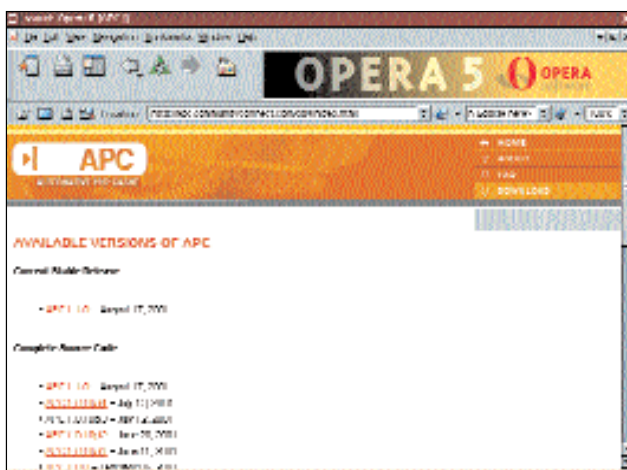


Gyorstár-megoldások PHP-hoz

Öveket becsatolni – sebességbe kapcsolunk!

Sok weboldal-üzemeltető cég találkozhat azzal a nehézséggel, hogy oldalainak letöltési ideje a látogatók számának növekedésével jelentősen megnő. Természetesen mint minden gond megoldásánál, itt is több lehetőség közül választhatunk. Akad olyan cég, amely a túlterheléssel járó nehézségeket gépvásárlással orvosolja. Ennél elegánsabb módszer, ha a leírásokban keresgélünk olyan áthidaló megoldás után, amellyel méréselhetjük a költségeket, és a megváltoztatandó állapothoz képest programból megvalósított módon érünk el javuló teljesítményt.



➔ <http://apc.communityconnect.com/download.html>

Írásunkban most az utóbbi megközelítéssel élve próbálunk megoldást lelteni a felmerülő gondokra: megismerkedünk az APC-vel (Alternative PHP Cache), a hozzá készített grafikus kezelőfelülettel, valamint a PEAR-rel. A Zend Technologies alkalmazásait pedig építhetjük megemlíthetjük.

Az egyik lehetséges megoldás: az APC

Az APC készítői olyan nyílt forráskódú programot szerettek volna készíteni, amely megállja a helyét az olyan kereskedelmi megoldások mellett, mint például a Zend Cache.

A program működési elve egyszerű: a PHP-kód a fordítás után a memóriába kerül, kivéve, ha már ott is van, mert akkor nem szükséges újra lefordítani. Ha ismét szükségünk van a kód lefordított kimenetére, egyszerűen kiolvassuk a gyorsítóból. A sebességnövekedés abból ered, hogy megtakarítjuk az újrafordítási időt. Mekkora mértékű lehet a teljesítményjavulás? Erre nem lehet pontos választ adni, mert a programjainktól is függ. Egyszerűbb programoknál a sebességnövekedés nem lesz szembetűnő, viszont ahol összetettebb a program és az általa megoldandó feladat, ott jó eséllyel számíthatunk kedvezőbb eredményre, mert esetükben lényegesen nagyobb a fordítási idő.

Telepítés

A fordítás és a telepítés kétféle módon történhet.

Ha már létezik a gépünkön PHP, és nem akarjuk az egészet újrafordítani, csupán az APC-t fordítsuk le:

```
./configure --enable-apc
--with-php-config=/a php-config helye
make
cp modules/php_apc.so /a kiterjesztések helye
```

A php.ini fájlba a következő bejegyzéseket kell beírni:

```
zend_extension= a kiterjesztések helye /php_apc.so
```

A következő három sorból csak egyet kell kiválasztani:

```
apc.mode = off
apc.mode = shm
apc.mode = mmap
```

Az első esetben kikapcsoljuk az APC-támogatást, a másodiknál a lefordított PHP-kódok a memóriába (shared memory) kerülnek, a harmadik esetben pedig egy-egy fájlba (memory mapped file). Erről a későbbiekben még szót ejtünk.

Ha azt szeretnénk, hogy az APC-támogatás bekerüljön a PHP-ba, vagyis az egészet újrafordítjuk, a következőket kell tennünk: csomagoljuk ki az APC-forrást a */PHP_forrás_elérési_útja/ext/apc* könyvtárba. A PHP-forrás legfelső szintjén adjuk ki a következő utasításokat:

```
autoconf
./configure --enable-apc <további kapcsolók>
make
make install
```

A php.ini fájlba ugyanazt a sort kell beilleszteni, amit az előzőekben a külön modulba fordítás esetén is.

Az APC használata

Először azt kell eldöntenünk, hol szeretnénk a lefordított fájlokat tárolni. A memóriában (shm) vagy fájlban (mmap)? Mielőtt erre válaszolnánk, ismerkedjünk meg az APC-hoz készített grafikus kezelőfelülettel. A programot PHP-ban írták és a PHP GD (Graphical Development) támogatását igényli, mivel a statisztikákhoz képfájlokat készít. Az APC grafikus kezelőfelületet magát a <http://apc.neuropeans.com> címről tölthetjük le, de az APC csomagjának *extras* könyvtárában is megtaláljuk. Telepítése egyszerű: másoljuk be az egészet oda, ahol webkiszolgálónk hozzáférhet a PHP-fájlokhoz. Az adatokat a webkiszolgáló vagy a PHP-kód segítségével érdekes védeni.

Az APC-vel való ismerkedés lezárásaképpen megosztanék néhány gondolatot a gyorsítárban lévő PHP-programok tárolásával, érvényességével kapcsolatban:

- A memóriában tárolt PHP-fájlokról adatokat csak a grafikus kezelőfelület segítségével szerezhethetünk, míg ha fájlban tároljuk őket, a változásokat könnyedén követhetjük.
- Fontos átgondolni a fájlok frissítését és élettartamát. Ha ugyanis valami már bekerült a gyorsítárba, akkor érvényességének lejár-

táig mindig onnan fogjuk megkapni. Ez jót és rosszat is jelenthet: jót, mert rendszerünk ezáltal gyorsabb lesz, hiszen az ügyfelek hamarabb kapnak választ a kiszolgálótól; hiba keletkezik viszont, ha a kiszolgálón tárolt programot módosítjuk, miközben az a gyorstárban már megtalálható.

Scriptname	Size (KB)	Last Access	Last Modified	TTL
/usr/share/php/Zend-1.0.0/Zend-1.0.0.dll	11.7	2000-01-01 00:00:00	2000-01-01 00:00:00	10
/usr/share/php/Zend-1.0.0/Zend-1.0.0.dll	11.7	2000-01-01 00:00:00	2000-01-01 00:00:00	10
/usr/share/php/Zend-1.0.0/Zend-1.0.0.dll	11.7	2000-01-01 00:00:00	2000-01-01 00:00:00	10
/usr/share/php/Zend-1.0.0/Zend-1.0.0.dll	11.7	2000-01-01 00:00:00	2000-01-01 00:00:00	10
/usr/share/php/Zend-1.0.0/Zend-1.0.0.dll	11.7	2000-01-01 00:00:00	2000-01-01 00:00:00	10
/usr/share/php/Zend-1.0.0/Zend-1.0.0.dll	11.7	2000-01-01 00:00:00	2000-01-01 00:00:00	10
/usr/share/php/Zend-1.0.0/Zend-1.0.0.dll	11.7	2000-01-01 00:00:00	2000-01-01 00:00:00	10
/usr/share/php/Zend-1.0.0/Zend-1.0.0.dll	11.7	2000-01-01 00:00:00	2000-01-01 00:00:00	10
/usr/share/php/Zend-1.0.0/Zend-1.0.0.dll	11.7	2000-01-01 00:00:00	2000-01-01 00:00:00	10
/usr/share/php/Zend-1.0.0/Zend-1.0.0.dll	11.7	2000-01-01 00:00:00	2000-01-01 00:00:00	10

Az APC használat közben

Lehetőségek a hiba kiküszöbölésére:

- PHP programból.

```
if(apc_reset_cache())
    print "Töröltem az összes bejegyzést";
else
    print "Nem sikerült törölni a bejegyzéseket";
```

A függvény csak az shm-nél működik.

- A bejegyzés élettartamának megadása a php.ini-ben.

```
apc.ttl = 10 ;10 másodpercet
```

A beállítás csak az shm-nél működik. Általánosan, minden gyorstárban lévő objektumra igaz lesz.

- Egyedileg is beállíthatjuk arra a PHP-programra, amelyből a függvényt meghívjuk.

```
apc_set_my_ttl(10);
```
- Bejegyzés törlése PHP-programból.

```
apc_rm(teljes_elérési_út_megadása);
```

A függvénynek a törölni kívánt PHP-fájlt és ne a gyorstárfájlt adjuk meg!

- A webkiszolgáló újraindítása.
Ez is egy lehetőség, de inkább ne éljünk vele, ekkor ugyanis az összes élő kapcsolat megszakad.

Tapasztalatok

- Az mmap használata során fontos a gyorstár könyvtárának megadása. A grafikus kezelőfelület érthetetlen okokból a /tmp könyvtárat adta vissza, miközben a php.ini-ben nem volt erre vonatkozó bejegyzés. Ráadásul az APC sem működött rendesen, míg a hibát nem javítottam.

```
apc.mode = mmap
apc.cachedir= /tmp
```

- A grafikus kezelőfelület 1.0.2-es változatának hosts.php-jában érdemes a módosítani következő sort, ellenkező esetben hibaüzenetet kapunk, miszerint nem meghatározott függvényt próbáltunk meghívni.

A hibás sor:

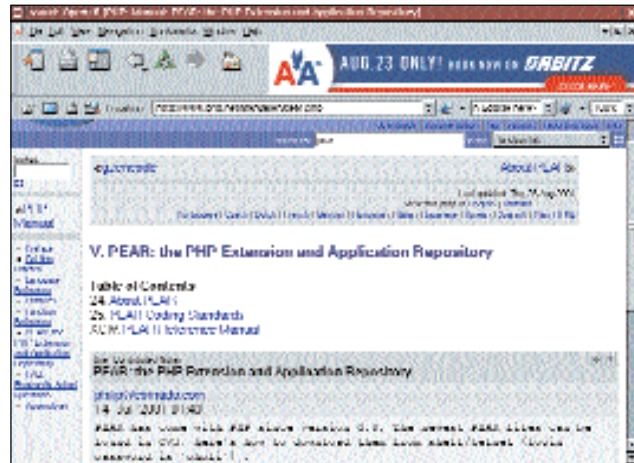
```
apc_restart_host($apc_hosts[$host]);
```

Helyette:

```
apc_reset_cache();
```

A gyakorlat

Lehetőségünk nyílik arra is, hogy egyszerű PHP-programok segítségével magukat a programkimeneteket tároljuk, így szükség esetén a megfelelő helyről meghívhatók – ez a hely lehet fájl,



➔ <http://www.php.net/manual/en/pear.php>

adatbázis vagy memória. Több megvalósítás is ismert, mint a PEAR Cache modulja, a Tontarta Cache, CacheIT stb. Ez nem teljesen azonos a cikk első felében bemutatott módszerrel, de használhatjuk annak kiegészítőjeként.

A PEAR és egy teljesítménypróba

A PEAR segítségével adatok, függvények, URL-ek tartalmának gyorsabb lekérdezésére nyílik lehetőségünk. Két internetes oldal letöltési idejét hasonlítottam össze egy olyan sebességteszt során, amikor is a letöltést többször egymás után megismételtem, majd a letöltési időket a végén összehasonlítottam. A próbát a gyorstár használatával és nélküle is elvégeztem.

Tested: http://www.clickagents.com
Downloaded: 1.1.0.0.0 (cache: 0)
Downloaded: 1.1.0.0.0 (cache: 0)
Tested: http://www.clickagents.com
Downloaded: 1.1.0.0.0 (cache: 0)
Downloaded: 1.1.0.0.0 (cache: 0)

Két gép összehasonlítása

A próbaprogram kódja

```
<?
include("Benchmark/Iterate.php");
include("Cache.php");
include("Cache/URL.php");

$FUNCTION_CACHE_CONTAINER = "file";
$FUNCTION_CACHE_CONTAINER_OPTIONS = array(
    "cache_dir" => "/tmp/",
    "filename_prefix" => "cache_");

function with($cache) {
    $data = get_cached_url($cache,10);
}

function without($cache) {
    $fd = fopen($cache,"r");
}

$num = 20;
$benchmark = new Benchmark_Iterate;

$benchmark->run
($num,'with','http://www.php.net/manual/en/');
$result = $benchmark->get();
$eredmeny = $result["mean"] *
$result["iterations"];
print $eredmeny . "<br>";
$benchmark->run
($num,'without','http://www.php.net/manual/en/');
$result = $benchmark->get();
$eredmeny = $result["mean"] *
$result["iterations"];
print $eredmeny;

?>
```

Maga a program rövid, nem bonyolult, viszont alkalmas arra, hogy az eddig leírtakat a gyakorlatban is megvizsgálhassuk. A programok forráskódja a *listán* látható.

Először beillesztjük a megfelelő PHP-kódokat, amelyek a méréshez és a gyorsítár használatához szükségesek. Az ezt követő két változó adja meg egyrészt, hogy a letöltött oldalakat fájlokban tárolnánk-e, másrészt hogy a kérdéses fájlok mely könyvtárban helyezkedjenek el és milyen előképzőt kapjanak. Ezután létrehozuk a *Benchmark_Iterate* osztály egy példányát, majd meghívjuk a függvényeket az iteráció számával, a függvények nevével és a letöltendő oldal URL-jével. Az eredmények a 4. ábrán láthatók és másodpercben értendők.

A gyorsítár kikapcsolása miatt a kívánt oldal minden ciklusban letöltődik, míg a bekapcsolásnál csak egyszer, mert azt követően már a gyorsítárból kapjuk meg. Ez adja az eredményekben felbukkanó eltérést.

Zend

Az előbb említett szabad forráskódú lehetőségek mellett kereskedelmi termékek is léteznek.

A Zend Technologies kifejezetten a PHP-hoz készít alkalmazásokat, így például a Zend IDE-t, a Zend Cache-t és a Zend Optimizert. Számunkra ez utóbbi kettő azért fontos, mert képesek PHP-programjaink végrehajtását felgyorsítani. A Zend Cache harmincnapos,

szabadon használható ingyenes változatát bárki letöltheti Linux, BSD vagy Solaris operációs rendszerekre. Kódunkat hatékonyabbá teszi és képes a fájlokat a gyorsítárban őrizni, hogy szükség esetén onnan tölthesse be őket. A Zend Optimizer teljesen ingyenes, nagy haszna, hogy egyszerűsíti és hatékonyabbá varázsolja programjainkat, így végrehajtásuk a megszokotthoz képest akár száz százalékkal is gyorsabbá válhat.



➔ <http://www.zend.com/store/products/zend-optimizer.php>

Összefoglalás

Két lehetséges megoldást láthattunk oldalaink felgyorsítására és mindkettővel javulást érhetünk el, főleg ha a gyakorlatban is alkalmazzuk őket. A következő számban egy másik érdekes PHP-modullal fogunk megismerkedni, de ennek neve addig maradjon titok.



Tóth László

(atom@isc.hu) A BME utolsó éves hallgatója, webfejlesztőként dolgozik a saját vállalkozásában. Szabadidejét legszívesebben a barát-nőjével tölti.

Kapcsolódó címek

Az APC hivatalos oldala

➔ <http://apc.communityconnect.com>

Az APC grafikus kezelőfelület oldala

➔ <http://apc.neuropeans.com>

A GD-modul letöltésének oldala

➔ <http://www.boutell.com/gd>

PHP-ről szóló cikkek és programok gyűjteménye

➔ <http://www.zend.com>

Zend Optimizer

➔ <http://www.zend.com/store/products/zend-optimizer.php>

A PHP-nyelv honlapja

➔ <http://www.php.net>

Magyar nyelvű PHP-könyv

➔ <http://www.kiskapu.hu/kiskapu/search.phtml?detailed=120449801>