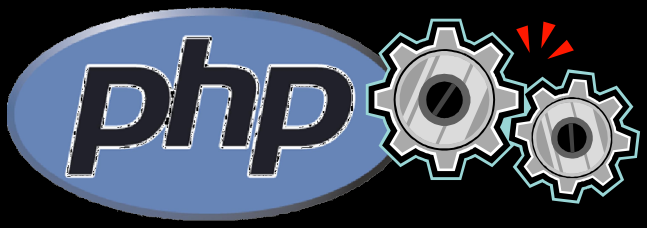
PHP
Tuning & Optimizing



Jinho Jung

2007. 04.19

목차

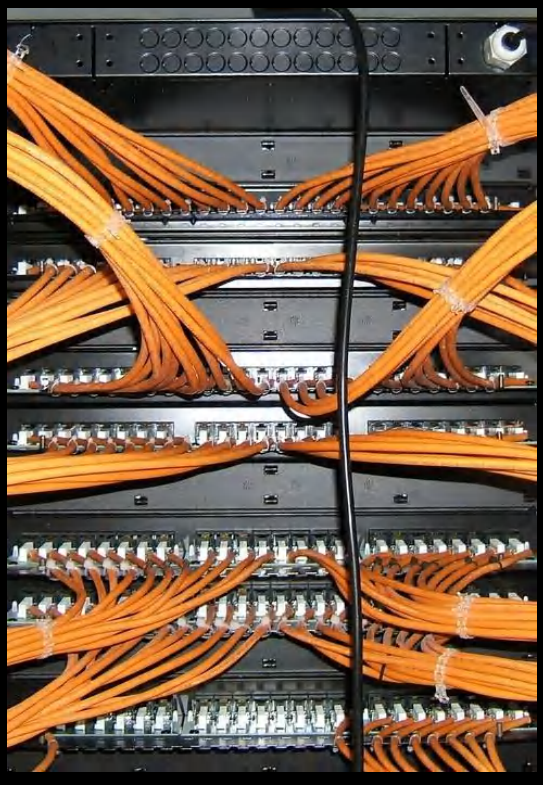
- 최적화에 대해 : 10%
  - 목적 , 대상 , 단계
- 문제점을 찾는 방법과 도구 : 20%
  - APD , Xdebug
- PHP 튜닝 : 40%
  - PHP4 와 PHP5
  - SQL, 함수
  - 캐시
  - PHP 설정파일
  - H/W 튜닝 팁
- 성능 TEST : 10%
  - ab , WAST

강의 목표

- PHP 의 튜닝에 필요한 각종 분석 방법과 도구를 살펴 보고 빠르고 쉽게 문제를 해결할 수 있는 방법을 소개한다

WEB 서비스의 5S

- Security
- Stability
- Speed
- Scalability
- Save money!



최적화란 무엇인가?

- " Identify the major bottlenecks in your application and target those for optimization "
  - Practical PHP Performance

최적화의 목적?

- 비용
  - TCO : H/W, N/W, 개발/운영 인력, 시간
- 안정성
  - NO Downtime! (24 H/365 D)
- 경험
  - 빠른 속도 = 사용자 만족

- " 최소비용으로 빠르고 안정적인 서비스로 사용자 경험을 향상시키자! "

최적화의 대상

- N/W : IDC 변경
- H/W : 서버 Upgrade, 서버 증설
- O/S : O/S Tuning, Upgrade, New O/S
- Web Server : Apache Tuning, Upgrade
- DB Server : MySQL/ORACLE Tuning
- Programming : PHP Tuning
- Presentation Layer : Semantic HTML/CSS
- 전 영역에 걸쳐 균형 있는 최적화 필요

최적화 단계

1. 문제 발견 :
   - CPU Load, 응답시간, 시스템 다운!
2. 원인 찾기 : Profiling
3. 문제 해결 : Tuning
4. 테스트 : Performance Testing

문제점 찾기

- “If you can’t find the part that makes it slow, you’re just wasting your time.”

- 효과적인 도구 : profiler
  - APD
  - Xdebug
  - mod_log_config
  - Zend IDE
  - Benchmark_Profiler

APD : Advanced PHP debugger

- Engine-level profiler/디버거
- 불필요한 부하가 적다
- 인터랙티브 디버깅 지원
- 이벤트 기반 로깅
  - Function 호출, Arguments 전달 등
- http://pecl.php.net/package/apd

APD : 설치와 사용법

1. # pear install apd
2. php.ini : 경로 추가

```
zend_extension = /absolute/path/to/apd.so
apd.dumpdir = /absolute/path/to/trace/directory
apd.statement_tracing = 0
```

3-1. 스크립트 시작 부분에 추가, 실행

```
apd_set_pprof_trace();
```

3-2. 명령 행에서 실행할 경우: '-e'

```
$ php -e -f script.php
```

4. 로그 파일 : apd.dumpdir/pprof_pid.ext
5. 분석 : pprofp 이용

APD : pprofp 명령

```
$ pprofp -R /tmp/pprof.22141.0 ( -R : sorts by time spent )

Trace for /home/test/test_apd.php
Total Elapsed Time = 0.00
Total System Time  = 0.00
Total User Time    = 0.00


Real       User       System          secs/    cumm
% Time (excl/cumm)  (excl/cumm)  (excl/cumm) Calls    call    s/call  Memory Usage
    Name
--------------------------------------------------------------------------------
100.0 0.00 0.00  0.00 0.00  0.00 0.00    1  0.0000   0.0009         0 main
56.9 0.00 0.00  0.00 0.00  0.00 0.00    1  0.0005   0.0005         0 apd_set_pprof_trace
28.0 0.00 0.00  0.00 0.00  0.00 0.00   10  0.0000   0.0000         0 preg_replace
14.3 0.00 0.00  0.00 0.00  0.00 0.00   10  0.0000   0.0000         0 str_replace
```

Xdebug

Xdebug

- 일반적인 profile Tool + ?
- 사용자 함수의 모든 parameter 표시
- 무한 루프 방지
- 스크립트 실행 분석
- GUI : Kcachegrind / KDE

Xdebug 인스톨

Xdebug

1. PEAR/PECL 이용

   # pecl install xdebug-beta

2. php.ini 에 경로 추가

   zend_extension="/usr/local/php/modules/xdebug.so"

3. webserver 다시 시작

4-1. phpinfo() 에서 Xdebug 모듈 확인

4-2. 명령행 실행

   $ php -m script.php

Xdebug

Xdebug

Function Summary Profile (sorted by avg. execution time)

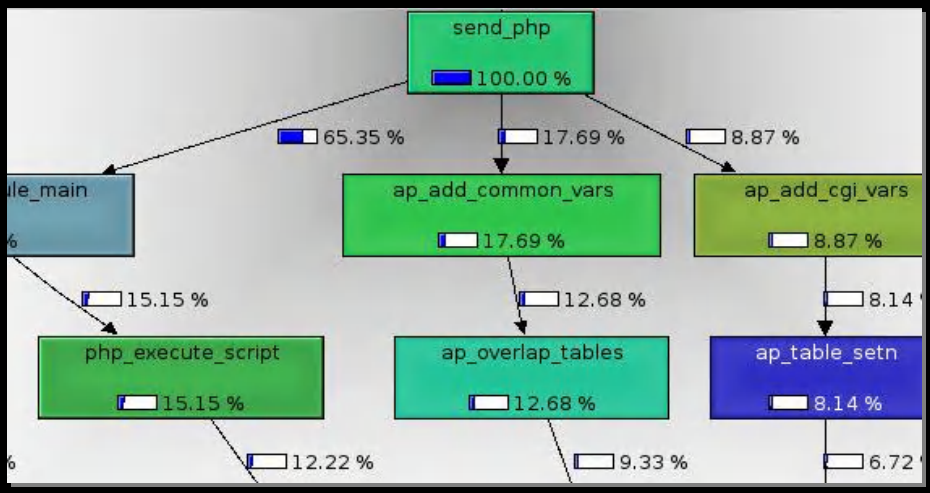
Total Time Taken	Avg. Time Taken	Number of Calls	Function Name
0.0011290550	0.0011290550	1	*test3
0.0008260127	0.0004130063	2	*test2
0.0006610128	0.0001652532	4	*test1
0.0000969507	0.0000969507	1	*my_class::my_method
0.0000930356	0.0000930356	1	*my_class->my_method
0.0001200305	0.0000300076	4	rand
0.0000340019	0.0000170009	2	explode
0.0000180922	0.0000045230	4	nl2br
0.0000169771	0.0000042443	4	urldecode
0.0000108776	0.0000036259	3	urlencode

Opcode Compiling: 0.0006699562
Function Execution: 0.0028060668
Ambient Code Execution: 0.0019309527
Total Execution: 0.0047370195

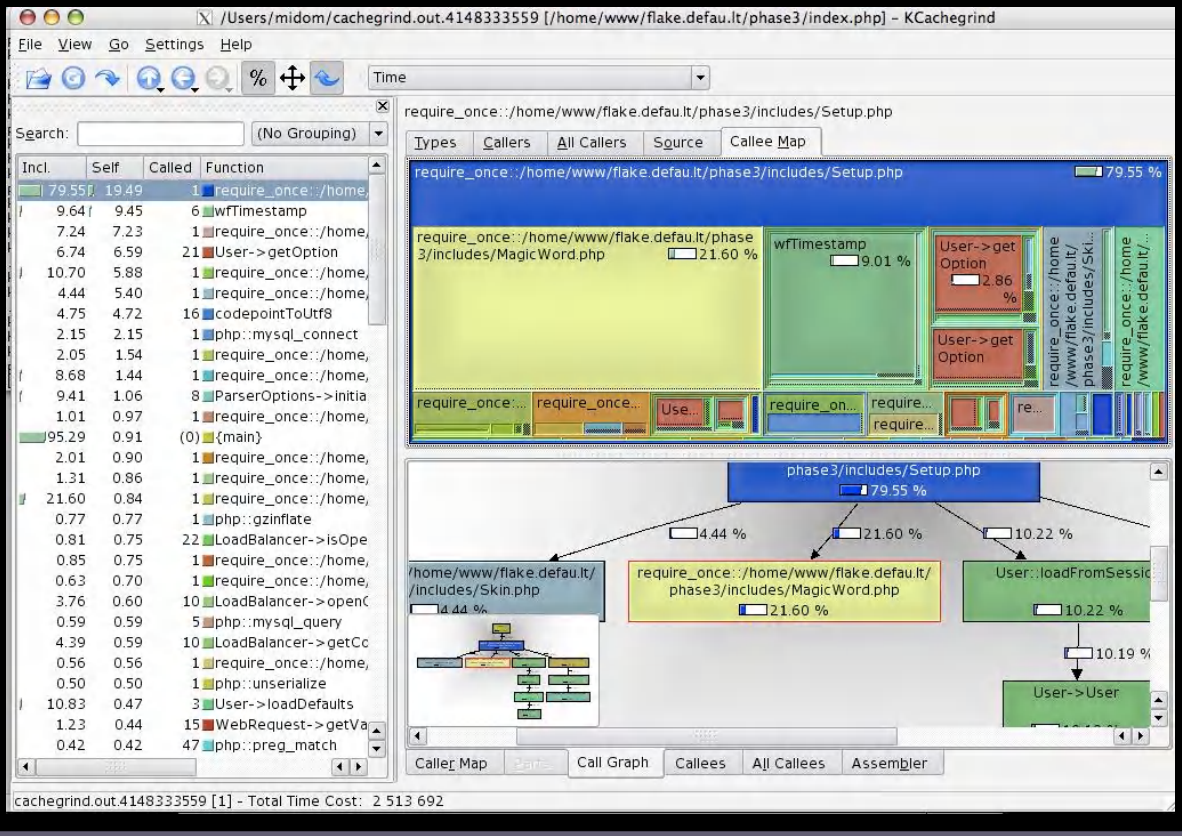
Total Processing: 0.0054069757

Kcachegrind

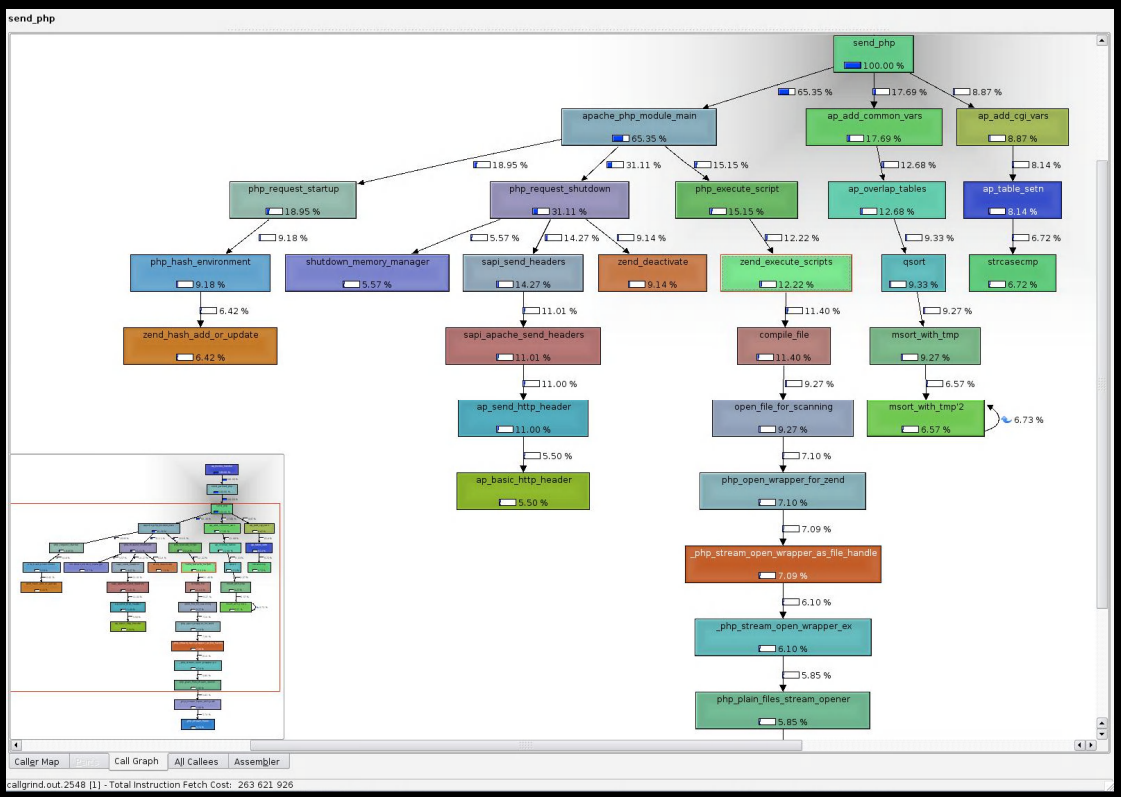
- http://kcachegrind.sourceforge.net
- KDE 기반
- Profiling Data 를 효과적으로 표시



Kcachegrind

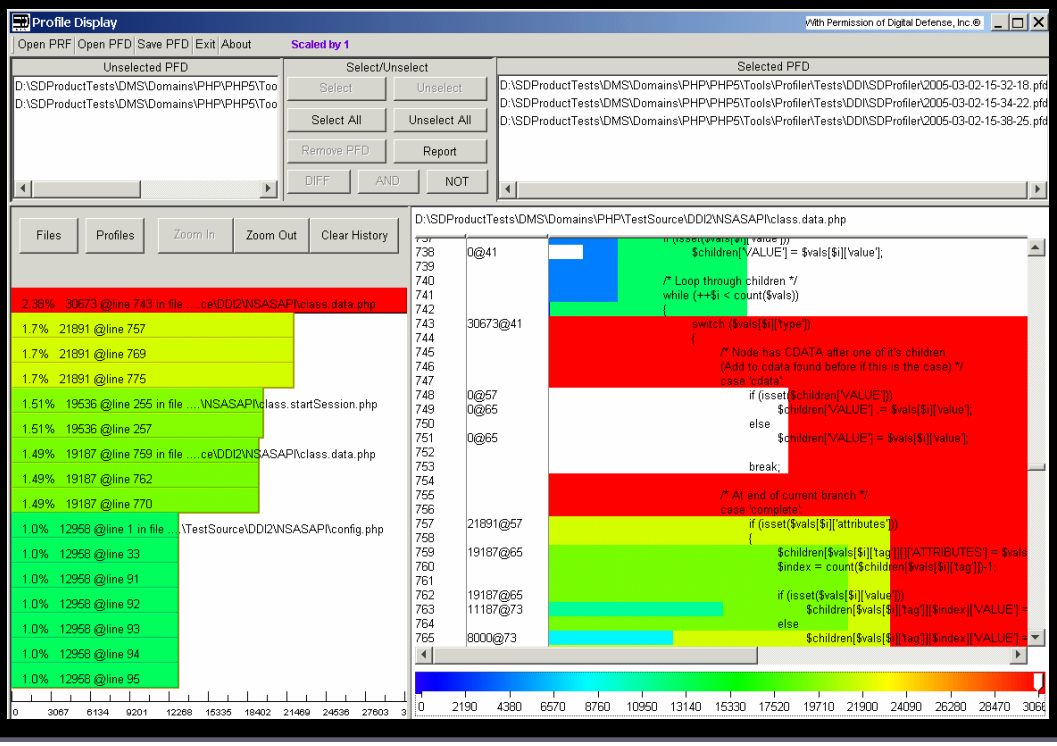


Kcachegrind



PHP Profiler Tool : Windows

- Semantic Designs – http://www.semdesigns.com



Speed Up!
PHP Tuning

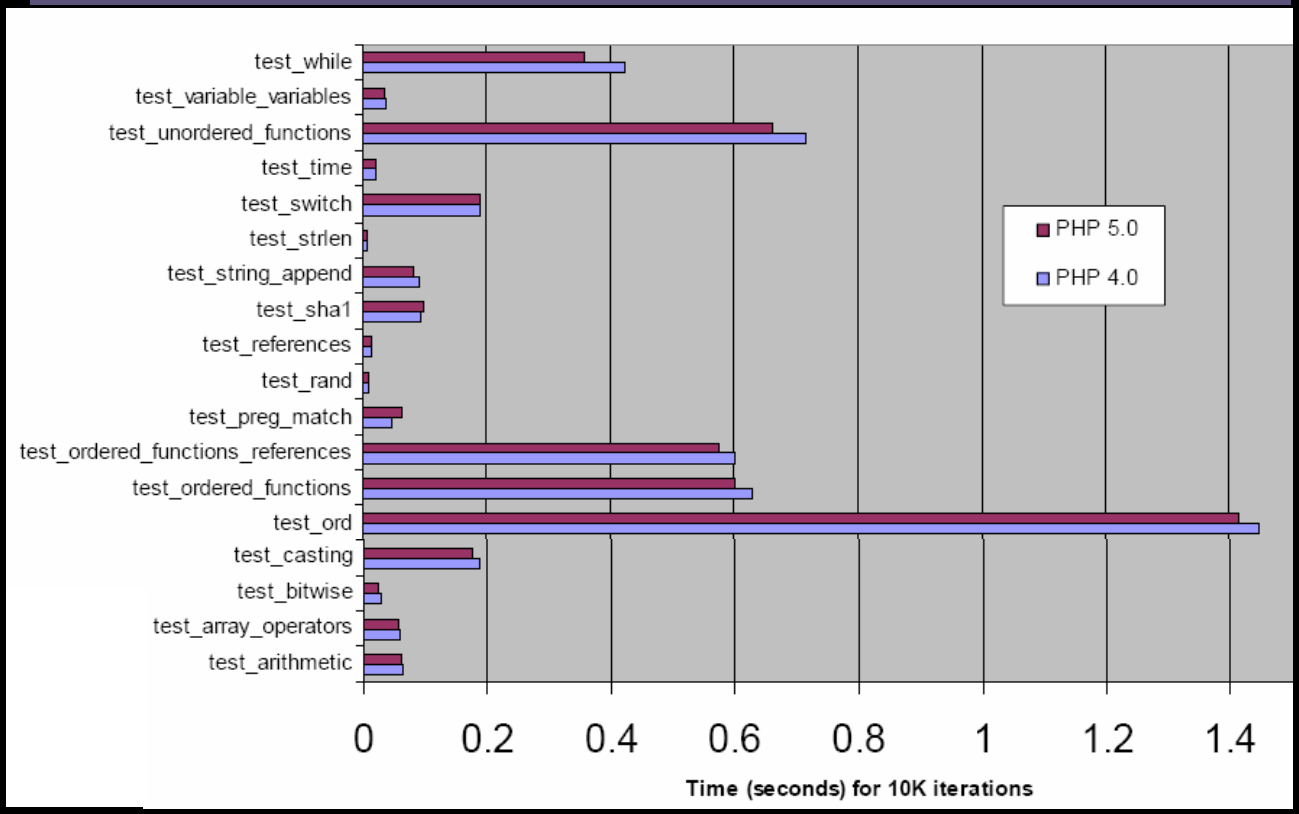
PHP Tuning

- PHP4 vs. PHP5
- 효율적인 SQL 사용
- 효율적인 함수 사용
- Opcode 캐시
- php.ini : PHP 설정파일
- H/W 튜닝 팁

PHP4 vs. PHP5

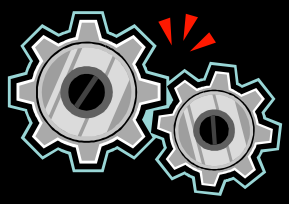
- PHP5 가 PHP4 보다 좀더 빠르게 설계
- Zend Engine 성능 향상
- 새로운 Extension
  - PHP Data Objects (PDO)
    - PHP 5.1 : 기본지원
    - PHP 5 : PECL extension
  - SQLite , SimpleXML
- 개선된 OOP 지원
- 개선된 CLI : -B , -R , -F , -E , -H

PHP4 vs. PHP5 성능비교



PHP4 vs. PHP5 성능비교

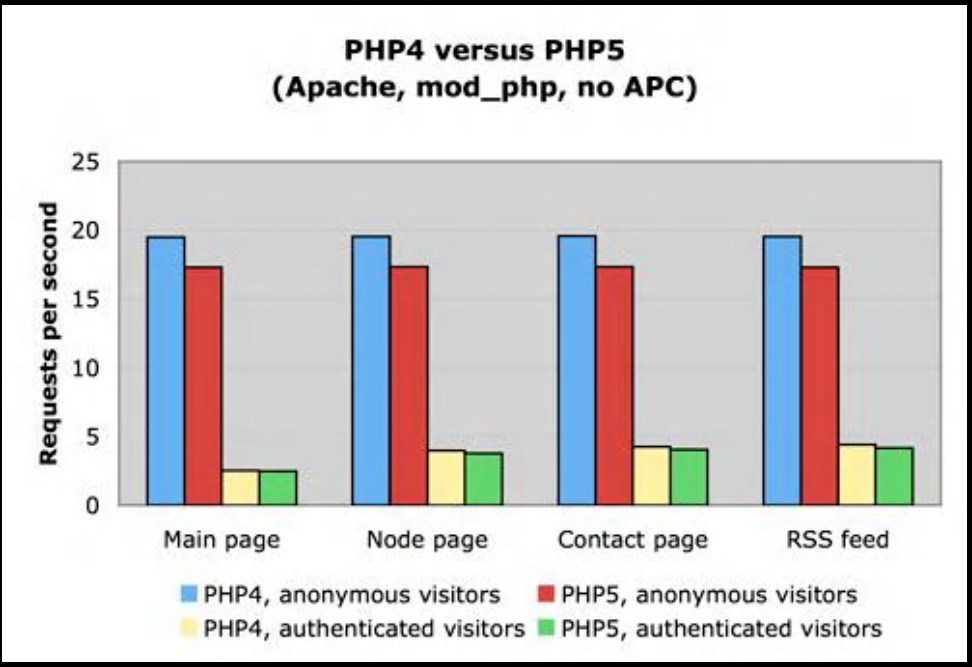
- 대부분의 경우 PHP5 가 좀 더 빠른 수행 성능을 보임 : 최대 20%



- 일부 경우에 PHP4 가 더 빠름
  - preg_match : PHP4 가 29.37% 더 빠름
- 출처 :http://www.sourcelabs.com/pdfs/SourceLabsPHP4vsPHP5.pdf

PHP4 vs. PHP5 성능비교

- http://buytaert.net/drupal-performance

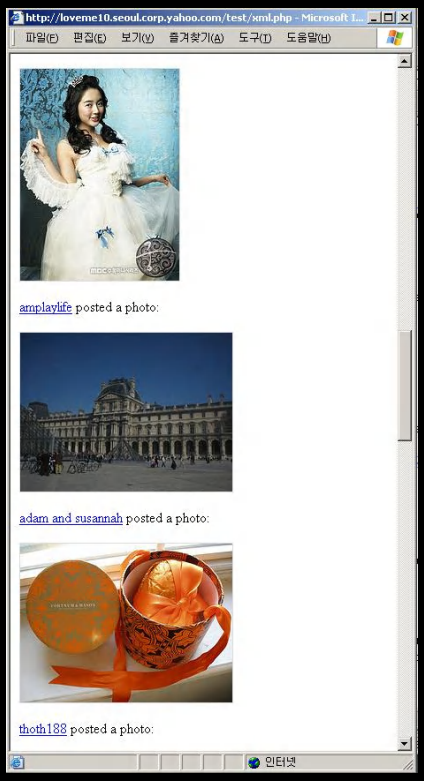


[DEMO] PHP5 : SimpleXML

- 2 line code

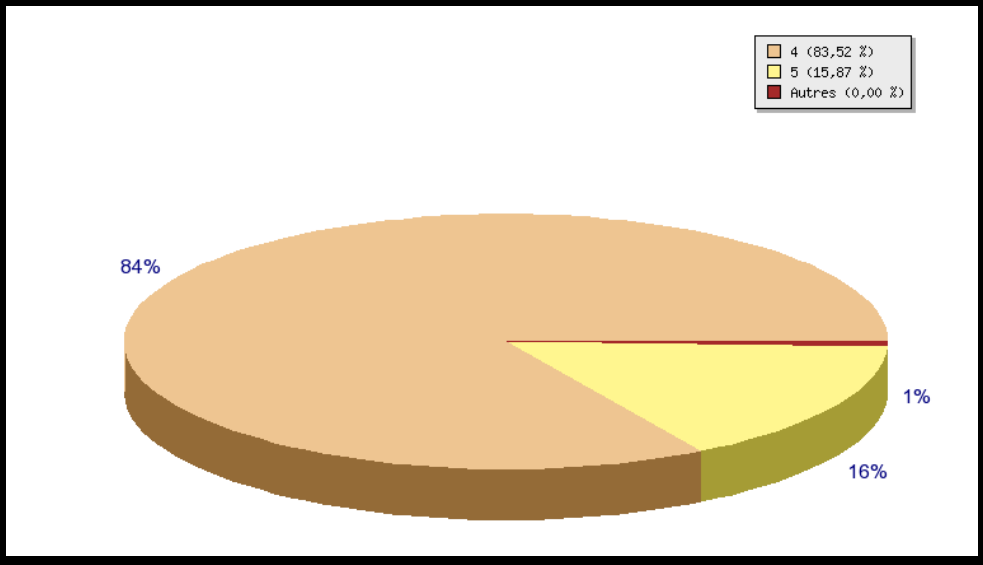
```
<?php
$url = 'http://www.flickr.com/services/feeds/photos_public.gne';
foreach(simplexml_load_file($url)->entry as $it) echo $it->content;
?>
```

[DEMO] PHP5 : SimpleXML



PHP5 Version 사용현황

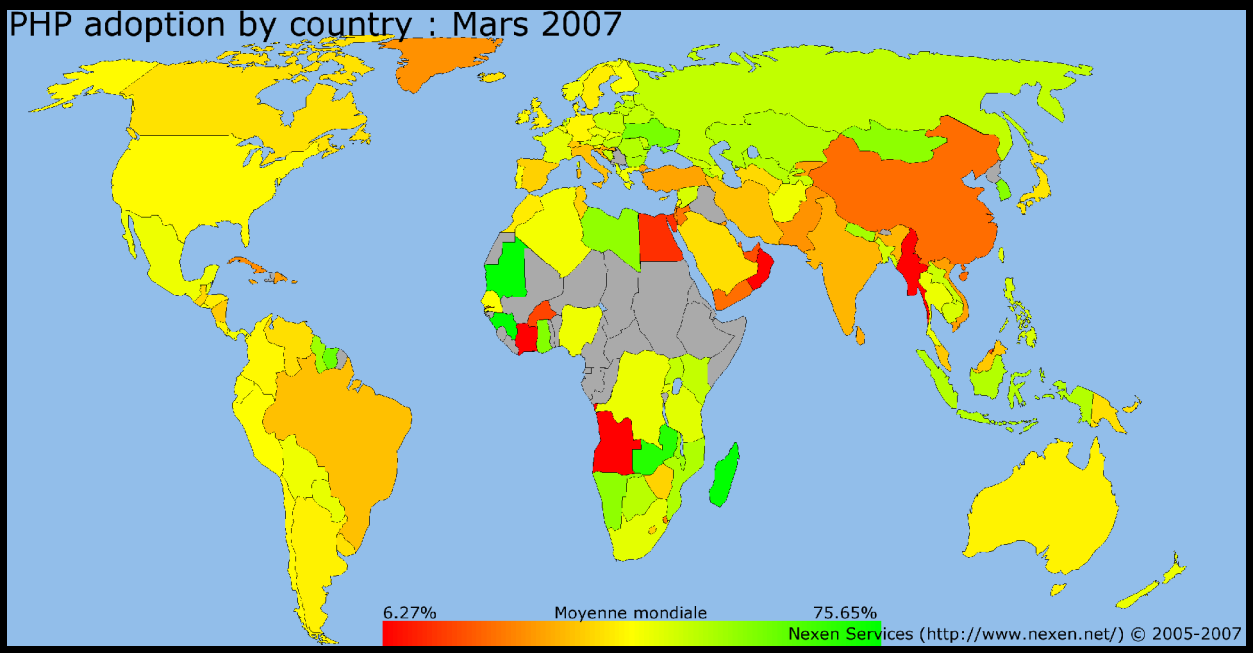
- PHP5 : 3년 전 Release
- 80% 이상 여전히 PHP4 사용 중



- http://www.nexen.net

PHP 사용 현황 : 국가별

- http://www.nexen.net



효율적 코딩 : Loop

- Bad

```
while ($a = 0; $a < count($array); $a++)
{
        echo $array($a);
}
```

- Good

```
$arraySize = count($array)
while ($a = 0; $a <$arraySize; $a++)
{
        echo $array($a);
}
```

효과적인 SQL 사용

- SQL 은 여러분의 코드보다 빠르다
- 문제는 잘 못 사용하는 경우
  - Query 수를 줄여라
  - 필요한 정보만을 가져와라
- Bad

```
SELECT * FROM users
WHERE username = “loveme”
```

- Good

```
SELECT ‘user_id’,‘name’ FROM users
WHERE username = “loveme”
```

효과적인 함수 사용

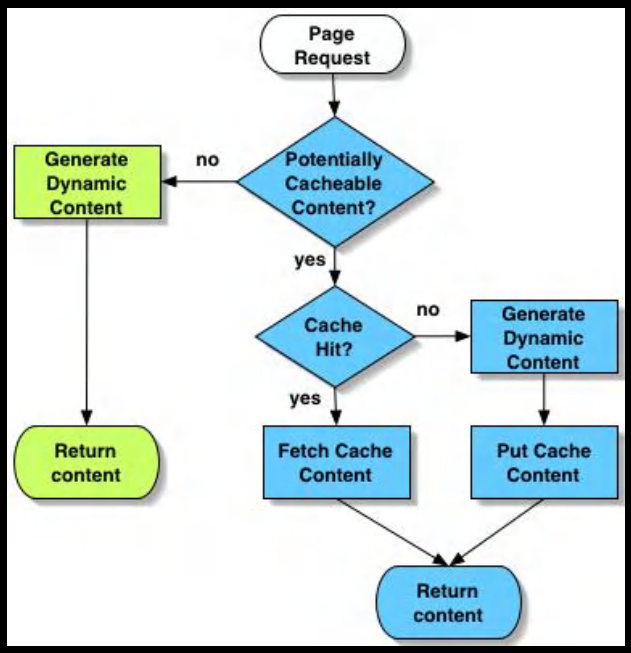
- mb_func() 대신 iconv() 사용
  - CPU 부하 10~20% 감소
- 문자열 비교에 substr() 를 불필요한 사용금지
  - X : ('http' == substr($path, 0, 4))
  - O :(strncmp($path, 'http', 4))
- File Include : 절대/상대 경로 사용
  - X : include "file.php";
  - O : include "/path/to/file.php";
  - O : include "./file.php";

효과적인 함수 사용

- 모든 스크립트에서 같은 파일을 Include 하지 말 것
  - 적절히 분리
- include 는 5개 미만 사용
- 빈 문자열은 strlen() 사용금지
  - X : strlen()
  - O : empty() or strcmp()

Opcode 캐시

- PHP 스크립트는 Opcode로 컴파일 된 후 실행된다
- 스크립트가 변경되지 않았다면 이 작업은 불필요하다

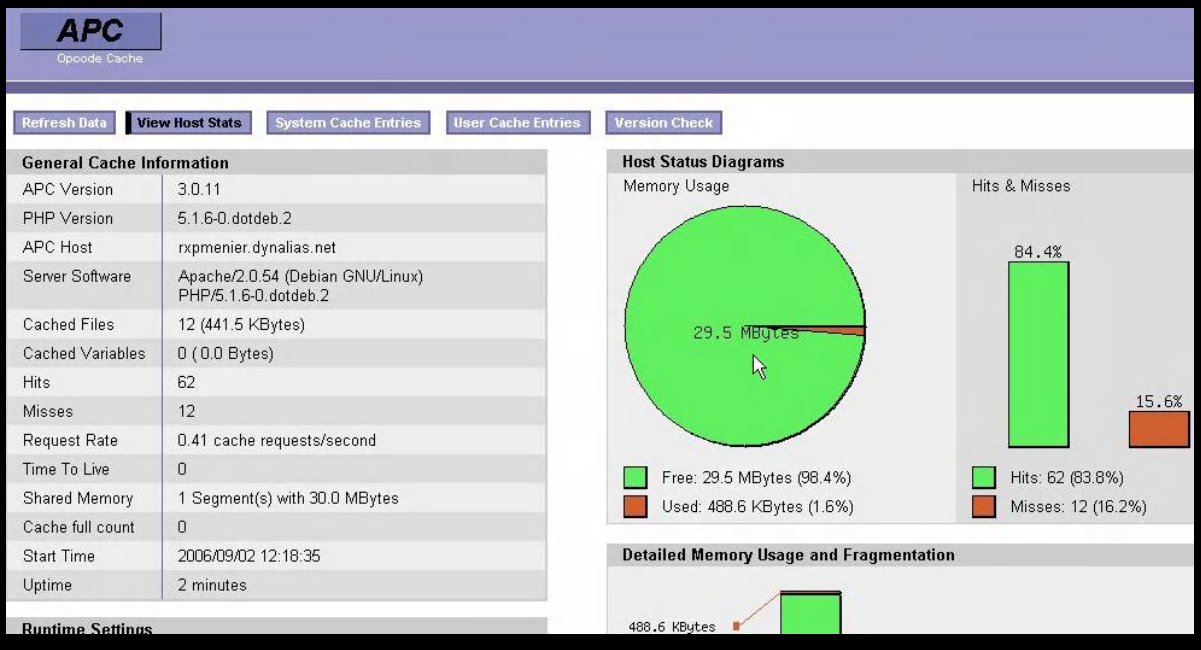


Opcode 캐시

- APC
  - 무료, PHP4 & PHP5,꾸준히 기능 개선
- Zend Suite
  - 상용, PHP4 & PHP5
  - 다양한 기능, 기술지원
- PHPA :
  - 무료, PHP4 , 개발중단
- Turck MMCache :
  - 무료, PHP4, 개발 중단

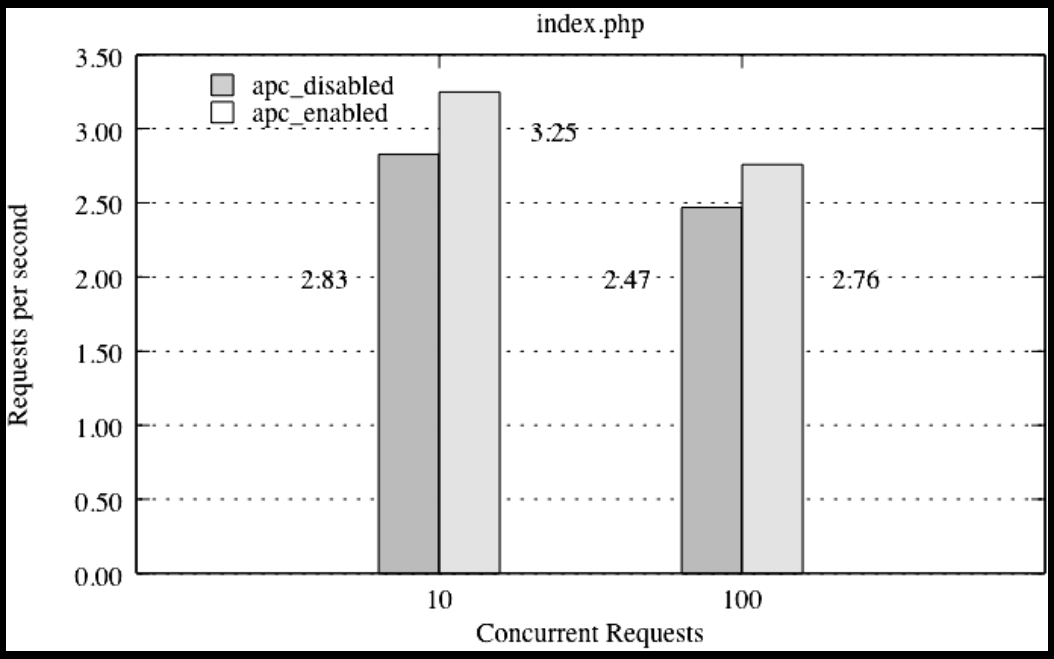
APC : Alternative PHP Cache

- PHP 5 용 무료 Cache
  - http://pecl.php.net/apc



APC 성능

- 약 15~20% 성능 향상 효과



Static HTML

- 동적 사이트인 경우에도 그다지 자주 변하지 않는 페이지 있음
- 주기적 Static HTML 생성
- Crontab / wget 이용

```
$ crontab -e

*/5 * * * * wget -q  http://localhost/index.php ₩
--output-document= /home/www/index.html
```

Output Control

- output_buffering 기능으로 성능개선

```
<?php
ob_start();
echo "Hello₩n";
setcookie("cookiename", "cookiedata");
ob_end_flush();
?>
```

php.ini : PHP 설정파일

- PHP 설정은 스크립트 실행에 중요한 영향을 미침
  - register_globals Off (4.2.0부터 기본적용)
  - magic_quotes_* Off
  - expose_php Off
  - register_argc_argv Off (for non-cli)
  - always_populate_raw_post_data Off

php.ini : PHP 설정파일

- open_basedir
  - 스크립트가 지정된 경로에 있는 경우만 실행
  - 유용하지만 불필요한 시스템 콜 사용
  - 반드시 필요한 경우만 사용
- variables_order
  - variables_order = "GP"

H/W 튜닝 팁

- PHP 스크립트의 속도 : CPU 가 좌우
  - Dynamic page : 빠른 CPU
  - Static Page : 많은 RAM
- Apache 2.0 : Thread 사용,속도 개선
- Output Buffe 사용 : 5~15% 성능개선
- PHP4 컴파일 옵션 (PHP 개발팀 추천)

  --enable-inline-optimization --disable-debug

Performance TEST

ab : ApacheBench

- Apache HTTP server benchmarking tool
- 웹서버의 HTTP 성능을 테스트
- 초당 처리 가능한 Request 수를 측정

```
$ ab -n 1000 -c 10 http://localhost:80/
```

ab : ApacheBench

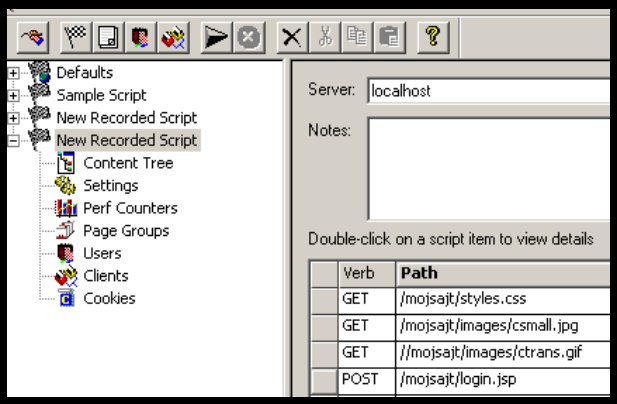
```
Server Hostname:        localhost
Server Port:            80
Document Path:          /
Document Length:        3326 bytes

Concurrency Level:      10
Time taken for tests:   0.641 seconds
Complete requests:      1000
Failed requests:        0
Broken pipe errors:     0
Total transferred:      3753226 bytes
HTML transferred:       3335978 bytes
Requests per second:    1560.06 [#/sec] (mean)
Time per request:       6.41 [ms] (mean)
Time per request:       0.64 [ms] (mean, across all concurrent requests)
Transfer rate:          5855.27 [Kbytes/sec] received

Connnection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    2   1.3      2      11
Processing:     2    4   4.7      3     146
Waiting:        0    3   4.6      2     145
Total:          5    6   4.6      6     149
```

WAST

- MS Web Application Stress Tool
- ab 의 한계 : 하나의 PHPSESSID
- 세션과 무관하게 throughput (pages/second) 일정
- 세션을 사용할 경우 결과 신뢰 할 수 없음



Microtime()

```
<?php
        $start = microtime(true);
                /*
                ...
                Your code goes here
                ...
                */
        $end = microtime(true);
        echo "<br>Time: " . ($end -$start) .
?>
```

Microtime()

- 장점
  - 사용하기 쉬움
  - 결과가 웹 페이지에 출력
  - 추가로 설치할 필요 없음
- 단점
  - 개개의 함수의 수행시간이 아닌 전체 페이지 단위의 수행시간만 표시
  - 페이지 속도가 약간 저하됨

More Information

- A HOWTO on Optimizing PHP
  - http://phplens.com/lens/php-book/optimizing-debugging-php.php
- PHP Performance Profiling
  - http://www.linuxjournal.com/article/7213
- Caching Strategies for Load Reduction on High Traffic Web Applications
  - http://alexander.kirk.at/papers/caching-strategies/

Thanks

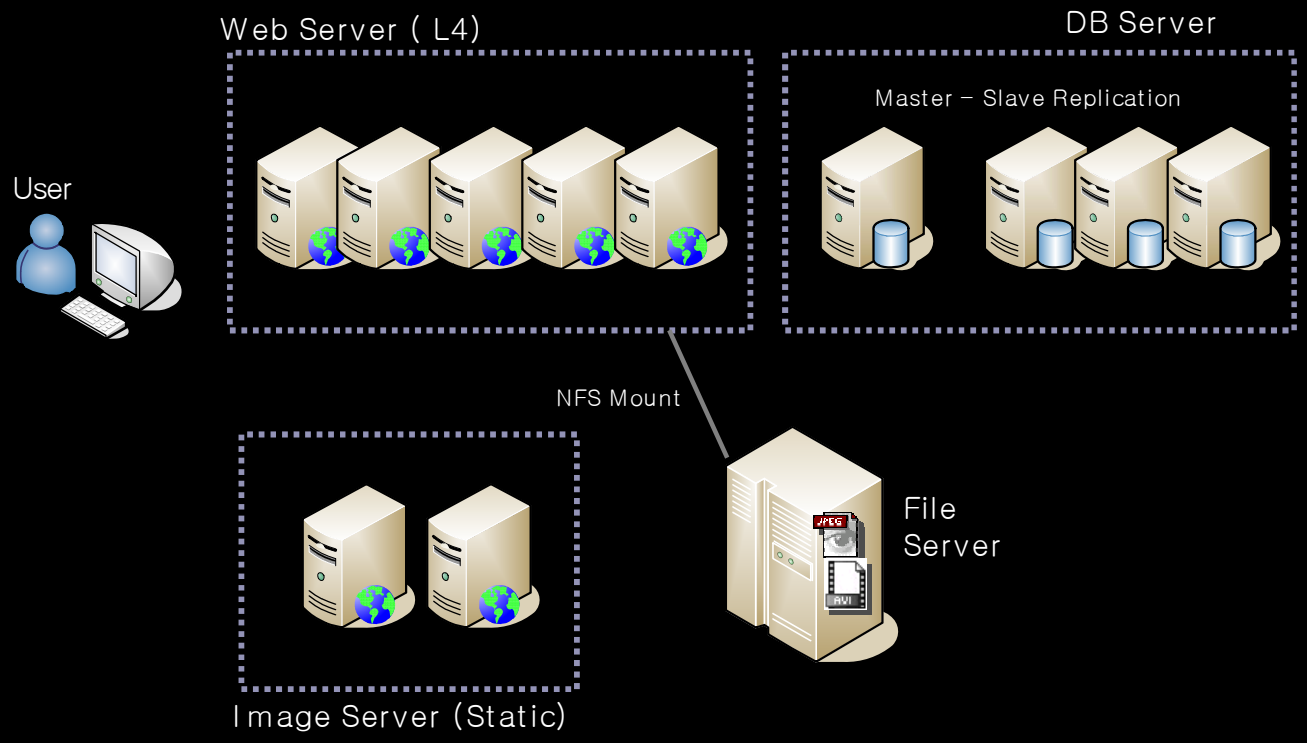
All Photos from flickr

Appendix

[참고] 서버 구성에 따른 처리량

서버	PV/Day	구성
1	50 만	WEB + DB
2	50 -100 만	WEB(1),DB(1)
4	100 -200 만	WEB(2), DB Mater(1) DB Slave(1)
12	500 만	WEB(5), IMG(2),File(1), DB Mater(1) ,Slave(3)

[참고] 500만 PV : 서버 구성 예



Apache Configuration

- Apache child process 중요 튜닝 대상
- StartServers :
  - 예상되는 평균 requests에 맞춰
- MaxSpareServers :
  - 피크타임을 고려 충분히 크게
- MaxClients :
  - 서버에서 처리가능 프로세스의 2/5 정도
- MaxRequestsPerChild :
  - 이론상으로 무한대(0)
  - 실제로는 충분히 큰 숫자로 설정