**[导读]** 1、全页面静态化缓存也就是将页面全部生成html静态页面，用户访问时直接访问的静态页面，而不会去走php服务器解析的流程。此种方式，在CMS系统中比较常见，比如dedecms；一种比较常用的实现方式是用输出缓存：Ob

**1、全页面静态化缓存**

也就是将页面全部生成html静态页面，用户访问时直接访问的静态页面，而不会去走php服务器解析的流程。此种方式，在CMS系统中比较常见，比如dedecms；

一种比较常用的实现方式是用输出缓存：

Ob\_start()

\*\*\*\*\*\*要运行的代码\*\*\*\*\*\*\*

$content = Ob\_get\_contents();

\*\*\*\*将缓存内容写入html文件\*\*\*\*\*

Ob\_end\_clean();

**2、页面部分缓存**

该种方式，是将一个页面中不经常变的部分进行静态缓存，而经常变化的块不缓存，最后组装在一起显示；可以使用类似于ob\_get\_contents的方式实现，也可以利用类似ESI之类的页面片段缓存策略，使其用来做动态页面中相对静态的片段部分的缓存(ESI技术，请baidu，此处不详讲)。

该种方式可以用于如商城中的商品页；

**3、数据缓存**

顾名思义，就是缓存数据的一种方式；比如，商城中的某个商品信息，当用商品id去请求时，就会得出包括店铺信息、商品信息等数据，此时就可以将这些数据缓存到一个php文件中，文件名包含商品id来建一个唯一标示；下一次有人想查看这个商品时，首先就直接调这个文件里面的信息，而不用再去数据库查询；其实缓存文件中缓存的就是一个php数组之类；

Ecmall商城系统里面就用了这种方式；

**4、查询缓存**

其实这跟数据缓存是一个思路，就是根据查询语句来缓存；将查询得到的数据缓存在一个文件中，下次遇到相同的查询时，就直接先从这个文件里面调数据，不会再去查数据库；但此处的缓存文件名可能就需要以查询语句为基点来建立唯一标示；

按时间变更进行缓存

其实，这一条不是真正的缓存方式；上面的2、3、4的缓存技术一般都用到了时间变更判断；就是对于缓存文件您需要设一个有效时间，在这个有效时间内，相同的访问才会先取缓存文件的内容，但是超过设定的缓存时间，就需要重新从数据库中获取数据，并生产最新的缓存文件；比如，我将我们商城的首页就是设置2个小时更新一次；

**5、按内容变更进行缓存**

这个也并非独立的缓存技术，需结合着用；就是当数据库内容被修改时，即刻更新缓存文件；  
比如，一个人流量很大的商城，商品很多，商品表必然比较大，这表的压力也比较重；我们就可以对商品显示页进行页面缓存；

当商家在后台修改这个商品的信息时，点击保存，我们同时就更新缓存文件；那么，买家访问这个商品信息时，实际上访问的是一个静态页面，而不需要再去访问数据库；

试想，如果对商品页不缓存，那么每次访问一个商品就要去数据库查一次，如果有10万人在线浏览商品，那服务器压力就大了；

**6、内存式缓存**

提到这个，可能大家想到的首先就是Memcached；memcached是高性能的分布式内存缓存服务器。 一般的使用目的是，通过缓存数据库查询结果，减少数据库访问次数，以提高动态Web应用的速度、 提高可扩展性。

它就是将需要缓存的信息，缓存到系统内存中，需要获取信息时，直接到内存中取；比较常用的方式就是 key–>value方式；

<?php

$memcachehost = '192.168.6.191';

$memcacheport = 11211;

$memcachelife = 60;

$memcache = new Memcache;

$memcache->connect($memcachehost,$memcacheport) or die ("Could not connect");

$memcache->set('key','缓存的内容');

$get = $memcache->get($key); //获取信息

?>

**7、apache缓存模块**

apache安装完以后，是不允许被cache的。如果外接了cache或squid服务器要求进行web加速的话，就需要在htttpd.conf里进行设置，当然前提是在安装apache的时候要激活mod\_cache的模块。

安装apache时：./configure –enable-cache –enable-disk-cache –enable-mem-cache

**8、php APC缓存扩展**

Php有一个APC缓存扩展，windows下面为php\_apc.dll，需要先加载这个模块，然后是在php.ini里面进行配置：

[apc]

extension=php\_apc.dll

apc.rfc1867 = on

upload\_max\_filesize = 100M

post\_max\_size = 100M

apc.max\_file\_size = 200M

upload\_max\_filesize = 1000M

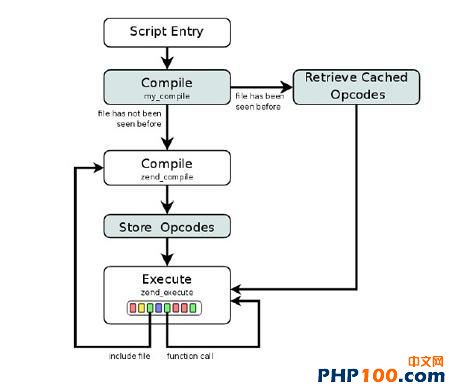
post\_max\_size = 1000M

max\_execution\_time = 600 ; 每个PHP页面运行的最大时间值(秒)，默认30秒

max\_input\_time = 600 ; 每个PHP页面接收数据所需的最大时间，默认60

memory\_limit = 128M ; 每个PHP页面所吃掉的最大内存，默认8M

**9、Opcode缓存**

我们知道，php的执行流程可以用下图来展示：  
  
首先php代码被解析为Tokens，然后再编译为Opcode码，最后执行Opcode码，返回结果；所以，对于相同的php文件，第一次运行时可以缓存其Opcode码，下次再执行这个页面时，直接会去找到缓存下的opcode码，直接执行最后一步，而不再需要中间的步骤了。

比较知名的是XCache、Turck MM Cache、PHP Accelerator等。