Kako ubrzati PHP program

4.5 brze metode

Ivan Voras, ivoras@fer.hr

Zašto je potrebno ubrzanje?

- **PHP JE SPOR!**
- Zbog prilično "labave" sintakse, PHP interpreter mora prolaziti kroz puno više provjera nego za neke druge skriptne programske jezike
- PHP-ov "model programiranja" je često neoptimalan s resursima
- (PHP programeri su često neiskusni)
- Primjer: http://www.bagley.org/~doug/shootout/bench/matrix/

4 Brze metode

1. "PHP Akceleratori"
2. Template sustavi
3. Cache SQL upita
4. Cache generiranog sadržaja

1. PHP Akceleratori

- Ne zahtijevaju posebnu podršku (izmjene) u izvornom kodu
- Zahtijevaju administratorski pristup serveru, i više memorije za rad
- Obično izvedeni kao PHP modul, uključuju se izravno u tijek interpretiranja i izvršavanja PHP skripte
- Postoji nekoliko inačica, komercijalnih i open source

ZEND PHP Optimiser

- Komercijalni proizvod
- (od autora PHPa)
- Nudi mnogo više od drugih (open-source) rješenja: Zend Performance Suite
- Po nekim mjerenjima najbrže rješenje, ali nedostupno za ovaj pregled

APC: Alternative PHP Cache

- Najraniji besplatni open source PHP Cache
- Kao i druga slična rješenja, radi cache interne reprezentacije PHP koda, spremne za izvršavanje u PHP interpreteru
- Više se ne održava, iako radi i sa najnovijim verzijama PHP-a. (zadnja službeno podržana verzija: PHP 4.1.x)

AfterBurner Cache

- Open source
- Slično APC-u (nešto brži po mjerenjima)
- Više se ne održava, iako navodno radi sa novijim verzijama PHP-a (zadnja službeno podržana verzija: 4.1.x)

<ivoras@fer.hr>

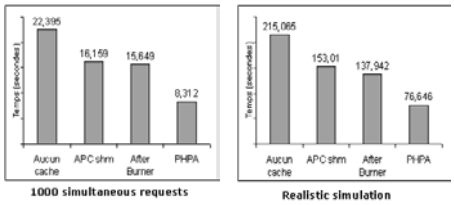
PHPA: ionCube PHP Accelerator

- Besplatan, ali nije open source! Dostupne su binarne verzije za FreeBSD (i386), OpenBSD (i386), Linux (i386, Alpha), Solaris (i386, Sparc)
- Po načinu cacheiranja sličan već opisanima
- Najbrže besplatno rješenje!

<ivoras@fer.hr>

Usporedba

- Mjerenja preuzeta sa http://www.idfr.net/etude_apache/optimisation.php



<ivoras@fer.hr>

2. Template sustavi

- Dvije velike prednosti:
  - Razdvajanje koda i dizajna što pridonosi njihovoj jasnoći
  - Mogu doprinijeti performansama ako se ispravno i u potpunosti koriste njihove mogućnosti
- Posebno efikasni u suradnji sa PHP akceleratorima

<ivoras@fer.hr>

Općenite mogućnosti

- Originalna namjena im je razdvajanje koda od dizajna – sve druge mogućnosti naknadne, i nisu prisutne kod svih biblioteka
- Dodatne mogućnosti: cache generiranog sadržaja, varijabli ili oboje

<ivoras@fer.hr>

Neki template sustavi

- Smarty
- FastTemplate
- PHPLib
- HTMLTemplate
- Biblioteke su različitih složenosti i namjena, a ističe se Smarty sustav jer nudi najviše mogućnosti

<ivoras@fer.hr>

Usporedbe brzine

- Mjerenja sa http://www.phpinsider.com/benchmarks/

print var 500			
Template Lang.	**Accelerated?**	**Requests/second**	**Graph**
FastTemplate 1.1.0	No	20.84	
PHPLib 7.2d	No	55.64	
TemplatePower 1.6.2	No	4.69	
Smarty 1.5.2	No	8.13	
Smarty 2.0	No	12.94	
FastTemplate 1.1.0	Yes	30.53	
PHPLib 7.2d	Yes	111.83	
TemplatePower 1.6.2	Yes	5.13	
Smarty 1.5.2	Yes	75.44	
Smarty 2.0	Yes	109.26	

<ivoras@fer.hr>

3. Cache SQL upita

- U općem slučaju nije dobro miješati se u dohvat podataka
- Ideja: izraditi cache za rezultate SQL upita
- Mogu se dobiti vrlo veliki dobitci ako se koristi puno identičnih upita (puno složenije ako su upiti ne-identični)
- Problemi: smještanje podataka (veličina) i njihova dostupnost iz PHP skripti (shareing), nemogućnost dijeljenja podataka između sessiona (sigurnost!)

<ivoras@fer.hr>

Moguće izrade

- Samostalna izrada
- Korištenjem PEAR Cache ili Cache_Lite klasa
- Dijeljenje podataka:
  - Unutar sessiona korištenjem ugrađenih session mehanizama
  - Između sessiona: nema gotovih zadovoljavajućih mehanizama (prijedlozi: shared memorija, filesystem, DB baze…)
- Moguće implementirati u biblioteci za pristup bazi, tako da najveći dio aplikacijskog koda ostaje neizmijenjen

<ivoras@fer.hr>

Efikasnost

- Efikasnost uvelike ovisi o strukturi web aplikacije
- Ako postoji mnogo identičnih upita (npr. unutar jednog sessiona, prilikom učitavanja svake stranice), dobitci su vrlo veliki
- Problemi:
  - Veliko zauzeće memorije
  - Ponekad i identični upiti trebaju vratiti različite rezultate (konkurentnost, problem zastarijevanja podataka)

<ivoras@fer.hr>

4. Cache generiranog sadržaja

- Obično zahtijeva izmjene u kodu i strukturi aplikacije (ne nužno…)
- Zahtjeva dobro poznavanje aplikacije i dinamike pristupa za efikasnu implementaciju
- Potencijalno najveći dobitci, izbjegavanjem izvršavanja i PHP koda i SQL upita

<ivoras@fer.hr>

Implementacija

- Ne mora biti kompleksna
- Ponekad je izravna posljedica modularnog dizajna
- PHP Output Buffering, ob_*

```
If ($vrijeme_sadrzaja+60 > time())
  ob_start()
  include ("prikazi_sadrzaj.php")
  $sadrzaj = ob_get_contents()
  $vrijeme_sadrzaja = time()
  ob_end_clean()
}
echo $sadrzaj
```

<ivoras@fer.hr>

4.5-ta metoda

- Najsporija za implementiranje
- U današnje vrijeme se smatra da je nedovoljno cost/benefit isplativa
- Jako varijabilni rezultati

<ivoras@fer.hr>

Educiranje developera

- Općenite programerske prakse!
- Efikasno korištenje jezika, baze (optimiranje SQL upita), regularnih izraza…
  - "Code complete" (Steve C. McConell, MS Press)
  - "Pisanje dobrog koda" (Steve Maquire, MS Press)

<ivoras@fer.hr>

Optimistična procjena ubrzanja

- Procjene za FER web:
  - Cache SQL koda: oko 1.5x
  - Smarty + PHP Accelerator: oko 8x
  - Cache generiranog HTMLa: oko 5x
- Ubrzanja nisu bila uvedena planski, već naknadno i pazilo se da se ne utječe na postojeći kod: ukupna ubrzanja od oko 10-15x **bez znanja developera**
  - Planskim uvođenjem bi se moglo postići više ☺

<ivoras@fer.hr>