目录：

1.优化nginx

1-1.安装nginx

下载nginx

使用如下命令下载nginx

```
# wget  http://nginx.org/download/nginx-1.8.1.tar.gz
```

.

编译使用的选项：

| 参数 | 安装选项 | 备注 |
| --- | --- | --- |
| --prefix=path | Install | 设置编译安装的路径 |
| --sbin-path=path | Install | 设置编译安装的sbin路径 |
| --conf-path=path | Install | 设置编译安装的配置文件的路径 |
| --pid-path=path | Install | 设置nginx的PID文件的路径 |
| --error-log-path=path | Install | 设置错误日志文件的路径 |
| --http-log-path=path | Install | 设置http日志文件的路径 |
| --user=name | Install | 设置运行nginx的用户 |
| --group=name | Install | 设置运行nginx的组 |
| --with-pcre=path | Install | 设置要加载的PCRE的库文件路径 |
| --with-zlib=path | Install | 设置要加载的ZLIB库文件的路径 |
| --with-cc-opt=parameters | Install | |
| --with-ld-opt=parameters | Install | |
| --with-select_module | uninstall | |
| --without-select_module | uninstall | |
| --with-poll_module | uninstall | |
| --without-poll_module | uninstall | |
| --without-http_gzip_module | uninstall | |
| --without-http_rewrite_module | uninstall | |
| --without-http_proxy_module | uninstall | |
| --with-http_ssl_module | | |

使用如下命令编译Nginx

```
#  ./configure --user=nginx --group=nginx  --prefix=/usr/local/nginx --with-http_stub_status_module –with-http_ssl_module   \

--add-module=/software/module/ngx_http_redis-0.3.2   \          ---------->Cache优化

--with-ld-opt="-ljemalloc"   \                                  ---------->Nginx内存管理的优化

 --with-pcre=/opt/pcre-8.34
```

使用如下命令安装nginx

```
# make && make install
```

1-2.优化配置

目前配置：

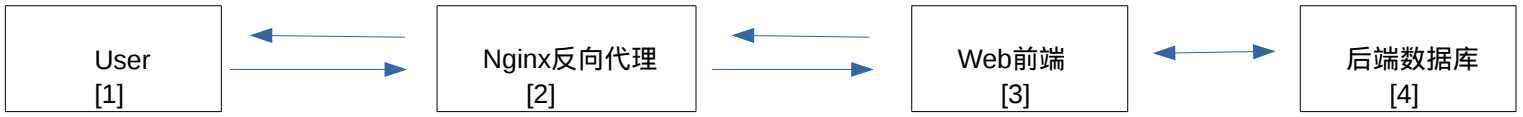
```
user www;
worker_processes 4;
error_log  /usr/local/nginx/logs/nginx_error.log;
#pid        /usr/local/nginx/nginx.pid;
#Specifies the value for maximum file descriptors that can be opened by this process.
worker_rlimit_nofile 65535;
events
{
  use epoll;
  worker_connections 65535;
}
```

```
http
{
  include       mime.types;
  default_type  application/octet-stream;
  #charset  gb2312;
  server_names_hash_bucket_size 128;
  client_header_buffer_size 32k;
  large_client_header_buffers 4 32k;
  client_max_body_size 8m;
  sendfile on;
  tcp_nopush     on;
  keepalive_timeout 60; #60
  server_tokens off;
  tcp_nodelay on;
  fastcgi_connect_timeout 300;  #300
  fastcgi_send_timeout 300;     #300
  fastcgi_read_timeout 300;     #300
  fastcgi_buffer_size 64k;
  fastcgi_buffers 4 64k;
  fastcgi_busy_buffers_size 128k;
  fastcgi_temp_file_write_size 128k;
  proxy_connect_timeout 300;
  proxy_read_timeout 300;
  proxy_send_timeout 300;
  gzip on;
  gzip_min_length  1k;
  gzip_buffers     4 16k;
  gzip_http_version 1.0;
  gzip_comp_level 2;
  gzip_types       text/plain application/x-javascript text/css application/xml;
  gzip_vary on;
    log_format access '$time_local # $connection # $connection_requests # $request_time # '
                      '$remote_addr # $remote_user # $server_name # "$request" # '
                      '$status {} $body_bytes_sent {} $bytes_sent {} $sent_http_content_length {} '
                      '"$http_referer" {} "$http_user_agent" {} "$http_x_forwarded_for" {} '
                      '$cookie_vk {} $http_via ';
```

优化后的配置：

## 1-3.Cache配置的优化

### 1-3-1.目前线上环境Nginx的Cache架构



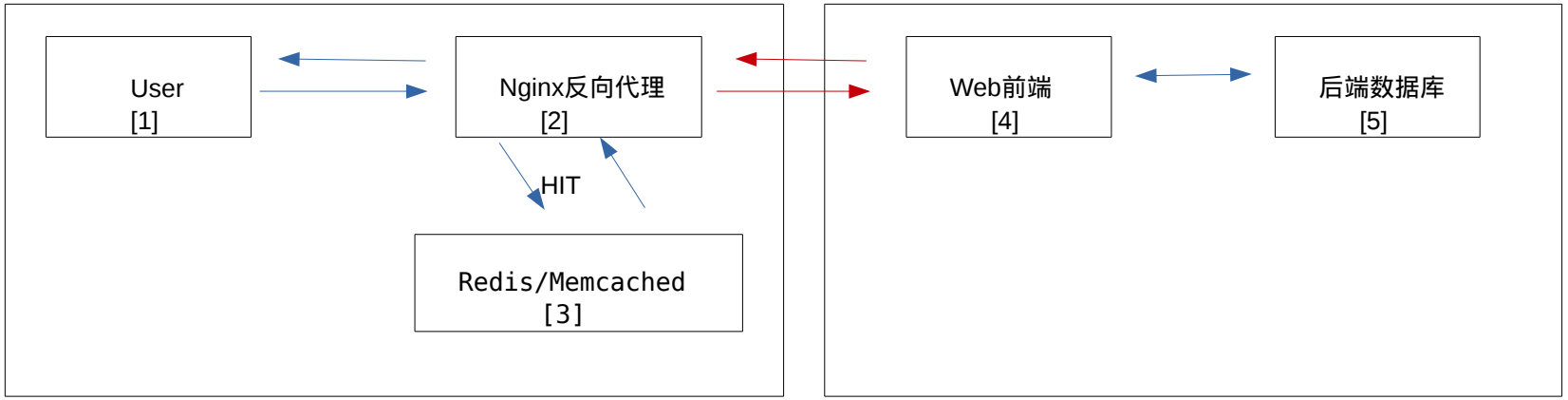

[1]:用户使用浏览器访问网站。

[2]:Nginx反向代理去WEB前端拉取用户需要现实的资源，包括图片，CSS，JS等所有的一切。当用户下一次访问的时候，一些图片信息已经缓存到用户本地的电脑缓存目录了。当有新用户再次访问网站的时候，会再一次执行一次这个过程。

[3]:提供WEB服务和WEB程序运行的平台。

[4]:提供数据查询和管理服务。

### 1-3-2.优化后环境Nginx的Cache架构



[1]:用户使用浏览器访问网站。

[2]:Nginx反向代理去WEB前端拉取用户需要现实的资源，包括图片，CSS，JS等所有的一切。当用户下一次访问的时候，一些图片信息已经缓存到用户本地的电脑缓存目录了。当有新用户再次访问网站的时候，会去Nginx反向代理服务器上的Redis/Memcached上去查询是否有已经缓存好的文件；如果有已经缓存的文件，就直接调用缓存中的文件。如果没有，就回去后面的WEB前端去拉取图片文件。

[3]:在nginx反向代理服务器上缓存文件

[4]:提供WEB服务和WEB程序运行的平台。

[5]:提供数据查询和管理服务。

1-3-3.缓存插件的性能的优缺点分析

| 对比项目 | Nginx本身Cache | RedisCache | Memcached Cache |
|---|---|---|---|
| 架构 | 无概念 | 客户端-服务器 | 客户端-服务器 |
| 事务 | 无概念 | 支持 | 不存在对事务的支持 |
| 数据备份 | 无概念 | Slave对数据进行备份 | |
| 有效性 | 很长 | 支持/可选 | 不保证存储的数据的有效性 |
| 持久化 | 很长 | 支持/可选 | 不做数据的持久化工作 |
| 性能 | 一般 | 较高 | 高 |
| 集群 | 高 | 官方和插件都支持 | 插件支持 |
| MR类批处理应用 | 无概念 | 支持 | 不支持 |
| 网络IO模型 | 一般 | 单线程的IO复用模型 | 多线程的非阻塞IO复用的网络模型 |
| 内存管理方面 | 一般 | 现申请内存 | 预分配的内存池 |
| 数据一致性问题 | 无概念 | 事物支持 | 支持 |
| 不同语言的客户端支持 | 无概念 | 多 | 比较多 |
| 数据结构 | 无概念 | 比较多 | 多 |

注：Redis使用最佳方式是全部数据in-memory，Redis采用hash结构来做key-value存储，由于其组合式的压缩，其内存利用率会高于Memcached，网络IO的次数和数据体积性能比较，我们决定使用Redis做为Cache。官方的资料：http://wiki.nginx.org/HttpRedis2Module

```
    include sites/*;

    upstream hqbjlbackground  {
            server  10.0.0.15:8080;
           }

    upstream bjlbg  {
            server  10.0.0.15:8080;
           }
    upstream hqbjlms {
            server 10.0.0.15:8180;
           }

}
```