**PHP缓存技术学习笔记**

**操作码介绍及缓存原理**

当客户端请求php程序时，php引擎会解析该php程序，并将其编译为特定的操作码(Opcode),

默认情况下编译好的操作码由php引擎执行后丢弃

而操作码缓存（Opcode） 的原理就是将编译后的操作码保存起来，

在下一次代用该php页面时重用它，避免相同代码重复编译

节省php引擎重复编译时间，降低服务器负载，同时减少CPU和内存的开销

**LNMP环境中**

1、 Nginx接收客户端的PHP程序访问请求。

2、Nginx根据扩展名等过滤规则将PHP程序请求传递给解析PHP的FCGI进程(PHP-FPM)

3、PHP FPM进程调用PHP解析器读取站点磁盘上的PHP文件，并加载到内存中。

4、PHP解析器将PHP程序编译成为opcode文件，然后把opcode缓存起来。

5、PHP FPM引擎执行opcode树，返回数据给Nginx，进而返回客户端。

6、Nginx接收客户新的PHP程序请求，PHP-FPM引擎就会直接读取缓存中的opcode 执行，将结果返回，

该过程中无需第4步操作，从而提升了 PHP编译解析效率。

**PHP缓存器加速软件和选择建议**

1、xcache

经过测试，效率更高，速度更快，社区活跃，支持高版本 php5.5 php5.6

2、ZendOpcache

官方研发，从php5.5版本已经整合到php软件里，可以通过—enable-opache参数来安装

旧版本可以通过插件的方式安装使用

3、eAccelerator 效果还可以，仅适合php5.4以下的版本

4、APC(Alternative PHP Ccche)

**生产环境插件的安装和建议**

对于功能性插件，如果业务产品不需要使用，可以暂时不考虑安装，

例如： PDO\_MYSQL\memcache\imagick等.如果不清楚是否要，最好还是装上，有备无患

对于性能优化插件，eaccelerator、xcache、ZendOpcache、APC可以安装任一种，

具 体情况看实际业务需求，在选择时最好能搭建相关环境进行压力测试，

然后根据实际测试 结果来选择，用数据说话很重要。

**常见的PHP扩展插件列表**

名称 类型 说明 状态 操作  
PATH\_INFO 扩展配置 MVC架构的程序需要开启,如typecho  
ZendGuardLoader 脚本解密 用于解密ZendGuard加密脚本  
ionCube 脚本解密 用于解密ionCube Encoder加密脚本  
fileinfo 通用扩展 用于获取文件MIME,编码等信息  
opcache 缓存器 用于加速PHP脚本  
xcache 缓存器 支持脚本缓存和变量缓存  
memcache 缓存器 强大的内容缓存器  
memcached 缓存器 比memcache支持更多高级功能  
redis 缓存器 基于内存亦可持久化的Key-Value数据库  
apcu 缓存器 脚本缓存器 安装  
imagemagick 通用扩展 Imagick高性能图形库 安装

xdebug 调试器 不多说,不了解的不要安装  
imap 邮件服务 邮件服务器必备  
exif 通用扩展 用于读取图片EXIF信息  
intl 通用扩展 提供国际化支持  
xsl 通用扩展 xsl解析扩展   
Swoole 通用扩展 异步、并行、高性能网络通信引擎  
yaf 框架 Yaf是一个C语言编写的PHP框架  
phalcon 框架 phalcon 是一个C语言编写的PHP框架



**安装缓存前操作**

**echo 'export LC\_ALL=C' >>/etc/profile**

source /etc/profile

[root@web01 /server/tools]# echo $LC\_ALL

C

**yum install -y autoconf**

**yum -y install perl-devel**

[root@web01 /server/tools/eaccelerator-0.9.6.1]# rpm -qa perl-devel

perl-devel-5.16.3-293.el7.x86\_64

**安装eAccelerator**

作用：（仅仅适合php5.4以及以下版本）

缓存php程序编译后的中间代码文件(opcode),session数据等，

降低php程序在编译解析时对服务器的性能开销。

还可以加快php执行速度，使php代码程序执行效率提高1-10倍。

eAccelerator会把编译好的php程序存放到共享内存里，每次从内存调用执行，

还可以设定把一些不适合的内处理缓存的编译结果，存储到磁盘上，

默认情况下，磁盘和内存缓存都会被eAccelerator使用。

eaccelerator-0.9.6.1.tar.bz2 最新版本

**安装步骤：**

wget 10.10.10.100/src/eaccelerator-0.9.6.1.tar.bz2

tar xf eaccelerator-0.9.6.1.tar.bz2

cd eaccelerator-0.9.6.1

/server/application/php53/bin/phpize

./configure --enable-eaccelerator=shared --with-php-config=/server/application/php53/bin/php-config

make

make install

生产so文件即为安装成功

**eaccelerator 配置与优化**

mkdir -p /tmp/eaccelerator

chown -R www:www /tmp/eaccelerator/

php.ini修改：

[eaccelerator]

extension=eaccelerator.so #加载加速模块，路径相对于extension\_dir

eaccelerator.cache\_dir="/tmp/eaccelerator" #缓存目录

eaccelerator.shm\_size="64" #缓存数据的共享内存大小，如果为0则看内核配置/proc/sys/kernel/shmmax

eaccelerator.enable="1" #激活是否缓存

eaccelerator.optimizer="1" #激活是否优化

eaccelerator.check\_mtime="1" #激活是否检查修改时间

eaccelerator.debug="0" #激活是否需要debug调试

eaccelerator.filter="" #需要过滤的

eaccelerator.shm\_max="0" #缓存的最长时间0 表示不限制

eaccelerator.shm\_ttl="3600" #对象在缓存里的生存周期

eaccelerator.shm\_prune\_period="3600" #当内存不够时候，删除之前的缓存文件

eaccelerator.shm\_only="0" #是否缓存数据到磁盘0表示准许，对于sessions和content没影响

eaccelerator.compress="1" #允许或禁止压缩内容缓存。默认值为 “1”，为允许压缩。

eaccelerator.compress\_level="9" #指定内容缓存的压缩等级。默认值为 “9”，为最高等级。

eaccelerator.keys = "disk\_only"

eaccelerator.sessions = "disk\_only"

eaccelerator.content = "disk\_only"

eaccelerator.shm\_prune\_period=”3600”   
当 eAccelerator 获取新脚本的共享内存大小失败时，他将试图从共享内存中删除早于”shm\_prune\_period” 秒的缓存脚本。默认值为 “0”，为不从共享内存中删除任何缓存文件。

eaccelerator.shm\_only=”0”

允许或禁止将已编译脚本缓存在磁盘上。该选项对 session 数据和内容缓存无效。默认值为 “0”，为使用磁盘和共享内存进行缓存。

eaccelerator.keys = “disk\_only”

eaccelerator.sessions = “disk\_only”

eaccelerator.content = “disk\_only”

设置内容缓存的存放的地方，可以设置为：

shm\_and\_disk 在共享缓存和硬盘(默认值)

shm 默认存在共享内存，如果共享内存已满或大小超过 “eaccelerator.shm\_max” 的值，就存到硬盘

shm\_only 只存放在共享内存

disk\_only 只存放在硬盘

none 不缓存数据

使用tmpfs优化eaccelerator

tmpfs是一种基于内存的文件系统、通常使用tmpfs作为数据临时存储，比本地磁盘快很多

此方法适合各类场景，比如很多软件会在/tmp下临时缓冲切图，存放session数据，

则可以让tmp使用tmpfs文件系统来加快访问效率，

下列将/tmp/eaccelerator 挂载到tmpfs 让访问缓冲数据更快

size = 16 根据自己的物理内存大小调整

[root@web01 /tmp/eaccelerator]# mount -t tmpfs -o size=16m tmpfs /tmp/eaccelerator/

[root@web01 /tmp/eaccelerator]# df -h

Filesystem Size Used Avail Use% Mounted on

/dev/mapper/cl-root 8.0G 3.0G 5.1G 37% /

devtmpfs 902M 0 902M 0% /dev

tmpfs 912M 0 912M 0% /dev/shm

tmpfs 912M 8.7M 904M 1% /run

tmpfs 912M 0 912M 0% /sys/fs/cgroup

/dev/sda1 1014M 139M 876M 14% /boot

tmpfs 183M 0 183M 0% /run/user/0

tmpfs 16M 0 16M 0% /tmp/eaccelerator

[root@web01 /tmp/eaccelerator]# tail -1 /etc/fstab

tmpfs /tmp/eaccelerator tmpfs size=16 0 0

**完成版配置文件**

extension\_dir = /server/application/php-5.3.29/lib/php/extensions/no-debug-non-zts-20090626/

;extension = eaccelerator.so

;extension = opcache.so

;extension = xcache.so

[eaccelerator]

extension=eaccelerator.so

eaccelerator.cache\_dir="/tmp/eaccelerator"

eaccelerator.shm\_size="64"

eaccelerator.enable="1"

eaccelerator.optimizer="1"

eaccelerator.check\_mtime="1"

eaccelerator.debug="0"

eaccelerator.filter=""

eaccelerator.shm\_max="0"

eaccelerator.shm\_ttl="3600"

eaccelerator.shm\_prune\_period="3600"

eaccelerator.shm\_only="0"

eaccelerator.compress="1"

eaccelerator.compress\_level="9"

eaccelerator.keys = "disk\_only"

eaccelerator.sessions = "disk\_only"

eaccelerator.content = "disk\_only"

**xcache-----缓存软件**

安装xcache-3.2.0 稳定版

wget <http://10.10.10.100/src/xcache-3.2.0.tar.gz>

tar zxvf xcache-3.2.0.tar.gz

cd xcache-3.2.0

/server/application/php53/bin/phpize

./configure --enable-xcache --with-php-config=/server/application/php53/bin/php-config

make

make install

**安装安装xcache-3.2.0后的使用与优化**

cat /server/tools/xcache-3.2.0/xcache.ini >>/server/application/php54/etc/php.ini

[xcache-common]

extension = xcache.so

[xcache.admin]

xcache.admin.enable\_auth = On

xcache.admin.user = "mOo"

xcache.admin.pass = "md5 encrypted password"

[xcache]

xcache.shm\_scheme = "mmap"

**xcache.size = 60M**

**xcache.count = 1**

xcache.slots = 8K

**xcache.ttl = 0**

**xcache.gc\_interval = 0**

**xcache.var\_size = 4M**

xcache.var\_count = 1

xcache.var\_slots = 8K

xcache.var\_ttl = 0

xcache.var\_maxttl = 0

xcache.var\_gc\_interval = 300

xcache.var\_namespace\_mode = 0

xcache.var\_namespace = ""

xcache.readonly\_protection = Off

xcache.mmap\_path = "**/dev/zero**"

xcache.coredump\_directory = ""

xcache.coredump\_type = 0

xcache.disable\_on\_crash = Off

xcache.experimental = Off

xcache.cacher = On

xcache.stat = On

xcache.optimizer = Off

[xcache.coverager]

xcache.coverager = Off

xcache.coverager\_autostart = On

xcache.coveragedump\_directory = ""

需要调整的参数：

|  |  |
| --- | --- |
| **xcache.size = 60M** | **xcache.size = 256M** |
| **xcache.count = 1** | **xcache.count = 2(cpu核数)** |
| xcache.slots = 8K | **xcache.slots = 8K** |
| **xcache.ttl = 0** | **xcache.ttl = 86400** |
| **xcache.gc\_interval = 0** | **xcache.gc\_interval = 3600** |
| **xcache.var\_size = 4M** | **xcache.var\_size = 64M** |
| xcache.mmap\_path = "**/dev/zero**" | **xcache.mmap\_path = "/tmp/xcache54"** |

关于下面两个参数说明：

**xcache.readonly\_protection = Off**

如果启用该参数改为On 性能微降，但是会提高一定的安全系数，

这个选项对于xcache.mmap\_path = "**/dev/zero**" 无效

**xcache.mmap\_path = "/dev/zero"**

对于xcache.mmap\_path是一个文件路径而并非目录，如果需要启用该参数，使用/tmp/xcache54这样的路径

而不是/dev/zero，如果开启了xcache.readonly\_protection 不通进程的php将不会共享同一个/tmp/xache

echo -n "111111"|md5sum

vim php.ini

[xcache.admin]

xcache.admin.enable\_auth = On

xcache.admin.user = "linyaohong"

xcache.admin.pass = "96e79218965eb72c92a549dd5a330112"

复制web界面网站文件到站点根目录

cp -a /server/tools/xcache-3.2.0/htdocs/ /server/wwwroot/linyaohong.com/xadmin

chown -R www:www cd /server/wwwroot/linyaohong.com/xadmin

pkill php-fpm

/server/application/php54/sbin/php-fpm



**完整版配置文件：**

extension\_dir = /server/application/php-5.4.45/lib/php/extensions/no-debug-non-zts-20100525/

[xcache-common]

extension = xcache.so

[xcache.admin]

xcache.admin.enable\_auth = On

xcache.admin.user = "linyaohong"

xcache.admin.pass = "96e79218965eb72c92a549dd5a330112"

[xcache]

xcache.shm\_scheme = "mmap"

xcache.size = 256M

xcache.count = 2

xcache.slots = 8K

xcache.ttl = 86400

xcache.gc\_interval = 3600

xcache.var\_size = 64M

xcache.var\_count = 1

xcache.var\_slots = 8K

xcache.var\_ttl = 0

xcache.var\_maxttl = 0

xcache.var\_gc\_interval = 300

xcache.var\_namespace\_mode = 0

xcache.var\_namespace = ""

xcache.readonly\_protection = Off

xcache.mmap\_path = "/dev/zero"

xcache.coredump\_directory = ""

xcache.coredump\_type = 0

xcache.disable\_on\_crash = Off

xcache.experimental = Off

xcache.cacher = On

xcache.stat = On

xcache.optimizer = Off

[xcache.coverager]

xcache.coverager = Off

xcache.coverager\_autostart = On

xcache.coveragedump\_directory = ""

**安装****ZendOpcache-----缓存软件**

安装ZendOpcache

wget <http://10.10.10.100/src/zendopcache-7.0.5.tgz>

tar zxvf zendopcache-7.0.5.tgz

cd zendopcache-7.0.5

/server/application/php53/bin/phpize

./configure --enable-opcache --with-php-config=/server/application/php53/bin/php-config

make

make install

**配置说明：**

opcache.enable = 1 #启用

opcache.memory\_consumption=128 Opcache共享内存空间大小，用于存放precompiledPHP code 默认64Mbytes

opcache.interned\_strings\_buffer=8 默认值为4 interned strings内存的数量，单位是 M

opcache.max\_accelerated\_files=4000 默认值为2000， OPcache哈希表的 key的最大数

opcache.revalidate\_freq=60 默认值为2,检査文件时间戳的频率，用于共享内存分配的变化

opcache.fast\_shutdown=1 默认值为0,如果激活，一个快速的关闭队列将被用来加速代码

opcache.enable\_cli=1 默认值为0，激活 PHPCLI的 OPcache，用于测试和调试

**完整版配置文件：**

;opcache

[Zend Opcache]

zend\_extension= /server/application/php-5.3.29/lib/php/extensions/no-debug-non-zts-20090626/opcache.so

opcache.enable = 1

opcache.memory\_consumption=128

opcache.interned\_strings\_buffer=8

opcache.max\_accelerated\_files=4000

opcache.revalidate\_freq=60

opcache.fast\_shutdown=1

opcache.enable\_cli=1

**不支持extension = opcache.so; 写法**

**安装memcache客户端**

wget http://10.10.10.100/src/memcache-2.2.7.tgz

tar zxvf memcache-2.2.7.tgz

cd memcache-2.2.7

/server/application/php54/bin/phpize

./configure --enable-memcache --with-php-config=/server/application/php54/bin/php-config

make

make install

**安装ImageMagick 并且php加载imagick-2.3.0.tar.gz插件**

imagick工作需要ImageMagick软件支持

imagick插件是一个可以供PHP调用ImageMagick功能的扩展模块，

使用这个扩展，可以使PHP具备和ImageMagick相同功能

安装ImageMagick

wget <http://10.10.10.100/src/ImageMagick-6.9.10-2.tar.gz>

tar zxvf ImageMagick-6.9.10-2.tar.gz

cd ImageMagick-6.9.10-2

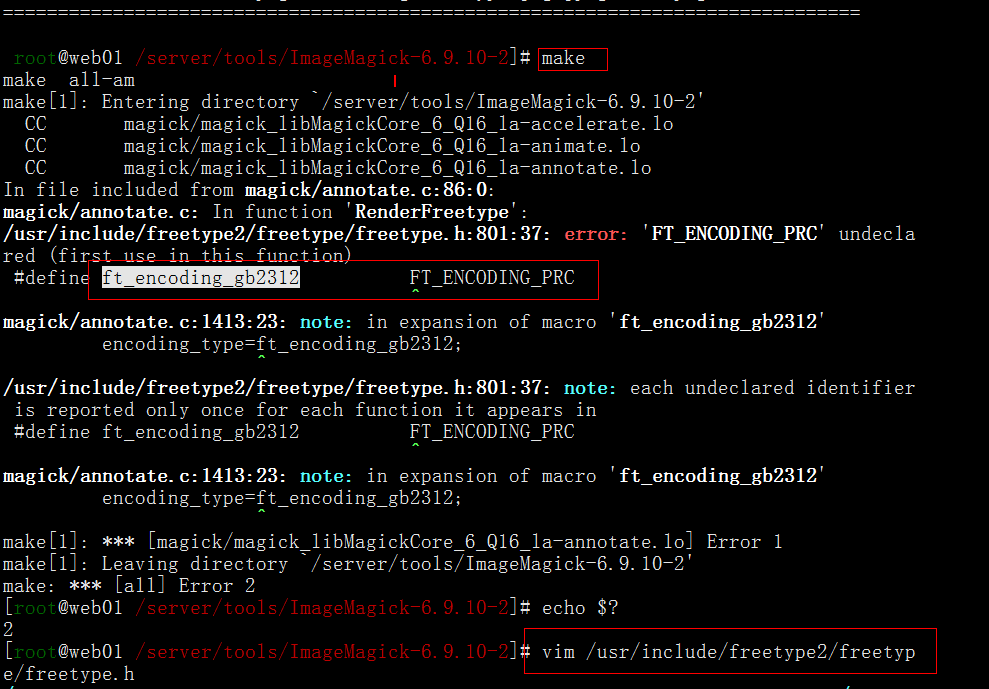
./configure

make

make install

如果make的时候出错如下图，可以根据图中信息修改：

换成大写



安装imagick

wget 10.10.10.100/src/imagick-3.4.3.tgz

tar zxvf imagick-3.4.3.tgz

cd imagick-3.4.3

/server/application/php54/bin/phpize

./configure --with-php-config=/server/application/php54/bin/php-config

**小结：**

总结：

[root@web01 /server/tools/imagick-3.4.3]# ls -l /server/application/php-5.4.45/lib/php/extensions/no-debug-non-zts-20100525/

total 2348

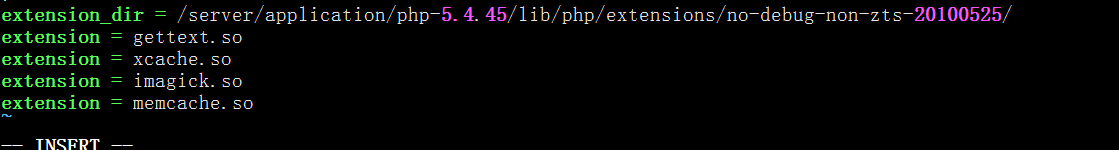
-rwxr-xr-x 1 root root 50168 Jan 18 16:09 gettext.so

-rwxr-xr-x 1 root root 1351688 Jan 19 00:40 imagick.so

-rwxr-xr-x 1 root root 290296 Jan 18 23:47 memcache.so

-rwxr-xr-x 1 root root 702968 Jan 18 23:29 xcache.so

修改php.ini



**安装PDO\_MYSQL扩展**

一般情况下安装PHP的时候都要直接带着这个扩展

PDO扩展为php访问数据库定义了一个轻量级一致性的接口，它提供了一个数据访问抽象层

无论使用的是什么数据库，都可以通过一致性的函数执行并且或许数据

--with-pdo-mysql=mysqlnd 参数安装

或者扩展插件安装

# wget http://pecl.php.net/get/PDO\_MYSQL-1.0.2.tgz

然后是解压缩。  
# tar -zxvf PDO\_MYSQL-1.0.2.tgz

进入解压后的目录，然后执行phpize。  
# /usr/local/php/bin/phpize  
Configuring for:  
PHP Api Version: 20100412  
Zend Module Api No: 20100525  
Zend Extension Api No: 220100525

下面生成makefile。后面的两个参数必须要添加，一开始我没有添加，结果分别都提示找不到php-config和mysql的header文件。  
# ./configure -with-php-config=/usr/local/php/bin/php-config -with-pdo-mysql=/usr/local/mysql

在make之前还要做一个mysql的header文件的软连接。因为mysql安装的时候指定了目录，不做软连接的话，还是找不到header文件。  
# ln -s /usr/local/mysql/include/\* /usr/local/include/

然后make和make install。  
# make  
# make install