

分布式计算第一次上机作业

姓名：张泽群

学号：19049100002

课程号：CS205105

1.概念介绍

- **1.正向代理：**客户端想要访问一个服务器，但是它可能无法直接访问这台服务器，这时候这可找一台可以访问目标服务器的另外一台服务器，而这两台服务器就被当做是代理人的角色，称之为代理服务器，于是客户端把请求发给代理服务器，由代理服务器获得目标服务器的数据并返回给客户端。客户端是清楚目标服务器的地址的，而目标服务器是不清楚来自客户端，它只知道来自哪个代理服务器，所以正向代理可以屏蔽或隐藏客户端的信息。
- **2.反向代理：**从上面的正向代理，你会大概知道代理服务器是为客户端作代理人，它是站在客户端这边的。其实反向代理就是代理服务器为服务器作代理人，站在服务器这边，它就是对外屏蔽了服务器的信息，常用的场景就是多台服务器分布式部署，像一些大的网站，由于访问人数很多，就需要多台服务器来解决人数多的问题，这时这些服务器就由一个反向代理服务器来代理，客户端发来请求，先由反向代理服务器，然后按一定的规则分发到明确的服务器，而客户端不知道是哪台服务器。常常用nginx来作反向代理。
- **3.Nginx的负载均衡：**

负载：就是Nginx接受请求

均衡：Nginx将收到的请求按照一定的规则分发到不同的服务器进行处理

2.负载均衡调度算法

Nginx支持的负载均衡调度算法方式如下：

- **1.轮询**：接收到的请求按照顺序逐一且循环分配到不同的后端服务器，某一台后端服务器宕机，能自动剔除。虽然这种方式简便、成本低廉。但缺点是：可靠性低和负载分配不均衡。适用于图片服务器集群和纯静态页面服务器集群。
- **2.指定权重**：接收到的请求按照顺序逐一分配到不同的后端服务器，即使在使用过程中，某一台后端服务器宕机，Nginx会自动将该服务器剔除出队列，请求受理情况不会受到任何影响。这种方式下，可以给不同的后端服务器设置一个权重值（weight），用于调整不同的服务器上请求的分配率；权重数据越大，被分配到请求的几率越大；该权重，主要是针对实际工作环境中不同的后端服务器硬件配置进行调整的。
- **3.IP哈希(基于一致性随机散列函数)**：每个请求按照发起客户端的ip的hash结果进行匹配，这样的算法下一个固定ip地址的客户端总会访问到同一个后端服务器，这也在一定程度上解决了集群部署环境下session共享的问题。
- **4.最小TCP链接数**：当有些请求占用的时间很长时，会导致其所在的后端负载较高。在这种场景下，把请求转发给连接数较少的后端，能够达到更好的负载均衡效果，可以运用least_conn算法。least_conn算法很简单，首选遍历后端集群，比较每个后端的conns/weight，选取该值最小的后端。此负载均衡策略适合请求处理时间长短不一造成服务器过载的情况。
- **5.fair 最小响应时间算法**：动态的根据后端服务器的请求处理到响应的时间进行均衡分配，响应时间短处理效率高的服务器分配到请求的概率高，响应时间长处理效率低的服务器分配到的请求少；结合了前两者优点的一种调度算法。但是需要注意的是Nginx默认不支持fair算法，如果要使用这种调度算法，请安装upstream_fair模块。
- **6.url_hash**：按照访问的url的hash结果分配请求，每个请求的url会指向后端固定的某个服务器，可以在nginx作为静态服务器的情况下提高缓存效率。同样要注意Nginx默认不支持这种调度算法，要使用的话需要安装nginx的hash软件包。

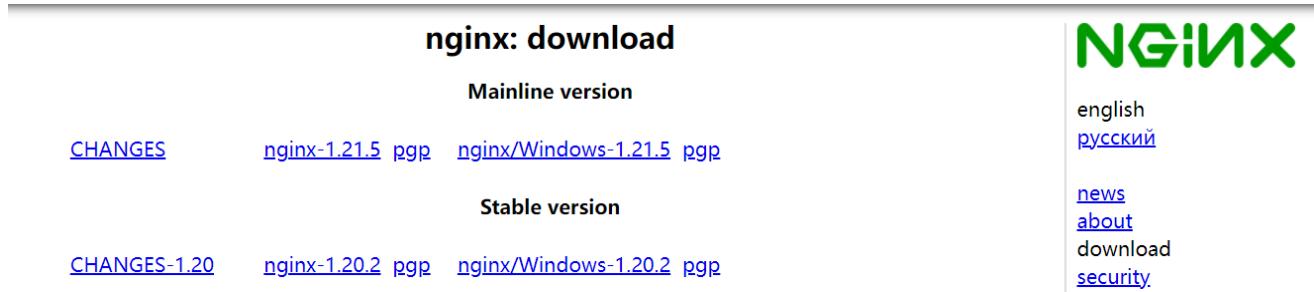
3.Nginx负载均衡调度算法实现流程

3.1 软件工具安装与简介

3.1.1 Nginx 安装

Nginx (engine x) 是一个高性能的HTTP和反向代理web服务器。

官方下载网址：<https://nginx.org/en/download.html>



这里推荐下载稳定版本 **nginx/Windows-1.20.2**，下载完毕后直接解压到自己电脑上存放nginx软件的文件夹中即可。

下载完毕后，利用命令行输入 `start nginx` 指令，启动nginx。打开浏览器访问 `http://localhost:80`，出现欢迎页就说明安装成功了。

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org. Commercial support is available at nginx.com.

Thank you for using nginx.

3.1.2 node.js 安装

node.js软件在本实验中用于配置服务器。

官方下载网址: <https://nodejs.org/zh-cn/download/>

The screenshot shows the official Node.js download page. At the top, there's a dark header with the Node.js logo and a navigation bar with links: 首页 | 关于 | 下载 | 文档 | 加入我们 | 安全 | 相关认证 | 新闻. Below the header, a large green button labeled '长期维护版' (Recommended for most users) is prominently displayed. To its right, a smaller green button labeled '最新尝鲜版' (Contains latest features) is visible. Under the '长期维护版' button, there are download links for Windows (Windows 安装包, node-v16.13.2-x64.msi), macOS (macOS 安装包, node-v16.13.2.pkg), and Source (源码, node-v16.13.2.tar.gz). A table below the Windows link shows 32-bit and 64-bit options for both Windows and macOS.

	32-bit	64-bit
Windows 安装包 (.msi)		
Windows 二进制文件 (.zip)		
macOS 安装包 (.pkg)		

这里我推荐利用安装包.`.msi`文件安装，因为`.msi`文件属于傻瓜式的一键安装，而`.zip`文件通过解压直接应用。

下载完成后，双击安装包，开始安装，一直点next即可，安装路径默认在C:\Program Files下，也可以自定义修改。

`.msi`格式的安装包已经将`node.exe`添加到系统环境变量`path`中，如果你下载的是`.zip`格式，因为没有安装过程，所以需要手动将`node.exe`所在目录添加到环境变量`path`中

3.1.3 curl 安装

curl是一个利用URL语法在命令行下工作的文件传输工具，在本实验中用于测试负载均衡算法的结果。

官方下载网址：<https://curl.haxx.se/windows/>

curl 7.81.0 for Windows

These are the latest and most up to date **official** curl binary builds for Microsoft Windows.

curl version: 7.81.0

Build: 7.81.0

Date: 2022-01-05

Changes: [7.81.0 changelog](#)

Related:

[Changelog](#)

[Downloads](#)

[FAQ](#)

[License](#)

[Manual](#)

Packages



curl for 64 bit

Size: 5.5 MB

sha256: 066b575e457e0c14182a0f96d95647bf3a9c4b0fb6caf456bab99f27f97881e8



curl for 32 bit

Size: 4.8 MB

sha256: d6f895e7a8bb83ed81812becb78229e1aadfacdf7dceee6904aabc12dba20a43

1. 下载：从官方网址中下载，并解压到自定义的软件文件夹中。

2. 配置环境变量：在系统高级环境变量中，配置

```
CURL_HOME      D:\DevelopSoftWare\DownLoadSoftWare\curl-7.56.1
path 中追加    ;%CURL_HOME%\I386
```

3. 测试安装是否成功：输入指令：curl --help 和 curl www.baidu.com 测试是否安装成功，如下图所示就是安装成功的显示

```
选择命令提示符
C:\Users\Administrator>curl --help
Usage: curl [options...] <url>
  --abstract-unix-socket <url> Connect via abstract Unix domain socket
  --anyauth      Pick any authentication method
-a, --append      Append to target file when uploading
  --basic       Use HTTP Basic Authentication
  --cacert <CA certificate> CA certificate to verify peer against
  --capath <dir>  CA directory to verify peer against
-E, --cert <certificate[:password]> Client certificate file and password
  --cert-status Verify the status of the server certificate
  --cert-type <type> Certificate file type (DER/PEM/ENG)
  --ciphers <list of ciphers> SSL ciphers to use
  --compressed Request compressed response
-K, --config <file> Read config from a file
  --connect-timeout <seconds> Maximum time allowed for connection
  --connect-to <HOST1:PORT1:HOST2:PORT2> Connect to host
-C, --continue-at <offset> Resumed transfer offset
-b, --cookie <data> Send cookies from string/file
-c, --cookie-jar <filename> Write cookies to <filename> after operation
  --create-dirs  Create necessary local directory hierarchy
  --crlf        Convert LF to CRLF in upload
  --crlfile <file> Get a CRL list in PEM format from the given file
-d, --data <data>   HTTP POST data
```

```
C:\ 命令提示符
Microsoft Windows [版本 10.0.19042.1288]
(c) Microsoft Corporation。保留所有权利。

C:\Users\Administrator>curl www.baidu.com
<!DOCTYPE html>
<!—STATUS OK—><html> <head><meta http-equiv=content-type content="text/html; charset=utf-8"><meta http-equiv=X-UA-Compatible content=IE-Edge><meta content=always name=referrer><link rel=stylesheet type=text/css href=http://s1.bdstatic.com/r/www/cache/bdorz/baidu.min.css><title>百度一下，你就知道</title></head> <body link=#0000cc> <div id=wrapper> <div id=head> <div class=head_wrapper> <div class=s_form> <div class=s_form_wrapper> <div id=lg> <img hidedefocus=true src=/www.baidu.com/img/bd_logo1.png width=270 height=129> </div> <form id=form name=f action=/www.baidu.com/s class=fm> <input type=hidden name=bdorz_come value=1> <input type=hidden name=ie value=utf-8> <input type=hidden name=f value=8> <input type=hidden name=rsv_bp value=1> <input type=hidden name=rsv_idx value=1> <input type=hidden name=tn value=baidu> <span class=bg_s_ipr_wr><input id=kw name=wd class=s_ipr valuemaxlength=255 autocomplete=off autofocus></span> <span class=bg_s_btn_wr><input type=submit id=su value=百度一下 class=bg_s_btn></span> </form> </div> <div id=u1> <a href=http://news.baidu.com name=tj_trnews class=mnav>新闻</a> <a href=http://www.hao123.com name=tj_trhao123 class=mnav>hao123</a> <a href=http://map.baidu.com name=tj_trmap class=mnav>地图</a> <a href=http://v.baidu.com name=tj_trvideo class=mnav>视频</a> <a href=http://tieba.baidu.com name=tj_trtieba class=mnav>贴吧</a> <noscript> <a href=http://www.baidu.com/bdorz/login.gif?login&tpl=mn&u=' + encodeURIComponent(window.location.href+ (window.location.search == "" ? "?" : "&") + "bdorz_come=1") + '&name=tj_login' class=lb>登录</a> </noscript> <script>document.write('<a href="http://www.baidu.com/bdorz/login.gif?login&tpl=mn&u=' + encodeURIComponent(window.location.href+ (window.location.search == "" ? "?" : "&") + "bdorz_come=1") + '&name=tj_login" class=lb>登录</a>');</script> <a href=/www.baidu.com/more/ name=tj_briicon class=bri style=display: block;>更多产品</a> </div> </div> <div id=ftCon> <div id=ftConv> <p id=lh> <a href=http://home.baidu.com>关于百度</a> <a href=http://ir.baidu.com>About Baidu</a> </p> <p id=cp>&copy;2017 Baidu&nbsp;<a href=http://www.baidu.com/duty/>使用百度前必读</a>&nbsp; <a href=http://jianyi.baidu.com/ class=cp>意见反馈</a>&nbsp;京ICP证030173号&nbsp; <img src=/www.baidu.com/img/gs.gif> </p> </div> </div> </div> </body> </html>
C:\Users\Administrator>
```

3.2 实验过程

3.2.1 配置负载均衡算法

在 `nginx.conf` 配置文件中可以添加配置：

```
35     upstream backserver {
36         server 127.0.0.1:2021 weight = 1;
37         server 127.0.0.1:2022 weight = 1;
38         server 127.0.0.1:2023 weight = 1;
39         server 127.0.0.1:2024 weight = 1;
40     }
41
42     server {
43         listen      80;
44         server_name localhost;
45
46         #charset koi8-r;
47
48         #access_log  logs/host.access.log  main;
49
50         location / {
51             root    html;
52             index  index.html index.htm;
53             proxy_pass http://backserver;
54         }
55     }
```

通过配置 `upstream` 和 `server` 中的 `location`，我们就可以配置并实验不同的负载均衡算法。

流程：

Step 1. 在http节点下，添加upstream节点，设置服务器以及使用策略 **eg.**本实验的配置有四台服务器，策略缺省。

Step 2. 将server节点下的location节点中的proxy_pass配置为：http:// + upstream名称。

eg. proxy_pass http://backserver;

注意：

- 1. 缺省配置就是轮询策略，上图所有的权重相同也是轮询策略。
- 2.nginx负载均衡支持http和https协议，只需要修改 proxy_pass后协议即可；
- 3.nginx支持FastCGI, uwsgi, SCGI,memcached的负载均衡,只需将 proxy_pass改为 fastcgi_pass, uwsgi_pass, scgi_pass,memcached_pass即可。

3.2.2 配置Web服务器功能

这里我们利用四个web服务软件 (.js) 分别在不同的端口监听，来模拟四个服务器。

在.js 文件中可以添加配置，以下是一个简单的服务器配置，用于获取反馈信息：

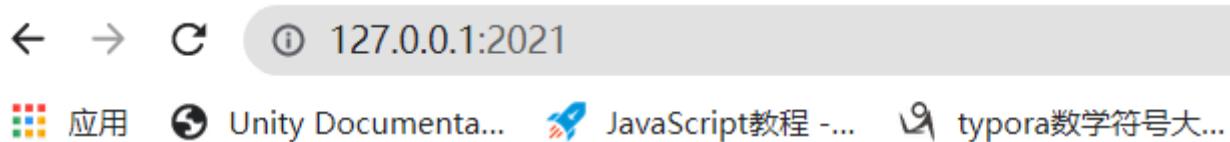
```
var http = require('http');
var server = http.createServer(function (request, response) {
  response.write('Hello from server1');
  response.end();
});
server.listen(2021);
console.log('running at http://127.0.0.1:2021');
```

3.2.3 实验步骤

- **Step 1:** 利用 nodejs 软件，在.js程序所在的目录下，利用命令行方式启动先前配置的4个web服务器软件，来启动模拟服务器。

```
D:\nodejs\webserver>node server1.js
running at http://127.0.0.1:2021
D:\nodejs\webserver>node server2.js
running at http://127.0.0.1:2022
D:\nodejs\webserver>node server3.js
running at http://127.0.0.1:2023
D:\nodejs\webserver>node server4.js
running at http://127.0.0.1:2024
```

此时已经可以通过浏览器获得响应。



- **Step 2:** 启动 nginx 软件，在nginx的安装目录下(含有nginx.exe)利用命令行输入 start nginx 指令，即可启动nginx。

名称	修改日期	类型	大小
conf	2021/11/16 22:46	文件夹	
contrib	2021/11/16 22:46	文件夹	
docs	2021/11/16 22:46	文件夹	
html	2021/11/16 22:46	文件夹	
logs	2022/1/18 0:10	文件夹	
temp	2022/1/18 0:10	文件夹	
nginx.exe	2021/11/16 22:29	应用程序	3,664 KB

```
D:\Nginx\nginx-1.20.2>start nginx
```

- **Addition:** 这里介绍几个我在运行过程中遇到的错误与解决方法。

```
D:\Nginx\nginx-1.20.2>start nginx
D:\Nginx\nginx-1.20.2>nginx -t
nginx: the configuration file D:\Nginx\nginx-1.20.2/conf/nginx.conf syntax is ok
nginx: [emerg] bind() to 0.0.0.0:8080 failed (10013: An attempt was made to access a socket in a way forbidden by its access permissions)
nginx: configuration file D:\Nginx\nginx-1.20.2/conf/nginx.conf test failed

D:\Nginx\nginx-1.20.2>nginx -s reload
nginx: [error] OpenEvent("Global\ngx_reload_15600") failed (2: The system cannot find the file specified)

D:\Nginx\nginx-1.20.2>start nginx
D:\Nginx\nginx-1.20.2>nginx -t
nginx: the configuration file D:\Nginx\nginx-1.20.2/conf/nginx.conf syntax is ok
nginx: configuration file D:\Nginx\nginx-1.20.2/conf/nginx.conf test is successful
```

(1) 如果curl指令无法检测localhost端口，首先检查配置的服务器是否能够用curl检测。一般来说可能是没有成功启动nginx。

(2) start nginx指令后可以通过nginx -t指令检查能否正常启动nginx,若没有正常启动，可以尝试修改配置server中的端口号(默认是80)并再次启动，因为很有可能是默认端口号被占用。

(3)`nginx -s reload 指令后可以通过下方的报错信息进行检查，一般来说是配置有语法错误或者未成功启动nginx。

- **Step 3:** 利用curl软件，在命令行中利用 curl localhost:8088，其中端口号是在nginx.conf 中自主定义的。

通过对于负载均衡器的监听，我们不难验证其负载均衡的算法及其效果。(如图即为轮询方法的结果)

```
C:\Users\Administrator>curl localhost:8088
Hello from server2
C:\Users\Administrator>curl localhost:8088
Hello from server3
C:\Users\Administrator>curl localhost:8088
Hello from server4
C:\Users\Administrator>curl localhost:8088
Hello from server1
C:\Users\Administrator>curl localhost:8088
Hello from server2
C:\Users\Administrator>curl localhost:8088
Hello from server3
C:\Users\Administrator>curl localhost:8088
Hello from server4
C:\Users\Administrator>curl localhost:8088
Hello from server1
C:\Users\Administrator>
```

- **Step 4:** 重新配置nginx.conf文件，使用其他负载均衡算法。

再在命令行中利用 nginx -s reload 指令加载配置的更新，然后重复**Step 3**

3.3 多种负载均衡算法的配置

3.3.1 轮询

```
upstream backserver {
    server 127.0.0.1:2021 ;
    server 127.0.0.1:2022 ;
    server 127.0.0.1:2023 ;
    server 127.0.0.1:2024 ;
}

server {
    listen      8088;
    server_name localhost;

    location / {
        root  html;
        index index.html index.htm;
        proxy_pass http://backserver;
    }
}
```

3.3.2 指定权重

```
upstream backserver {
    server 127.0.0.1:2021 weight=50;
    server 127.0.0.1:2022 weight=30;
    server 127.0.0.1:2023 weight=10;
    server 127.0.0.1:2024 weight=10;
}

server {
    listen      8088;
    server_name localhost;

    location / {
        root  html;
        index index.html index.htm;
        proxy_pass http://backserver;
    }
}
```

这里需要说明一个报错：

```
D:\Nginx\nginx-1.20.2>nginx -t
nginx: [emerg] invalid parameter "weight" in D:\Nginx\nginx-1.20.2/conf/nginx.conf:36
nginx: configuration file D:\Nginx\nginx-1.20.2/conf/nginx.conf test failed
```

在配置代理时重新加载时可能报这个错，这是因为文件格式在保存的时候有非UTF-8字符(空格) ，

需要把weight = 50， 改为weight=50。

3.3.3 IP哈希

```
upstream backserver {
    ip_hash;
    server 127.0.0.1:2021 weight=50;
    server 127.0.0.1:2022 weight=30;
    server 127.0.0.1:2023 weight=10;
    server 127.0.0.1:2024 weight=10;
}

server {
    listen      8088;
    server_name localhost;

    location / {
        root  html;
        index index.html index.htm;
        proxy_pass http://backserver;
    }
}
```

3.3.4 最小TCP链接数

```
upstream backserver {
    least_conn;
    server 127.0.0.1:2021 weight=50;
    server 127.0.0.1:2022 weight=30;
    server 127.0.0.1:2023 weight=10;
    server 127.0.0.1:2024 weight=10;
}

server {
    listen      8088;
    server_name localhost;

    location / {
        root  html;
        index index.html index.htm;
        proxy_pass http://backserver;
    }
}
```

3.3.5 fair 最小响应时间算法

```
upstream backserver {
    server 127.0.0.1:2021 weight=50;
    server 127.0.0.1:2022 weight=30;
    server 127.0.0.1:2023 weight=10;
    server 127.0.0.1:2024 weight=10;
    fair;
}

server {
    listen      8088;
    server_name localhost;

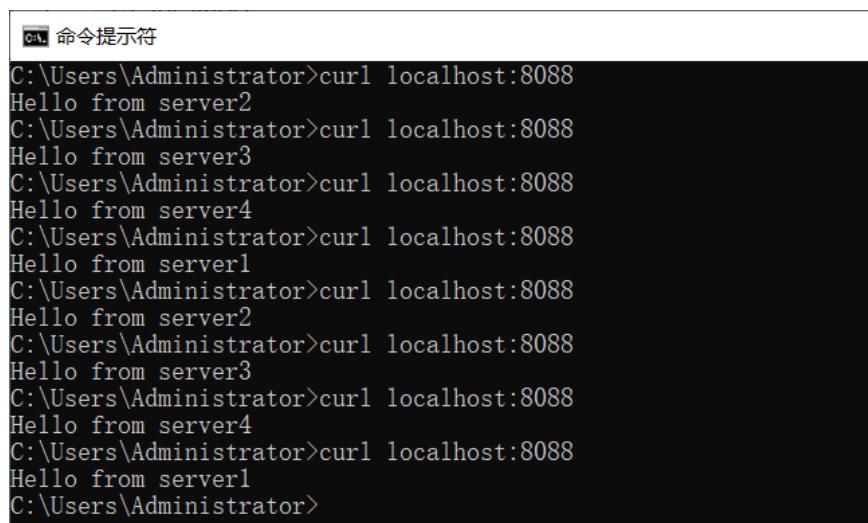
    location / {
        root  html;
        index index.html index.htm;
        proxy_pass http://backserver;
    }
}
```

3.3.6 url_hash

Nginx默认不支持这种调度算法，要使用的话需要安装nginx的hash软件包。若感兴趣可以自行实现。

3.4 实验结果

3.4.1 轮询



命令提示符

```
C:\Users\Administrator>curl localhost:8088
Hello from server2
C:\Users\Administrator>curl localhost:8088
Hello from server3
C:\Users\Administrator>curl localhost:8088
Hello from server4
C:\Users\Administrator>curl localhost:8088
Hello from server1
C:\Users\Administrator>curl localhost:8088
Hello from server2
C:\Users\Administrator>curl localhost:8088
Hello from server3
C:\Users\Administrator>curl localhost:8088
Hello from server4
C:\Users\Administrator>curl localhost:8088
Hello from server1
C:\Users\Administrator>
```

可见其顺序即为12341234...，验证了轮询方法的功能。

3.4.2 指定权重

```
命令提示符
Microsoft Windows [版本 10.0.19042.1288]
(c) Microsoft Corporation。保留所有权利。

C:\Users\Administrator>curl localhost:8088
Hello from server1
C:\Users\Administrator>curl localhost:8088
Hello from server2
C:\Users\Administrator>curl localhost:8088
Hello from server1
C:\Users\Administrator>curl localhost:8088
Hello from server3
C:\Users\Administrator>curl localhost:8088
Hello from server1
C:\Users\Administrator>curl localhost:8088
Hello from server2
C:\Users\Administrator>curl localhost:8088
Hello from server4
C:\Users\Administrator>curl localhost:8088
Hello from server1
C:\Users\Administrator>curl localhost:8088
Hello from server2
C:\Users\Administrator>curl localhost:8088
Hello from server1
C:\Users\Administrator>curl localhost:8088
Hello from server2
C:\Users\Administrator>curl localhost:8088
Hello from server1
C:\Users\Administrator>
```

指定轮询几率， weight和访问比率成正比，由上图可见访问比率按照 5: 3: 1: 1 的比率。指定权重方法适用于后端服务器性能不均的情况。

3.4.3 IP哈希

```
选择命令提示符
Microsoft Windows [版本 10.0.19042.1288]
(c) Microsoft Corporation。保留所有权利。

C:\Users\Administrator>curl localhost:8088
Hello from server1
C:\Users\Administrator>
```

nginx使用请求客户端的ip地址进行哈希计算，确保使用同一个服务器响应请求。这里我们都用本台电脑的IP，因此总是服务器1返回响应。

3.4.4 最小TCP链接数

```
选择命令提示符
Microsoft Windows [版本 10.0.19042.1288]
(c) Microsoft Corporation。保留所有权利。

C:\Users\Administrator>curl localhost:8088
Hello from server1
C:\Users\Administrator>curl localhost:8088
Hello from server1
C:\Users\Administrator>curl localhost:8088
Hello from server1
C:\Users\Administrator>
```

这种策略选取最小TCP链接数，由于本实验只有单机进行，无法很好地体现效果。如图是因此4个服务器的均为最小TCP链接，因此按序为服务器1返回响应。

3.4.5 fair 最小响应时间算法

按后端服务器的响应时间来分配请求，响应时间短的优先分配。这里由于单机实验的缺陷，仍然为服务器1返回响应，因此不贴图。