电影售票系统 性能测试报告

14331156 廖溢机

测试环境说明

以下实验均使用 ab（ApacheBench）进行测试。

主要参考指标：

- 负载（单位时间处理请求数）：Requests per second（越大越优）
- 吞吐量（单位时间流量）：Transfer rate（越大越优）
- 平均请求等待时间：Times per request（越小越优）

测试命令：ab -n 总请求数 -c 并发用户数 要测试的 URL

总请求数增大时会导致总耗时增大；并发用户数增大时会引起阻塞，导致平均请求等待时间增大和吞吐量减小。

测试结果

一、机器配置对性能的影响

测试命令：ab -n **10000** -c **100** http://localhost:8080/

使用双核 CPU、8G 内存机器测试：

```
Concurrency Level:      100
Time taken for tests:   14.424 seconds
Complete requests:      10000
Failed requests:        0
Total transferred:      167820000 bytes
HTML transferred:       166300000 bytes
Requests per second:    693.30 [#/sec] (mean)
Time per request:       144.238 [ms] (mean)
Time per request:       1.442 [ms] (mean, across all concurrent requests)
Transfer rate:          11362.22 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    1   1.5      0      87
Processing:    30  143  34.8    140     369
Waiting:       13  132  33.9    132     368
Total:         30  143  34.8    140     370

Percentage of the requests served within a certain time (ms)
  50%    140
  66%    147
  75%    154
  80%    162
  90%    184
  95%    206
  98%    232
  99%    249
 100%    370 (longest request)
```

使用四核 CPU、16G 内存机器测试：

```
Concurrency Level:      100
Time taken for tests:   7.111 seconds
Complete requests:      10000
Failed requests:        0
Total transferred:      167820000 bytes
HTML transferred:       166300000 bytes
Requests per second:    1406.19 [#/sec] (mean)
Time per request:       71.114 [ms] (mean)
Time per request:       0.711 [ms] (mean, across all concurrent requests)
Transfer rate:          23045.61 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.6      0      24
Processing:    15   70  18.2     68     160
Waiting:        7   64  15.5     64     147
Total:         15   71  18.2     69     160

Percentage of the requests served within a certain time (ms)
  50%     69
  66%     72
  75%     76
  80%     79
  90%     95
  95%    110
  98%    121
  99%    125
 100%    160 (longest request)
```

列表对比如下：

指标	双核 CPU、8G 内存	四核 CPU、16G 内存
Requests per second（个/s）	693.3	1406.19
Transfer rate（Kb/s）	11362.22	23045.61
Times per request（ms）	144.238	71.114

可见四核机器比双核机器性能有几乎一倍的提升。故机器配置越高，性能也越高。

二、缓存后的性能

测试命令：ab -n **1000** -c **1000** http://localhost:8080/selectShow?movie_id=1

测试机器：四核 CPU、16G 内存

缓存前：

```
Concurrency Level:      1000
Time taken for tests:   1.131 seconds
Complete requests:      1000
Failed requests:        0
Total transferred:      4801000 bytes
HTML transferred:       4649000 bytes
Requests per second:    884.12 [#/sec] (mean)
Time per request:       1131.065 [ms] (mean)
Time per request:       1.131 [ms] (mean, across all concurrent requests)
Transfer rate:          4145.19 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.4      0       6
Processing:   182  609 232.1    669     939
Waiting:      175  607 233.0    669     939
Total:        183  609 232.1    669     939

Percentage of the requests served within a certain time (ms)
  50%    669
  66%    784
  75%    833
  80%    848
  90%    870
  95%    896
  98%    917
  99%    928
 100%    939 (longest request)
```

缓存后：

```
Concurrency Level:      1000
Time taken for tests:   0.646 seconds
Complete requests:      1000
Failed requests:        0
Total transferred:      4801000 bytes
HTML transferred:       4649000 bytes
Requests per second:    1547.90 [#/sec] (mean)
Time per request:       646.037 [ms] (mean)
Time per request:       0.646 [ms] (mean, across all concurrent requests)
Transfer rate:          7257.29 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.4      0       2
Processing:   217  306  57.6    285     424
Waiting:      196  301  62.0    280     423
Total:        217  307  57.6    286     424

Percentage of the requests served within a certain time (ms)
  50%    286
  66%    338
  75%    356
  80%    366
  90%    392
  95%    404
  98%    414
  99%    419
 100%    424 (longest request)
```

列表对比如下：

指标	缓存前	缓存后

Requests per second（个/s）	884.12	1547.9
Transfer rate（Kb/s）	4145.19	7257.29
Times per request（ms）	1131.065	646.037

可见缓存后性能大幅提升。

三、服务器数量对性能的影响

负载均衡（Load Balance）是分布式系统架构设计中必须考虑的因素之一，它通常是指将请求均匀分摊到多个服务器上执行，从而提高效率。

使用 Nginx 做负载均衡，配置如下：

```
worker_processes  8;

events {
    worker_connections  1024;
}

http {
    include       mime.types;
    default_type  application/octet-stream;
    sendfile        on;
    keepalive_timeout  65;

    #负载均衡配置
    #这里根据需要修改为实际对应的 IP 和端口号
    upstream backend {
        server 127.0.0.1:8081;
        server 127.0.0.1:8082;
    }

    #反向代理配置
    server {
        listen 8080;
        server_name  localhost;

        location / {
            #反向代理的地址，这里填的是 upstream 的模块名，即反向代理到多个不同的服务器，从而实现负载均衡
```

```
            proxy_pass  http://backend;
        }
    }
}
```

实验使用两台配置接近的电脑做负载均衡：

测试命令：ab -n **10000** -c **1000** http://localhost:8080/selectShow?movie_id=1

列表对比如下：

指标	无负载均衡	负载均衡
Requests per second（个/s）	1006.23	1891.25
Transfer rate（Kb/s）	14365.72	26161.85
Times per request（ms）	1071.595	524.43

可见服务器数量增加后，性能也几乎成倍提升。

四、页面动静分离后的性能

动静分离是让动态网站里的动态网页根据一定规则把不变的资源和经常变的资源区分开来，动静资源做好了拆分以后，我们就可以根据静态资源的特点将其做缓存操作，从而提高速度。

动静分离+负载均衡的实现：使用 Nginx 服务器缓存静态资源，静态资源直接从 Nginx 服务器本地获取，动态资源则通过 Nginx 反向代理到动态服务器组并做负载均衡。

Nginx 配置如下：

```
# 动静分离配置
# 注意此配置仅限在 Windows 下使用，Linux 下需要修改路径

#Nginx 进程数，建议设置为等于 CPU 总核心数
worker_processes 8;

#开启全局错误日志类型
error_log logs/error.log;
```

```
#进程文件
pid logs/nginx.pid;

#一个 Nginx 进程打开的最多文件描述数目 建议与 ulimit -n 一致
#如果面对高并发时 注意修改该值 ulimit -n 还有部分系统参数 而并非这个单独确定
worker_rlimit_nofile 65535;

events {
    #单个进程最大连接数
    worker_connections 65535;
}

http{
    #扩展名与文件类型映射表
    include mime.types;

    #默认类型
    default_type application/octet-stream;

    #日志
    log_format    main    '$remote_addr  -  $remote_user  [$time_local]
"$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';

    access_log  logs/access.log  main;

    sendfile on;
    tcp_nopush on;
    tcp_nodelay on;
    keepalive_timeout 65;
    types_hash_max_size 2048;

    #gzip 压缩传输
    gzip on;
    gzip_min_length 1k;
    gzip_buffers 16 64k;
    gzip_http_version 1.1;
    gzip_comp_level 2;
    gzip_types     text/plain     application/x-javascript     text/css
application/xml application/javascript;
    gzip_vary on;

    #动态服务器组
```

```
    #这里根据需要修改为实际对应的 IP 和端口号
    upstream dynamic_backend {
        server 127.0.0.1:8081;
        server 127.0.0.1:8082;
    }

    #配置代理参数
    client_max_body_size 10m;          #允许客户端请求的最大单文件字节数
    client_body_buffer_size 1024k;     #缓冲区代理缓冲用户端请求的最大字节数
    proxy_connect_timeout 90;          #nginx 跟后端服务器连接超时时间(代理连
接超时)
    proxy_read_timeout 90;             #连接成功后，后端服务器响应时间(代理接
收超时)
    proxy_buffer_size 16k;             #设置代理服务器（nginx）保存用户头信息
的缓冲区大小
    proxy_buffers 4 1024k;             #proxy_buffers 缓冲区，网页平均在 32k
以下的话，这样设置
    proxy_busy_buffers_size 2048k;     #高负荷下缓冲大小（proxy_buffers*2）

    server {
        listen 8080;
        server_name localhost;

        #匹配 selectShow 页面静态资源
        location ^~ /selectShow/css/ {
            alias static/css/;
        }
        location ^~ /selectShow/js/ {
            alias static/js/;
        }
        location ^~ /selectShow/images/ {
            alias static/images/;
        }
        location ^~ /selectShow/fonts/ {
            alias static/fonts/;
        }

        #匹配首页静态资源
        location ^~  /css/ {
            alias static/css/;
        }
        location ^~ /js/ {
            alias static/js/;
        }
```

```
        location ^~ /images/ {
            alias static/images/;
        }
        location ^~ /fonts/ {
            alias static/fonts/;
        }

        #匹配所有静态资源，静态资源直接从 nginx 服务器的硬盘读取，不经过动态服务
器
        location
~ .*\.(js|css|ico|png|jpg|jpeg|gif|bmp|otf|eot|ttf|woff)$ {
            root static;
            expires 30d;  #缓存 30 天
        }

        #动态资源反向代理到动态服务器
        location ~ .*$ {
            proxy_pass http://dynamic_backend;

            proxy_redirect off;
            proxy_set_header Host $host:8080;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        }
    }
}
```

测试命令：ab -n **10000** -c **1000** http://localhost:8080/selectShow?movie_id=1

列表并与之前对比如下：

指标	无负载均衡	负载均衡	动静分离+负载均衡
Requests per second（个/s）	1006.23	1891.25	1976.33
Transfer rate（Kb/s）	14365.72	26161.85	27358.46
Times per request（ms）	1071.595	524.43	481. 57

可见添加动静分离后，性能比之前有所提高。