**简介**

字节码缓存不是php的新特性，有很多独立性的扩展可以实现缓存，比如PHP Cache（APC），eAccelerator，ionCube和XCache等等。但是到目前为止，这些独立的扩展并没有集成到php核心当中。所有在php5.5.0之后，php内置了字节码缓存功能，叫做Zend Opcache。

zend Opcache前身是Zend Optimizer +，在03年改名Opcache，通过opcode缓存和优化提供更快的PHP执行过程。他会将预编译后的php文件存储在共享内存中以供以后的使用，避免从磁盘读取文件在进行解释的重复过程，减少时间和内存的消耗。在php5.5中及之后的版本中自带了zend opcache模块扩展，但是需要使用时需要我们开启和配置。在php5.2- 5.4版本我们也可以使用Opcache，但是需要我们自行安装。

**什么是操作码缓存**

那么什么是字节码缓存呢？php是一门解释型的语言，php解释器执行php脚本时会解析php脚本代码，将php脚本代码编译成一系列可以直接运行的中间代码，也称为操作码（Operate Code，opcode）。然后执行这些操作码．

每次请求php文件都是这样，所以会消耗很多资源，如果每次ＨＴＴＰ请求ＰＨＰ都必须解析，编译和运行脚本，消耗的资源将会更多．

Opcode cache 的目地是避免重复编译，减少 CPU 和内存开销。需要注意的是如果动态内容的性能瓶颈不在于 CPU 和内存，而在于 I/O 操作，比如数据库查询带来的磁盘 I/O 开销，那么 opcode cache 的性能提升是非常有限的．

现代操作码缓存器（Optimizer+，APC2.0+，其他）使用共享内存进行存储，并且可以直接从中执行文件，而不用在执行前“反序列化”代码。这将带来显着的性能加速，通常降低了整体服务器的内存消耗，而且很少有缺点．

**在开启opcache之前，我们先介绍一下编译与解释**  
编译器是把源程序的每一条语句都编译成机器语言,并保存成二进制文件,这样运行时计算机可以直接以机器语言来运行此程序,速度很快;  
而解释器则是只在执行程序时,才一条一条的解释成机器语言给计算机来执行,所以运行速度是不如编译后的程序运行的快的.

解释型语言的实现中，翻译器并不产生目标机器代码，而是产生易于执行的中间代码，这种中间代码与机器代码是不同的，中间代码的解释是由软件支持的，不能直接使用硬件，软件解释器通常会导致执行效率较低。用解释型语言编写的程序是由另一个可以理解中间代码的解释程序执行的。与编译程序不同的是，解释程序的任务是逐一将源程序的语句解释成可执行的机器指令，不需要将源程序翻译成目标代码后再执行。对于解释型Basic语言，需要一个专门的解释器解释执行

**在很多时候我们成为编译，但是它实际是进行解释的**

对于一个编译型程序，它的编译和执行是分开的，先编译成二进制可执行文件，然后在次执行。

对于PHP、Python属于解释型语言，不产生机器码，而是产生中间码（中间码是不能直接执行，这个中间码只有解释器可以识别到，中间码要靠解析器来进行执行）

**比如说PHP的解析器是Zend，PHP使用Zend引擎，中间码我们也称作为操作码（opcode）**

Basic程序，每条语言只有在执行才被翻译。这种解释型语言每执行一次就翻译一次，因而效率低下。

1、编辑：用编辑软件（EDIT.EXE或记事本）形成源程序（.ASM）,如：LX.ASM;  
2、汇编：用汇编程序（MASM.EXE）对源程序进行汇编，形成目标文件（.OBJ），格式如下：MASM LX.ASM;  
3、连接：用连接程序（LINK.EXE）对目标程序进行连接，形成可执行文件（.EXE），格式如下：LINK LX.OBJ;  
4、执行：如果结果在屏幕在显示，则直接执行可执行文件。  
5、调试：用调试程序（DEBUG.EXE）对可执行文件进行调试，格式如下：DEBUG LX.EXE

鸟哥在博客中说，提高PHP 7性能的几个tips，第一条就是开启opache，引用下原文：

记得启用Zend Opcache, 因为PHP7即使不启用Opcache速度也比PHP-5.6启用了Opcache快,

APC与Opcache都是字节码缓存也就是，PHP在被编译的时候，首先会把php代码转换为字节码，字节码然后被执行。

php文件第二次执行时，同样还是会重新转换为字节码，但是很多时候，文件内容几乎是一样的，比如静态HTML文件，生成后内容许久都不会改变，用户访问请求直接由服务器读取响应给客户端浏览器。都不用经过PHP进行解析构建了。

内存中的字节码数据，可以直接缓存进行二次编译。这样程序就会快一些，cpu的消耗也少了。  
（这里字节码 就是 opcode）

**常用配置**

[复制代码](javascript:void(0);)

;开关打开

opcache.enable=1

;开启CLI

opcache.enable\_cli=1

;可用内存, 酌情而定, 单位为：Mb

opcache.memory\_consumption=528

;Zend Optimizer + 暂存池中字符串的占内存总量.(单位:MB)

opcache.interned\_strings\_buffer=8

;对多缓存文件限制, 命中率不到 100% 的话, 可以试着提高这个值

opcache.max\_accelerated\_files=10000

;Opcache 会在一定时间内去检查文件的修改时间, 这里设置检查的时间周期, 默认为 2, 定位为秒

opcache.revalidate\_freq=1

;打开快速关闭, 打开这个在PHP Request Shutdown的时候回收内存的速度会提高

opcache.fast\_shutdown=1

;检查脚本时间戳是否有更新的周期，以秒为单位。设置为 0 会导致针对每个请求， OPcache 都会检查脚本更新。

opcache.revalidate\_freq=0

;如果启用，那么 OPcache 会每隔 opcache.revalidate\_freq 设定的秒数 检查脚本是否更新。

;如果禁用此选项，你必须使用 opcache\_reset() 或者 opcache\_invalidate() 函数来手动重置 OPcache，也可以 通过重启 Web 服务器来使文件系统更改生效。

opcache.validate\_timestamps=0

[复制代码](javascript:void(0);)

注意：如果设置opcache的opcache.validate\_timestamps的指令设成０，那么zend opcache就察觉不到ＰＨＰ脚本的变化，我们必须手动清空zend opcache缓存的字节码，让他发现php脚本的变动．这个设置适合在生产环境中设置成０，在开发环境下最好还是设置成1．

我们可以这样配置，启用自动重新验证缓存功能：

opcache.validate\_timestamps=1

opcache.revalidate\_freq=0

更多的配置指令可以看这里：<http://php.net/manual/zh/opcache.configuration.php>

