APC & Memcache the High Performance Duo

Dutch PHP Conference 2010

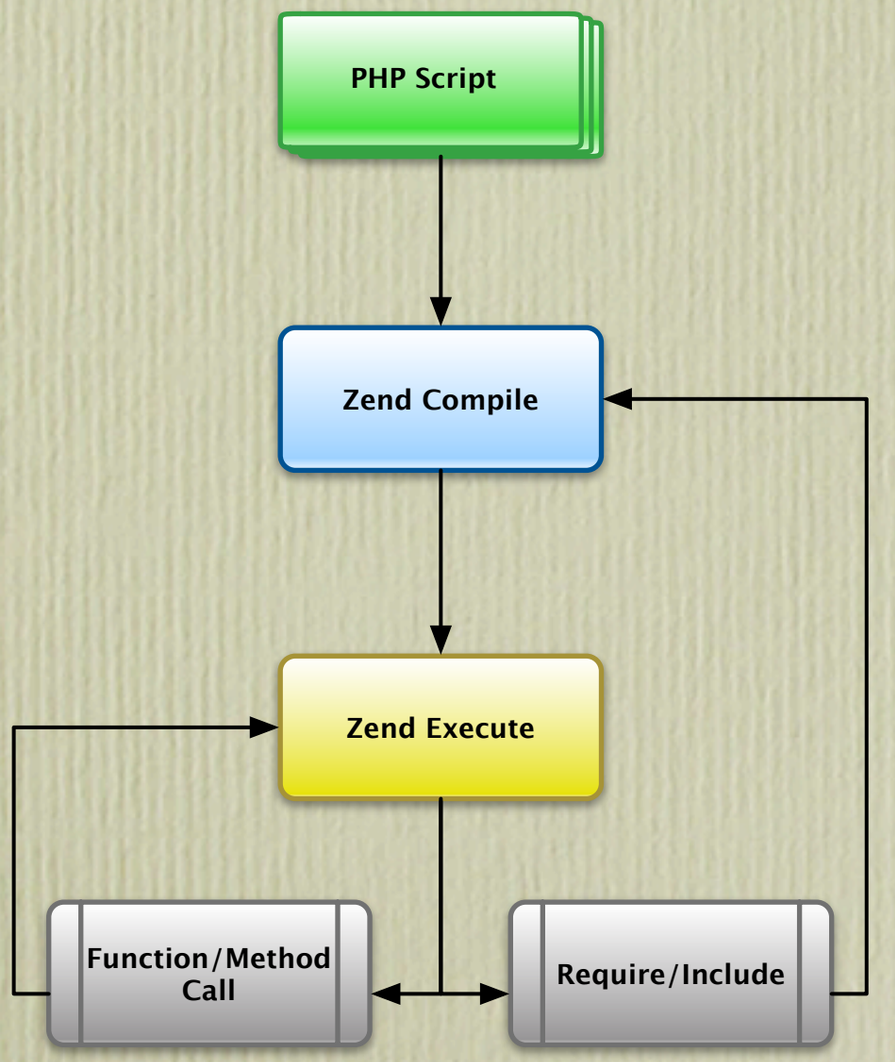
Ilia Alshanetsky
http://ilia.ws/

What is APC?

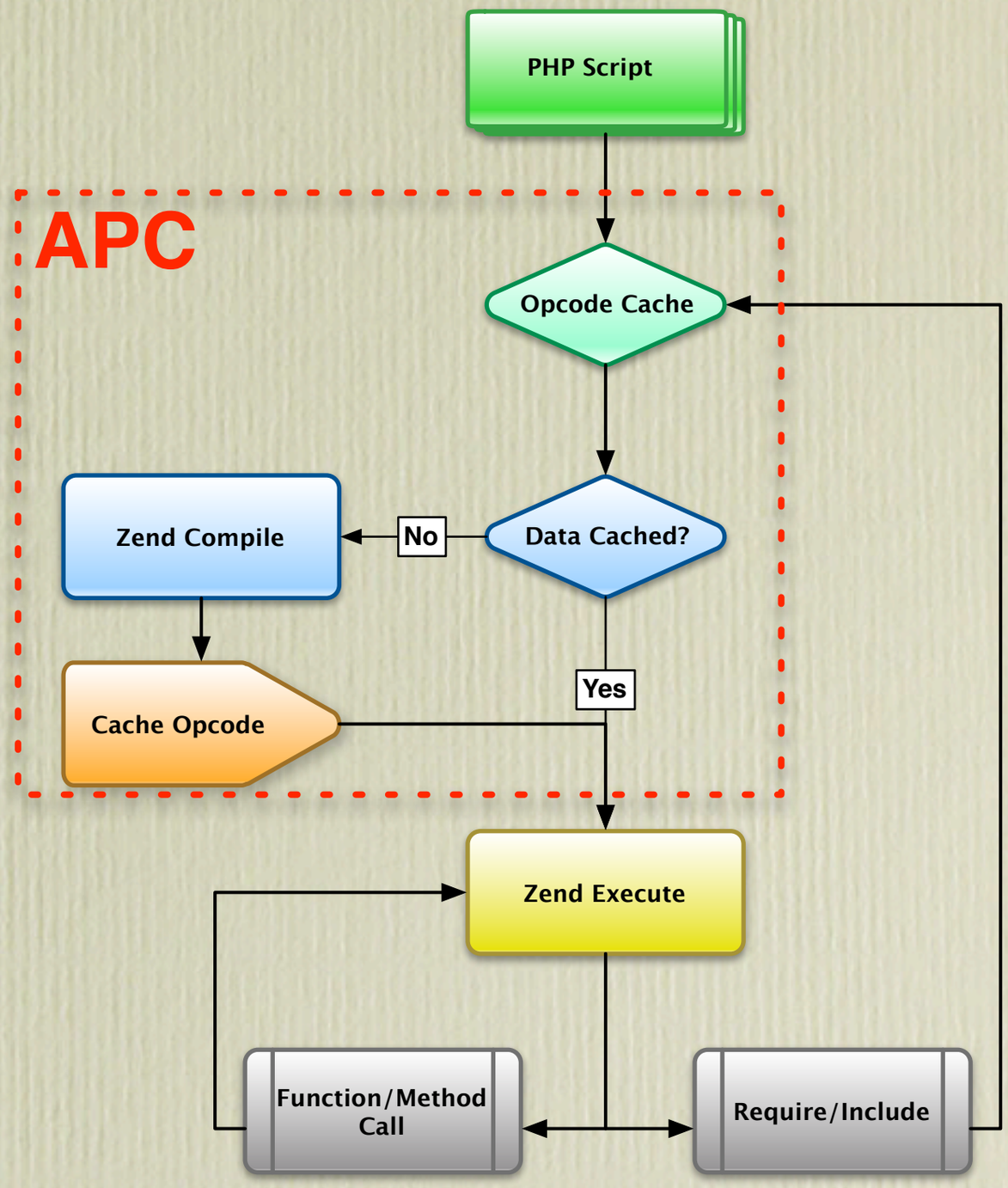
- Alternative PHP Cache
- Primarily designed to accelerate script performance via opcode caching
- Extends opcode caching to facilitate user-data caching
- Actively maintained & well supported

Opcode Caching

Default Mode



With APC



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 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
    }
}

$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 update 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

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

Memcached

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

Memcache vs Memcached

- Memcached Advantages
  - Faster
    - Igbinary serializer
    - fastlz compression
  - Multi-Server Interface
  - Fail-over callback support

Basics in Practice

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

// connect to memcache on local machine, on default port
$mc->addServer('localhost', '11211');

// try to add an array with a retrieval key for 1 day
if (!$mc->add('key', array(1,2,3), 86400)) {
    // if already exists, let's replace it
    if ($mc->replace('key', array(1,2,3), 86400)) {
        die("Critical Error");
    }
}

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

Data Retrieval Gotcha(s)

```php
$mc = new MemCached();
$mc->addServer('localhost', '11211');

$mc->add('key', 0);

if (!($data = $mc->get('key'))) {
  die("Not Found?"); // not true
  // The value could be 0,array(),NULL,""
  // always compare Memcache::get() result to
  // FALSE constant in a type-sensitive way (!== FALSE)
}

// The “right” way!
if (($data = $mc->get('key')) !== FALSE) {
  die("Not Found");
}
```

Data Retrieval Gotcha(s)

```php
$mc = new MemCached();
$mc->addServer('localhost', '11211');

$mc->add('key', FALSE);

if (($data = $mc->get('key')) !== FALSE) {
  die("Not Found?"); // not true
  // The value could be FALSE, you
  // need to check the response code
}

// The “right” way!
if (
        (($data = $mc->get('key')) !== FALSE)
        &&
        ($mc->getResultCode() != MemCached::RES_SUCCESS)
) {
  die("Not Found");
}
```

Interface Basics Continued...

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

// add/or replace, don't care just get it in there
// without expiration parameter, will remain in cache “forever”
$mc->set('key1', array(1,2,3));

$key_set = array('key1' => “foo”, 'key1' => array(1,2,3));

// store multiple keys at once for 1 hour
$mc->setMulti($key_set, 3600);

// get multiple keys at once
$data = $mc->getMulti(array_keys($key_set));
/*
array(
    'key1' => ‘foo’
    'key2' => array(1,2,3)
)
*/
```

For multi-(get|set), all ops must succeed for successful return.

Multi-Server Environment

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

// add multiple servers to the list
// as many servers as you like can be added
$mc->addServers(
      array('localhost', 11211, 80), // high-priority 80%
      array('192.168.1.90', 11211, 20)// low-priority 20%
);

// You can also do it one at a time, but this is not recommended
$mc->addServer('localhost', 11211, 80);
$mc->addServer('192.168.1.90', 11211, 20);

// Get a list of servers in the pool
$mc-> getServerList();
// array(array('host' => … , 'port' => … 'weight' => …))
```

Data Segmentation

- Memcached interface allows you to store certain types of data on specific servers

```
$mc = new MemCached();
$mc->addServers( … );

// Add data_key with a value of “value” for 10 mins to server
// identified by “server_key”
$mc->addByKey('server_key', 'data_key', 'value', 600);

// Fetch key from specific server
$mc->getByKey('server_key', 'data_key');

// Add/update key on specific server
$mc->setByKey('server_key', 'data_key', 'value', 600);

// Remove key from specific server
$mc->deleteByKey('server_key', 'data_key');
```

And there is more ...

- The specific-server interface also supports multi-(get|set)

```
$mc = new MemCached();
$mc->addServers( … );
```

```
$key_set = array('key1' => “foo”, 'key1' => array(1,2,3));

// store multiple keys at once for 1 hour
$mc->setMultiByKey('server_key', $key_set, 3600);

// get multiple keys at once
$data = $mc->getMultiByKey('server_key', array_keys($key_set));
```

Delayed Data Retrieval

- One of the really neat features of Memcached extension is the ability to execute the “fetch” command, but defer the actual data retrieval until later.

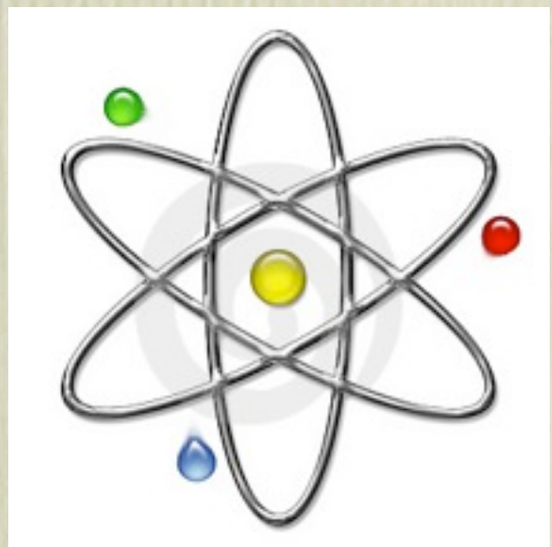
- Particularly handy when retrieving many keys that won’t be needed until later.

Delayed Data Retrieval

```php
$mc = new MemCached();
$mc->addServer('localhost', '11211');

$mc->getDelayed(array('key')); // parameter is an array of keys

/* some PHP code that does “stuff” */

// Fetch data one record at a time
while ($data = $mc->fetch()) { ... }

// Fetch all data in one go
$data = $mc->fetchAll();
```

Atomic Counters

```php
$mc = new MemCached();
$mc->addServer('localhost', 11211);

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

// increase count by 1
$mc->increment('my_counter');

// increase count by 10
$mc->increment('my_counter', 10);

// decrement count by 1
$mc->decrement('my_counter');

// decrement count by 10
$mc->decrement('my_counter', 10);
```



Counter Trick

```php
$mc = new MemCached();
$mc->addServer('localhost', 11211);

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

Data Compression

- In many cases performance can be gained by compressing large blocks of data. Since in most cases network IO is more expensive then CPU speed + RAM.

```
$mc = new MemCached();
$mc->addServer('localhost', 11211);
// enable compression
$mc->setOption(Memcached::OPT_COMPRESSION, TRUE);

        Related INI settings (INI_ALL)

        Other possible value is zlib
        memcached.compression_type=fastlz

        minimum compression rate
        memcached.compression_factor=1.3

        minimum data size to compress
        memcached.compression_threshold=2000
```

PHP Serialization

If you are using memcached to store complex data type (arrays & objects), they will need to be converted to strings for the purposes of storage, via serialization.

Memcached can make use of igbinary serializer that works faster (~30%) and produces more compact data set (up-to 45% smaller) than native PHP serializer.

http://github.com/phadej/igbinary

Enabling Igbinary

Install Memcached extension with

--enable-memcached-igbinary

```
$mc = new MemCached();
$mc->addServer('localhost', 11211);

// use Igbinary serializer
$mc->setOption(
    Memcached::OPT_SERIALIZER,
    Memcached::SERIALIZER_IGBINARY
);
```

Utility Methods

```php
$mc = new MemCached();
$mc->addServer('localhost', 11211);

// memcached statistics gathering
$mc->getStats();

// clear all cache entries
$mc->flush();

// clear all cache entries
// in 10 minutes
$mc->flush(600);
```

```
Array
(
    [server:port] => Array
        (
            [pid] => 4933
            [uptime] => 786123
            [threads] => 1
            [time] => 1233868010
            [pointer_size] => 32
            [rusage_user_seconds] => 0
            [rusage_user_microseconds] => 140000
            [rusage_system_seconds] => 23
            [rusage_system_microseconds] => 210000
            [curr_items] => 145
            [total_items] => 2374
            [limit_maxbytes] => 67108864
            [curr_connections] => 2
            [total_connections] => 151
            [a] => 3
            [bytes] => 20345
            [cmd_get] => 213343
            [cmd_set] => 2381
            [get_hits] => 204223
            [get_misses] => 9120
            [evictions] => 0
            [bytes_read] => 9092476
            [bytes_written] => 15420512
            [version] => 1.2.6
        )
)
```

Installing Memcached

Download memcached from http://www.memcached.org and compile it.

Download libmemcached from http://tangent.org/552/libmemcached.html and compile it.

pecl install memcached (configure, make, make install)

Enable Memcached from your php.ini file

Memcached Session Handler

```
# Session settings

session.save_handler # set to “memcached

session.save_path # set to memcache host server:port

memcached.sess_prefix # Defaults to memc.sess.key.

# Locking Controls

# Whether to enable session lock, on by default
memcached.sess_locking

# Maximum number of microseconds to wait on a lock
memcached.sess_lock_wait
```

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/1554

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

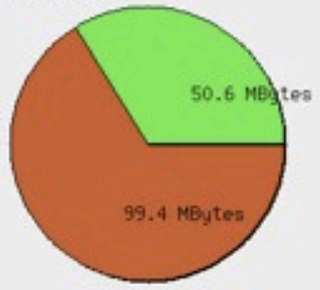
Cached Files	2670 (86.4 MBytes)
Hits	6108114
Misses	161050
Request Rate (hits, misses)	22.00 cache requests/second
Hit Rate	21.43 cache requests/second
Miss Rate	0.57 cache requests/second
Insert Rate	0.56 cache requests/second
Cache full count	31

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

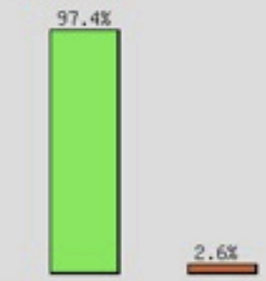
Host Status Diagrams

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.