编号等)等。

3、将上面两步收集的数据按预定义格式解析并拼接(get 请求参数)。

4、请求一个后端脚本，将信息放在 http request 参数中携带给后端脚本。

这里唯一的问题是步骤 4，javascript 请求后端脚本常用的方法是 ajax，但是ajax 是不能跨域请求的。一种通用的方法是 js 脚本创建一个 Image 对象，将 Image 对象的 src 属性指向后端脚本并携带参数，此时即实现了跨域请求后端。这也是 后端脚本为什么通常伪装成 gif 文件的原因。

示例JS代码：

(function () {

var params = {};

//Document对象数据

if(document) {

params.domain = document.domain || '';

params.url = document.URL || '';

params.title = document.title || '';

params.referrer = document.referrer || '';

}

//Window对象数据

if(window && window.screen) {

params.sh = window.screen.height || 0;

params.sw = window.screen.width || 0;

params.cd = window.screen.colorDepth || 0;

}

//navigator对象数据

if(navigator) {

params.lang = navigator.language || '';

}

//解析\_maq配置

if(\_maq) {

for(var i in \_maq) {

switch(\_maq[i][0]) {

case '\_setAccount':

params.account = \_maq[i][1];

break;

default:

break;

}

}

}

//拼接参数串

var args = '';

for(var i in params) {

if(args != '') {

args += '&';

}

args += i + '=' + encodeURIComponent(params[i]);

}

//通过Image对象请求后端脚本

var img = new Image(1, 1);

img.src = 'http://192.168.214.152/log.gif?' + args;

})();

整个脚本放在匿名函数里，确保不会污染全局环境。其中 log.gif 表面上请求静态资源，到了Nginx后台实则为一个后台脚本

#### 3.4日志格式

日志格式主要考虑日志分隔符，一般会有以下几种选择:固定数量的字符、制表符分隔符、空格分隔符、其他一个或多个字符、特定的开始和结束文本。

我们在 nginx 的配置文件中定义日志格式:

log\_format

"$msec||$remote\_addr||$status||$body\_bytes\_sent||$u\_domain||$u\_url|

|$u\_title||$u\_referrer||$u\_sh||$u\_sw||$u\_cd||$u\_lang||$http\_user\_ag

ent||$u\_account";

注意这里以 u\_开头的是我们待会会自己定义的变量，其它的是nginx内置变量

#### 3.5后台脚本

log.gif 是后端脚本，是一个伪装成 gif 图片的脚本。后端脚本一般需要完 成以下几件事情:

1、解析 http 请求参数得到信息。

2、从 Web 服务器中获取一些客户端无法获取的信息，如访客 ip 等。

3、将信息按格式写入 log。

4、生成一副 1×1 的空 gif 图片作为响应内容并将响应头的 Content-type设为 image/gif。

5、在响应头中通过 Set-cookie 设置一些需要的 cookie 信息。

之所以要设置 cookie 是因为如果要跟踪唯一访客，通常做法是如果在请求 时发现客户端没有指定的跟踪 cookie，则根据规则生成一个全局唯一的 cookie 并 种植给用户，否则 Set-cookie 中放置获取到的跟踪 cookie 以保持同一用户 cookie 不变。这种做法虽然不是完美的(例如用户清掉 cookie 或更换浏览器会被认为是两个用户)，但是目前被广泛使用的手段。

我们使用 nginx 的 access\_log 做日志收集，不过有个问题就是 nginx 配置本身的逻辑表达能力有限，所以选用 OpenResty 做这个事情。

番外：什么是OpenResty?

OpenResty是一个基于 Nginx 与 Lua 的高性能 Web 平台，其内部集成了大量精良的 Lua 库、第三方模块以及大多数的依赖项。其中的核心是通过 ngx\_lua 模块集成了 Lua，从而在 nginx 配置文 件中可以通过 Lua 来表述业务。而Lua 是一种轻量小巧的脚本语言，用标准 C 语言编写并以源代码形式开放， 其设计目的是为了嵌入应用程序中，从而为应用程序提供灵活的扩展和定制功能。

这里给出Nginx配置文件

worker\_processes 2;

events {

worker\_connections 1024;

}

http {

include mime.types;

default\_type application/octet-stream;

log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" '

'$status $body\_bytes\_sent "$http\_referer" '

'"$http\_user\_agent" "$http\_x\_forwarded\_for"';

log\_format user\_log\_format "$msec||$remote\_addr||$status||$body\_bytes\_sent||$u\_domain||$u\_url||$u\_title||$u\_referrer||$u\_sh||$u\_sw||$u\_cd||$u\_lang||$http\_user\_agent||$u\_account";

sendfile on; #允许sendfile方式传输文件，默认为off

keepalive\_timeout 65; #连接超时时间，默认为75s

server {

listen 80;

server\_name localhost;

location /log.gif {

#伪装成gif文件

default\_type image/gif;

#nginx本身记录的access\_log，日志格式为main

access\_log logs/access.log main;

access\_by\_lua "

-- 用户跟踪cookie名为\_\_utrace

local uid = ngx.var.cookie\_\_\_utrace

if not uid then

-- 如果没有则生成一个跟踪cookie，算法为md5(时间戳+IP+客户端信息)

uid = ngx.md5(ngx.now() .. ngx.var.remote\_addr .. ngx.var.http\_user\_agent)

end

ngx.header['Set-Cookie'] = {'\_\_utrace=' .. uid .. '; path=/'}

if ngx.var.arg\_domain then

-- 通过subrequest到/i-log记录日志，将参数和用户跟踪cookie带过去

ngx.location.capture('/i-log?' .. ngx.var.args .. '&utrace=' .. uid)

end

";

#此请求资源本地不缓存

add\_header Expires "Fri, 01 Jan 1980 00:00:00 GMT";

add\_header Pragma "no-cache";

add\_header Cache-Control "no-cache, max-age=0, must-revalidate";

#返回一个1×1的空gif图片

empty\_gif;

}

location /i-log {

#内部location，不允许外部直接访问

internal;

#设置变量，注意需要unescape

set\_unescape\_uri $u\_domain $arg\_domain;

set\_unescape\_uri $u\_url $arg\_url;

set\_unescape\_uri $u\_title $arg\_title;

set\_unescape\_uri $u\_referrer $arg\_referrer;

set\_unescape\_uri $u\_sh $arg\_sh;

set\_unescape\_uri $u\_sw $arg\_sw;

set\_unescape\_uri $u\_cd $arg\_cd;

set\_unescape\_uri $u\_lang $arg\_lang;

set\_unescape\_uri $u\_account $arg\_account;

#打开subrequest（子请求）日志

log\_subrequest on;

#自定义采集的日志，记录数据到user\_defined.log

access\_log logs/user\_defined.log user\_log\_format;

#输出空字符串

echo '';

}

}

}

### 4.系统环境搭建部署

注：这里是对192.168.214.152日志处理Nignx服务器的搭建，对于web服务器，直接在192.168.214.150服务器上执行yum install httpd即可安装成功。

#### 4.1准备安装包并上传服务器

这里自己准备需要的安装包——注：Linux上传方式：三种方式实现Linux的文件上传下载

[root@centos6-3 software]# cd /export/software/

[root@centos6-3 software]# ll

total 5900

//一个Lua包，一个nginx包，一个openresty包，4个nginx模块包

-rw-r--r--. 1 root root 64779 Jun 26 08:38 echo-nginx-module-0.58.tar.gz

-rw-r--r--. 1 root root 847615 Jun 26 08:38 LuaJIT-2.0.4.tar.gz

-rw-r--r--. 1 root root 569372 Jun 26 08:41 lua-nginx-module-0.10.0.tar.gz

-rw-r--r--. 1 root root 833473 Jun 26 08:38 nginx-1.8.1.tar.gz

-rw-r--r--. 1 root root 65029 Jun 26 08:38 ngx\_devel\_kit-0.2.19.tar.gz

-rw-r--r--. 1 root root 3616491 Jun 26 08:40 openresty-1.9.7.3.tar.gz

-rw-r--r--. 1 root root 35495 Jun 26 08:38 set-misc-nginx-module-0.29.tar.gz

[root@centos6-3 software]#

#### 4.2依赖安装

[root@centos6-3 software]# yum -y install gcc perl pcre-devel openssl openssl-devel

#### 4.3解压编译安装LuaJIT

[root@centos6-3 software]# tar -zxvf LuaJIT-2.0.4.tar.gz -C /usr/local/src/

LuaJIT-2.0.4/

LuaJIT-2.0.4/COPYRIGHT

LuaJIT-2.0.4/Makefile

LuaJIT-2.0.4/README

LuaJIT-2.0.4/doc/

LuaJIT-2.0.4/doc/bluequad-print.css

....

[root@centos6-3 software]# cd /usr/local/src/LuaJIT-2.0.4/

[root@centos6-3 LuaJIT-2.0.4]# make && make install PREFIX=/usr/local/luajit

==== Building LuaJIT 2.0.4 ====

make -C src

...

[root@centos6-3 LuaJIT-2.0.4]#

#### 4.4设置LuaJIT环境变量

[root@centos6-3 LuaJIT-2.0.4]# vi /etc/profile

//文件末尾追加以下内容

export LUAJIT\_LIB=/usr/local/luajit/lib

export LUAJIT\_INC=/usr/local/luajit/include/luajit-2.0

//更新环境变量

[root@centos6-3 LuaJIT-2.0.4]# source /etc/profile

[root@centos6-3 LuaJIT-2.0.4]#

#### 4.5解压modules

[root@centos6-3 LuaJIT-2.0.4]# mkdir -p /usr/local/nginx/modules

[root@centos6-3 LuaJIT-2.0.4]# cd /export/software/

[root@centos6-3 software]# mv set-misc-nginx-module-0.29.tar.gz /usr/local/nginx/modules/

[root@centos6-3 software]# mv lua-nginx-module-0.10.0.tar.gz /usr/local/nginx/modules/

[root@centos6-3 software]# mv ngx\_devel\_kit-0.2.19.tar.gz /usr/local/nginx/modules/

[root@centos6-3 software]# mv echo-nginx-module-0.58.tar.gz /usr/local/nginx/modules/

[root@centos6-3 software]# cd /usr/local/nginx/modules/

[root@centos6-3 modules]# ll

total 724

-rw-r--r--. 1 root root 64779 Jun 26 08:38 echo-nginx-module-0.58.tar.gz

-rw-r--r--. 1 root root 569372 Jun 26 08:41 lua-nginx-module-0.10.0.tar.gz

-rw-r--r--. 1 root root 65029 Jun 26 08:38 ngx\_devel\_kit-0.2.19.tar.gz

-rw-r--r--. 1 root root 35495 Jun 26 08:38 set-misc-nginx-module-0.29.tar.gz

[root@centos6-3 modules]# tar -zxvf lua-nginx-module-0.10.0.tar.gz

[root@centos6-3 modules]# tar -zxvf set-misc-nginx-module-0.29.tar.gz

[root@centos6-3 modules]# tar -zxvf ngx\_devel\_kit-0.2.19.tar.gz

[root@centos6-3 modules]# tar -zxvf echo-nginx-module-0.58.tar.gz

[root@centos6-3 modules]# ll

total 740

drwxrwxr-x. 6 root root 4096 Jun 22 2015 echo-nginx-module-0.58

-rw-r--r--. 1 root root 64779 Jun 26 08:38 echo-nginx-module-0.58.tar.gz

drwxrwxr-x. 9 root root 4096 Jan 12 2016 lua-nginx-module-0.10.0

-rw-r--r--. 1 root root 569372 Jun 26 08:41 lua-nginx-module-0.10.0.tar.gz

drwxrwxr-x. 9 root root 4096 Sep 26 2013 ngx\_devel\_kit-0.2.19

-rw-r--r--. 1 root root 65029 Jun 26 08:38 ngx\_devel\_kit-0.2.19.tar.gz

drwxrwxr-x. 6 root root 4096 Jun 22 2015 set-misc-nginx-module-0.29

-rw-r--r--. 1 root root 35495 Jun 26 08:38 set-misc-nginx-module-0.29.tar.gz

[root@centos6-3 modules]# rm -rf \*.tar.gz

[root@centos6-3 modules]# ll

total 16

drwxrwxr-x. 6 root root 4096 Jun 22 2015 echo-nginx-module-0.58

drwxrwxr-x. 9 root root 4096 Jan 12 2016 lua-nginx-module-0.10.0

drwxrwxr-x. 9 root root 4096 Sep 26 2013 ngx\_devel\_kit-0.2.19

drwxrwxr-x. 6 root root 4096 Jun 22 2015 set-misc-nginx-module-0.29

[root@centos6-3 modules]#

#### 4.6安装OpenResty

[root@centos6-3 modules]# cd /export/software/

[root@centos6-3 software]# ll

total 5176

-rw-r--r--. 1 root root 847615 Jun 26 08:38 LuaJIT-2.0.4.tar.gz

-rw-r--r--. 1 root root 833473 Jun 26 08:38 nginx-1.8.1.tar.gz

-rw-r--r--. 1 root root 3616491 Jun 26 08:40 openresty-1.9.7.3.tar.gz

[root@centos6-3 software]# tar -zxvf openresty-1.9.7.3.tar.gz -C /usr/local/src/

[root@centos6-3 software]# cd /usr/local/src/openresty-1.9.7.3/

[root@centos6-3 openresty-1.9.7.3]# ./configure --prefix=/usr/local/openresty --with-luajit && make && make install

[root@centos6-3 openresty-1.9.7.3]#

#### 4.7解压编译安装Nginx

[root@centos6-3 openresty-1.9.7.3]# cd /export/software/

[root@centos6-3 software]# ll

total 5176

-rw-r--r--. 1 root root 847615 Jun 26 08:38 LuaJIT-2.0.4.tar.gz

-rw-r--r--. 1 root root 833473 Jun 26 08:38 nginx-1.8.1.tar.gz

-rw-r--r--. 1 root root 3616491 Jun 26 08:40 openresty-1.9.7.3.tar.gz

[root@centos6-3 software]# tar -zxvf nginx-1.8.1.tar.gz -C /usr/local/src/

[root@centos6-3 software]# cd /usr/local/src/nginx-1.8.1/

[root@centos6-3 nginx-1.8.1]# ./configure --prefix=/usr/local/nginx \

> --with-ld-opt="-Wl,-rpath,/usr/local/luajit/lib" \

> --add-module=/usr/local/nginx/modules/ngx\_devel\_kit-0.2.19 \

> --add-module=/usr/local/nginx/modules/lua-nginx-module-0.10.0 \

> --add-module=/usr/local/nginx/modules/set-misc-nginx-module-0.29 \

> --add-module=/usr/local/nginx/modules/echo-nginx-module-0.58

make -j2

[root@centos6-3 nginx-1.8.1]# make install

#### 4.8启动检测安装

[root@centos6-3 nginx-1.8.1]# cd /usr/local/nginx/

[root@centos6-3 nginx]# ll

total 20

drwxr-xr-x. 2 root root 4096 Jul 3 17:09 conf

drwxr-xr-x. 2 root root 4096 Jul 3 17:09 html

drwxr-xr-x. 2 root root 4096 Jul 3 17:09 logs

drwxr-xr-x. 6 root root 4096 Jul 3 16:59 modules

drwxr-xr-x. 2 root root 4096 Jul 3 17:09 sbin

[root@centos6-3 nginx]# ./sbin/nginx -c conf/nginx.conf

[root@centos6-3 nginx]#

访问192.168.214.152(默认80端口)



后台日志处理Nginx服务器搭建成功！

### 5.实施采集测试

#### 5.1编写index.html

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>枣面包的面包坊</title>

<script type="text/javascript">

var \_maq = \_maq || [];

\_maq.push(['\_setAccount', 'zaomianbao']);

(function() {

var ma = document.createElement('script');

ma.type = 'text/javascript';

ma.async = true;

ma.src = 'http://192.168.214.152/ma.js';

var s = document.getElementsByTagName('script')[0];

s.parentNode.insertBefore(ma, s);

})();

</script>

</head>

<body>

<h1 align="center">枣面包的面包坊</h1>

</body>

</html>

#### 5.2部署index.html和ma.js

在192.168.214.150的Apache服务器部署index.html

上传index.html到Apache目录下:/var/www/html

注：Linux上传方式：三种方式实现Linux的文件上传下载

[root@centos6-1 html]# rz

rz waiting to receive.

Starting zmodem transfer. Press Ctrl+C to cancel.

Transferring index.html...

100% 568 bytes 568 bytes/sec 00:00:01 0 Errors

[root@centos6-1 html]# ls

index.html

[root@centos6-1 html]#

在192.168.214.152的Nginx服务器部署ma.js

上传ma.js到Nginx目录下:/usr/local/nginx/html

[root@centos6-3 html]# rz

rz waiting to receive.

Starting zmodem transfer. Press Ctrl+C to cancel.

Transferring ma.js...

100% 1 KB 1 KB/sec 00:00:01 0 Errors

[root@centos6-3 html]# ll

total 11

-rw-r--r--. 1 root root 537 Jul 3 17:09 50x.html

-rw-r--r--. 1 root root 1249 Jul 3 2018 ma.js

#### 5.3修改Nginx配置文件

将nginx.conf文件修改成上面3.5的内容，这里不做重复，读者在按照博主上面系统环境搭建部署完成后，可以直接复制配置内容使用，可以不必对配置中使用到的lua语义过分深究

5.4启动Apache，Nginx

启动192.168.214.150下的Apache

[root@centos6-1 html]# httpd

[root@centos6-1 html]# ps -ef|grep httpd

root 42392 1 0 19:41 ? 00:00:00 httpd

apache 42393 42392 0 19:41 ? 00:00:00 httpd

apache 42394 42392 0 19:41 ? 00:00:00 httpd

apache 42395 42392 0 19:41 ? 00:00:00 httpd

apache 42396 42392 0 19:41 ? 00:00:00 httpd

apache 42397 42392 0 19:41 ? 00:00:00 httpd

apache 42398 42392 0 19:41 ? 00:00:00 httpd

apache 42399 42392 0 19:41 ? 00:00:00 httpd

apache 42400 42392 0 19:41 ? 00:00:00 httpd

root 42404 42010 0 19:42 pts/0 00:00:00 grep httpd

[root@centos6-1 html]#

启动192.168.214.152下的Nginx

[root@centos6-3 nginx]# sbin/nginx -c conf/nginx.conf

[root@centos6-3 nginx]# ps -ef|grep nginx

root 50763 1 0 19:31 ? 00:00:00 nginx: master process sbin/nginx -c conf/nginx.conf

nobody 50764 50763 0 19:31 ? 00:00:00 nginx: worker process

nobody 50765 50763 0 19:31 ? 00:00:00 nginx: worker process

root 50792 50325 0 19:43 pts/0 00:00:00 grep nginx

#### 5.5请求测试观察

在192.168.214.152的nginx目录下执行命令实时观察日志变化

[root@centos6-3 nginx]# tail -f logs/user\_defined.log

页面访问192.168.214.150的index.html页面触发

这里写图片描述

发现请求到了ma.js文件，同时发送了log.gif请求并且状态为200，即请求成功。再去刚才实时观察的shell窗口发现日志已经被收集

[root@centos6-3 nginx]# tail -f logs/user\_defined.log

1530618381.695||192.168.214.1||200||0||192.168.214.150||http://192.168.214.150/||\xE6\x9E\xA3\xE9\x9D\xA2\xE5\x8C\x85\xE7\x9A\x84\xE9\x9D\xA2\xE5\x8C\x85\xE5\x9D\x8A||||1050||1680||24||en-US||Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_13\_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.99 Safari/537.36||zaomianbao

### 6.总结

到这里一个整体的结构就成功构建好了，后面使用就行了，当然，这里只有访问页面的触发，并没有添加用户点击事件，这个留给读者在使用中根据业务添加即可。以上内容并没有假如日志切分的展现，读者直接添加即可。本篇博客的架构设计经过优化是可以运用到生产实际当中的。如有错误，欢迎指正。

参考资料：

http://openresty.org/en/

http://nginx.org/en/download.html

https://blog.csdn.net/weixin\_37490221/article/details/80849163

http://www.lua.org/about.html

http://www.runoob.com/lua/lua-tutorial.html

以及博主学习使用的PDF文档