APC & Memcache the High Performance Duo

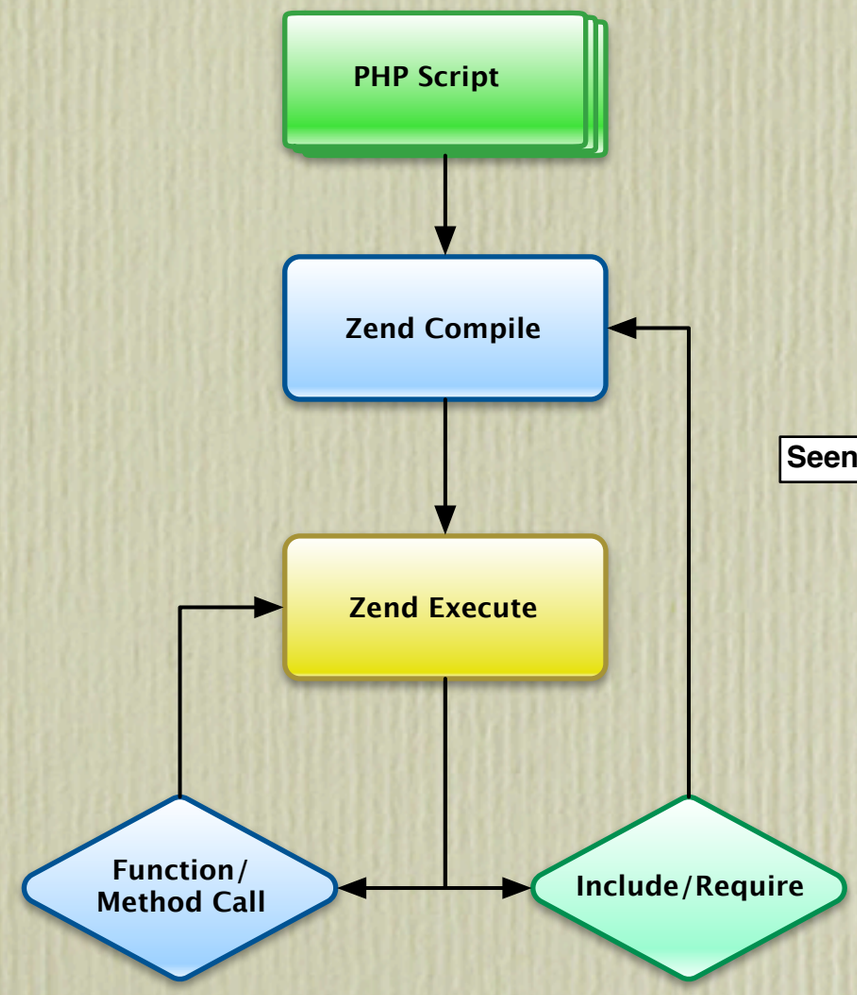
ZendCon 2009 - Ilia Alshanetsky

What is APC?

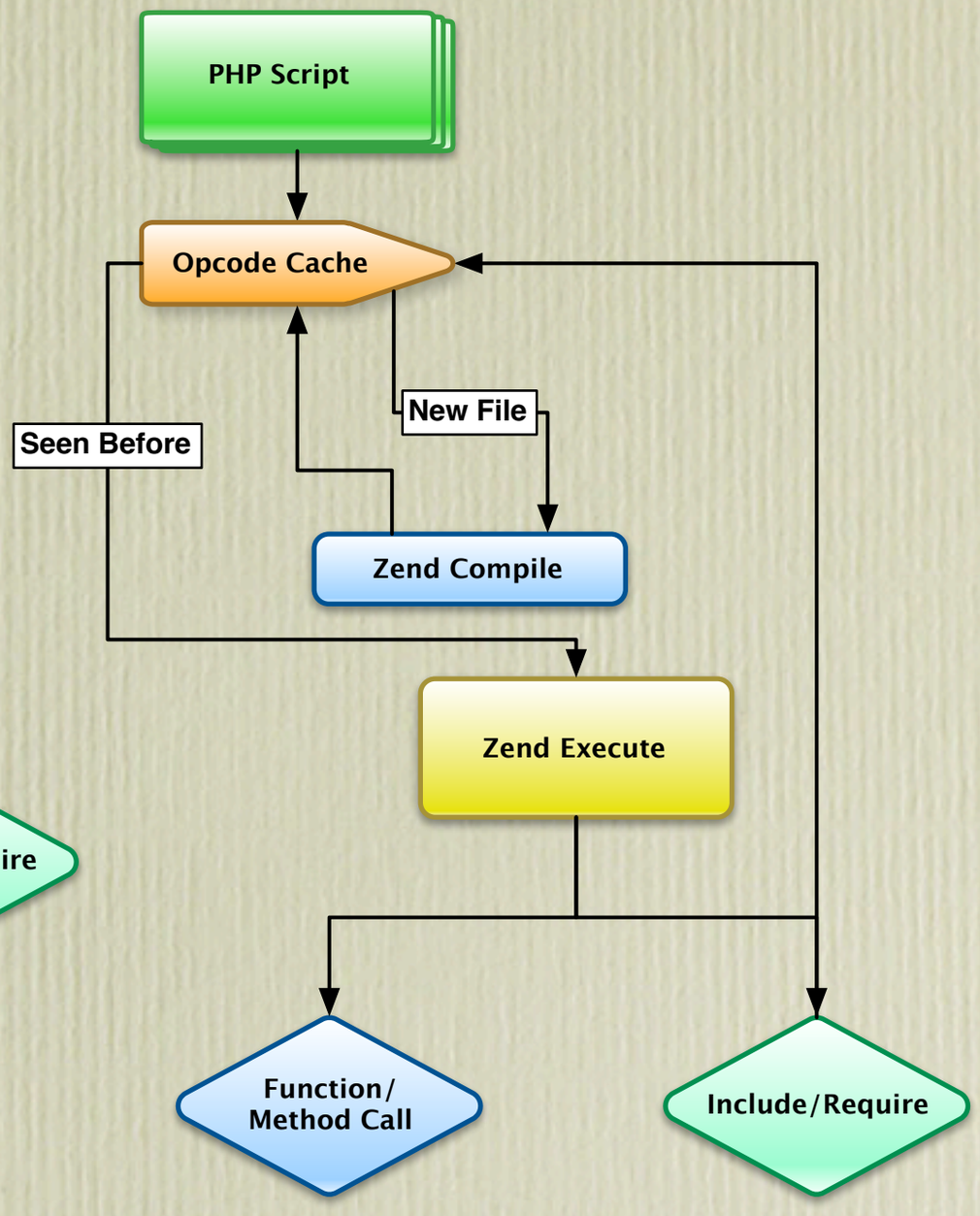
- Alternative PHP Cache
- Primarily designed to accelerate script performance via opcode caching
- Extends opcode caching to facilitate user-data caching
- Actively maintained & well supported

Opcode Caching

Normal Execution



With APC



Opcode Caching Hot Sheet

- Improves script performance by 40-300%
- Substantially reduces file IO (makes other things faster)
- Leaves option for optimization, fine tuned opcodes for faster execution.

APC User-Cache

- Allows you to apply the same caching logic to your data as applied to PHP scripts.

SLIDE MOTTO:

NOT EVERYTHING HAS TO BE REAL-TIME!

APC Interface

- Very simple, you only need to remember 4 functions, and one you can ignore if you are lazy.

```
bool apc_store ( string $key , mixed $var [, int $ttl = 0 ] )

bool apc_add ( string $key , mixed $var [, int $ttl = 0 ] )

mixed apc_fetch ( mixed $key [, bool &$success ] )

bool apc_delete ( string $key )
```

APC in Practice

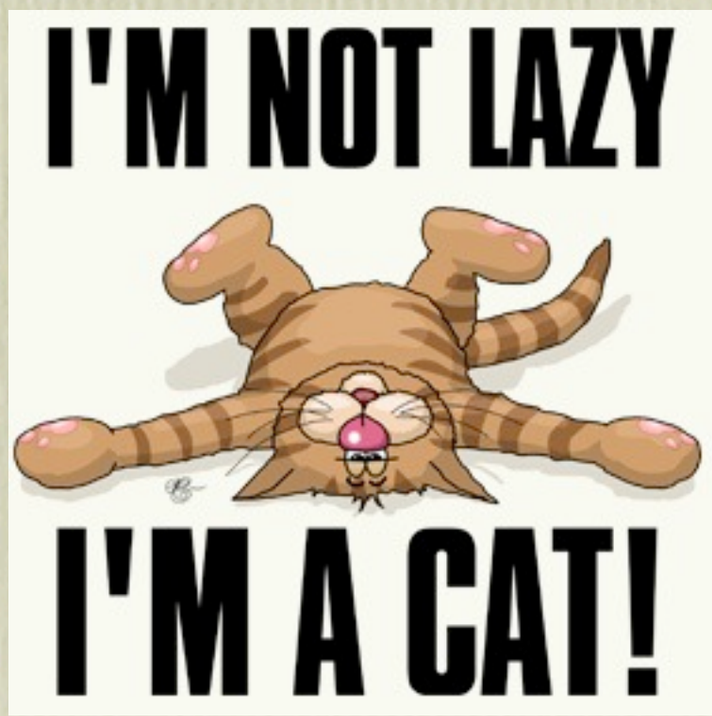
```
// store an array of values for 1 day, referenced by "identifier"
if (!apc_add("identifier", array(1,2,3), 86400)) {
    // already exists? let's update it instead
    if (!apc_store("identifier", array(1,2,3), 86400)) {
        // uh, oh, b0rkage
        mail("gopal, brian, kalle", "you broke my code", "fix it!");
    }
}

$ok = null;
// fetch value associated with "identified" and
// put success state into $ok variable
$my_array = apc_fetch("identifier", $ok);
if ($ok) {
    // changed my mind, let's delete it
    apc_delete("identifier");
}
```

Let's be lazy

```
// create or add an array of values for 1 day
if (!apc_store("identifier", array(1,2,3), 86400)) {
   // uh, oh, b0rkage
   mail("gopal, brian, kalle", "you broke my code", "fix it!");
}
```

- If you don't care whether your are adding or updating values you can just use **`apc_store()`** and keep your code simpler



Don't Delete

- Deleting from cache is expensive as it may need to re-structure internal hash tables.

  - Rely on auto-expiry functionality instead
  - Or an off-stream cron job to clean up stale cache entries

- In many cases it is simpler just to start from scratch.

apc_clear_cache("user")

Installing APC

Unix

sudo bash (open root shell)

pecl install apc (configure, compile & install APC)

Windows

Copy the php_apc.dll file into your php's ext/ directory

Common

Enable APC from your php.ini file

APC Configuration

```
apc.enable # enable APC
apc.enable_cli # enable for CLI sapi
apc.max_file_size # max PHP file to be cached
apc.stat # turn off ONLY if your files never change
apc.file_update_protection # good idea if you edit files on
live environment

apc.filters # posix (ereg) expressions to filter out files
from being cached

apc.mmap_file_mask # /tmp/apc.XXXXXX (use mmap IO, USE IT!)
apc.shm_segments # if not using mmap IO, otherwise 1
apc.shm_size # how much memory to use
```

Make PHBs Happy

Refresh Data | View Host Stats | System Cache Entries | User Cache Entries | Version Check

General Cache Information

APC Version	3.0.19
PHP Version	5.2.10-dev
APC Host	ilia.ws (penguin) (216.235.13.26)
Server Software	Apache/2.2.11 (Unix) mod_ssl/2.2.11 OpenSSL/0.9.8i PHP/5.2.10-dev
Shared Memory	1 Segment(s) with 150.0 MBytes (mmap memory, spin locking)
Start Time	2009/10/12 04:02:34
Uptime	3 days, 7 hours and 10 minutes
File Upload Support	1

File Cache Information

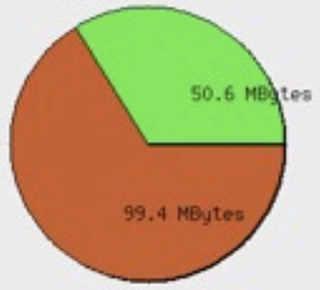
Cached Files	2670 (86.4 MBytes)
Hits	6108114
Misses	161050
Request Rate (hits, misses)	22.00 cache requests/second
Hit Rate	21.43 cache requests/second
Miss Rate	0.57 cache requests/second
Insert Rate	0.56 cache requests/second
Cache full count	31

User Cache Information

Cached Variables	25 (300.6 KBytes)
Hits	177582
Misses	0
Request Rate (hits, misses)	0.62 cache requests/second
Hit Rate	0.62 cache requests/second
Miss Rate	0.00 cache requests/second
Insert Rate	0.01 cache requests/second
Cache full count	31

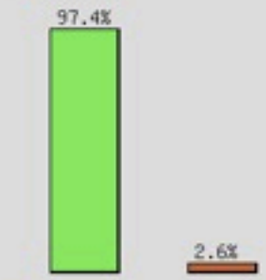
Host Status Diagrams

Memory Usage



Free: 50.6 MBytes (33.7%)
Used: 99.4 MBytes (66.3%)

Hits & Misses



Hits: 6108114 (97.4%)
Misses: 161050 (2.6%)

Detailed Memory Usage and Fragmentation

Fragmentation: 0%

* Pretty graphics require GD extension

- For raw data you can use the apc_cache_info() and apc_sma_info() functions.

Advantages of APC

- If you (or your ISP) uses opcode caching, chances are it is already there.
- Really efficient at storing simple types (scalars & arrays)
- Really simple to use, can't get any easier...
- Fairly stable

APC Limitations

- PHP only, can't talk to other "stuff"
- Not distributed, local only
- Opcode + User cache == all eggs in one basket
- Could be volatile

Memcache

- Interface to Memcached - a distributed caching system
- Provides Object Oriented & Procedural interface to caching system
- Offers a built-in session handler
- Can only be used for “user” caching
- Purpose built, so lots of nitfy features

Procedural Interface

object memcache_connect (string $host [, int $port [, int $timeout]])

bool memcache_add (object $mem, string $key , mixed $var [, int $flag [, int $expire]])

mixed memcache_get (object $mem, string $key [, int &$flags])

bool memcache_set (object $mem, string $key , mixed $var [, int $flag [, int $expire]])

bool memcache_delete (object $mem, string $key [, int $timeout])

bool memcache_replace (object $mem, string $key, mixed $var [, int $flag [, int $expire]])

For OO fans

bool Memcache::connect (string $host [, int $port [, int $timeout]])

bool Memcache::add (string $key , mixed $var [, int $flag [, int $expire]])

mixed Memcache::get (string $key [, int &$flags])

bool Memcache::set (string $key , mixed $var [, int $flag [, int $expire]])

bool Memcache::delete (string $key [, int $timeout])

bool Memcache::replace (string $key , mixed $var [, int $flag [, int $expire]])

The Basics

```php
$mc = new memCache();
// connect to memcache on local machine, on default port
// with 1 second timeout
$mc->connect('localhost', '11211', 1);

// try to add an array with a retrieval key for 1 day
if ($mc->add('key', array(1,2,3), NULL, 86400)) {
    // if already exists, let's replace it
    if ($mc->replace('key', array(1,2,3), NULL, 86400)) {
        mail("tony", "no workie", ":-(");
    }
}

// let's fetch our data
if (($data = $mc->get('key')) !== FALSE) {
    // let's delete it now
    $mc->delete('key', 0); // RIGHT NOW!
}

// drop connection to server
$mc->close();
```

Fancy Stuff

```php
$mc = new memCache();
// on local machine we can connect via Unix Sockets for better speed
$mc->connect('unix:///var/run/memcached/11211.sock', 0);

// add/or replace, don't care just get it in there
// also user the default expire time of 30 days
$mc->set('key1', array(1,2,3));

// for long values you can enable compression to save RAM
$mc->set('key2', range(1,10000), MEMCACHE_COMPRESSED);

// get multiple keys at once
$data = $mc->get(array('key1', 'key2'));
/*
array(
    'key1' => ...
    'key2' => ...
)
*/
```

Failover, Failover, Failover

```php
$mc = new memCache();

// main server via a persistent connection, heavily weighted
$mc->addServer('unix:///var/run/memcached/11211.sock', 0, 1, 100);

// secondary server, retry every 10 seconds on error (15 by default)
$mc->addServer('secondary', 11211, 1, 50, 10);

// add backup, if backup is offline, then
call the "error_callback" function
$mc->addServer('backup', 11211, 1, 10, 10, TRUE, 'error_callback');
```


```php
// add dummy server, that is marked offline
$mc->addServer('offline', 11211, 0, -1, 1, FALSE);
```

The Neat “Stuff”

```
$mc = new memCache();
$mc->addServer('unix:///var/run/memcached/11211.sock', 0);

// automatically compress values longer then 10000 bytes
// but only do so if compression achieves at least a 30% saving
$mc->setCompressThreshold(10000, 0.3);
```

Atomic Counters

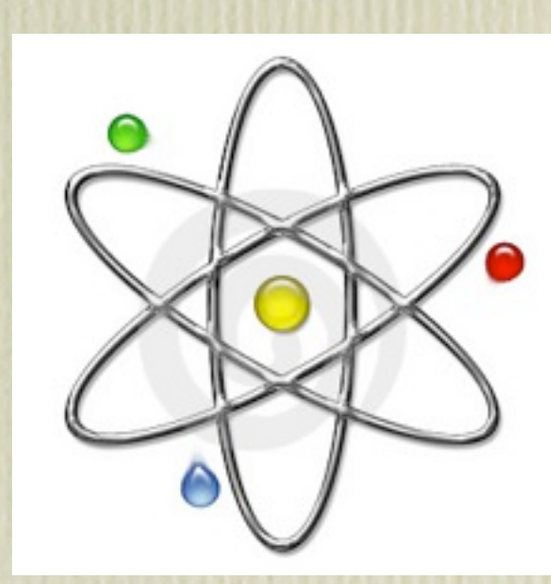
```php
$mc = new memCache();
$mc->addServer('unix:///var/run/memcached/11211.sock', 0);

// initialize counter to 1
$mc->set('my_counter', 1);

// increase count by 3
$mc->increment('my_counter', 3);

$mc->get('my_counter'); // 4

// decrement count by 2
$mc->decrement('my_counter', 2);
```



Counter Trick

```php
$mc = new memCache();
$mc->addServer('unix:///var/run/memcached/11211.sock', 0);

// add key position if does not already exist
if (!$mc->add('key_pos', 1)) {
    // otherwise increment it
    $position = $mc->increment('key_pos', 1);
} else {
    $position = 1;
}

// add real value at the new position
$mc->add('key_value_' . $position, array(1,2,3));
```

- Simplifies cache invalidation
- Reduces lock contention (or eliminates it)

Installing Memcache

Unix

sudo bash (become root user)

Download memcached from http://www.danga.com/memcached/ and compile it.

pecl install memcache (configure, compile & install memcache)

Windows - Upgrade to *NIX

Common

Enable Memcache from your php.ini file

Memcache INI Settings

```
memcache.allow_failover # Allow failover

memcache.chunk_size # increase for big data (8k default)

memcache.compress_threshold # compress data over this size

memcache.max_failover_attempts # total failover retries


session.save_handler # set to “memcache” to use memcache for
session handling

session.save_path # set to memcache host tcp://192.168.1.1:11211
```

Can make PHP Happy Too

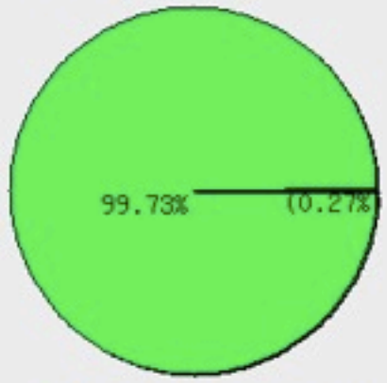
Refresh Data | View Host Stats | Variables

General Cache Information	
PHP Version	5.2.10-pl0-gentoo
Memcached Host	1.unix:///var/run/memcached/11211.sock
Total Memcache Cache	64.0 MBytes

Memcache Server Information	
unix:///var/run/memcached/11211.sock	[Flush this server]
Start Time	2009/09/04 14:29:50
Uptime	5 weeks, 6 days, 19 hours and 4 minutes
Memcached Server Version	1.3.3
Used Cache Size	175.6 KBytes
Total Cache Size	64.0 MBytes

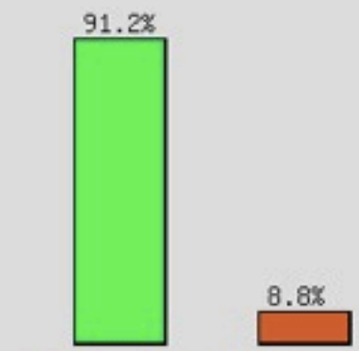
Host Status Diagrams

Cache Usage



Free: 63.8 MBytes (99.7%)
Used: 175.6 KBytes (0.3%)

Hits & Misses



Hits: 6656 (91.2%)
Misses: 642 (8.8%)

Cache Information	
Current Items(total)	638 (640)
Hits	6656
Misses	642
Request Rate (hits, misses)	0.00 cache requests/second
Hit Rate	0.00 cache requests/second
Miss Rate	0.00 cache requests/second
Set Rate	0.00 cache requests/second

* Pretty graphics require GD extension

Advantages of Memcache

- Allows other languages to talk to it
- One instance can be shared by multiple servers
- Failover & redundancy
- Nifty Features
- Very stable

It is not perfect because?

- Slower then APC, especially for array storage
- Requires external daemon
- You can forget about it on shared hosting

That's all folks

Any Questions?

Slides at: **http://ilia.ws**

Comments: **http://joind.in/911**