Memcached, the Better Memcache Interface

ZendCon 2010

Ilia Alshanetsky

@iliaa

Memcached

- Interface to Memcached - a distributed caching system
- Provides Object Oriented interface to caching system
- Offers a built-in session handler
- Purpose built, so lots of nifty features

Memcache vs Memcached

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

Basics in Practice

```
$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)

```
$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 Memcached::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)

```
$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...

```
$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

```
$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", 'key2' => 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));
```

Fail-Over Callbacks

```
$m = new Memcached();
$m->addServer('localhost', 11211);

$data = $m->get('key', 'cb');

function cb(Memcached $memc, $key, &$value) {
    $value = 'retrieve value';
    $memc->add($key, $value);
    return $value;
}
```

Only supported for get() & getByKey() methods

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

```
$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();
```

Delayed Result C.B.

- The delayed result callback allows execution of code upon successful delayed retrieval.

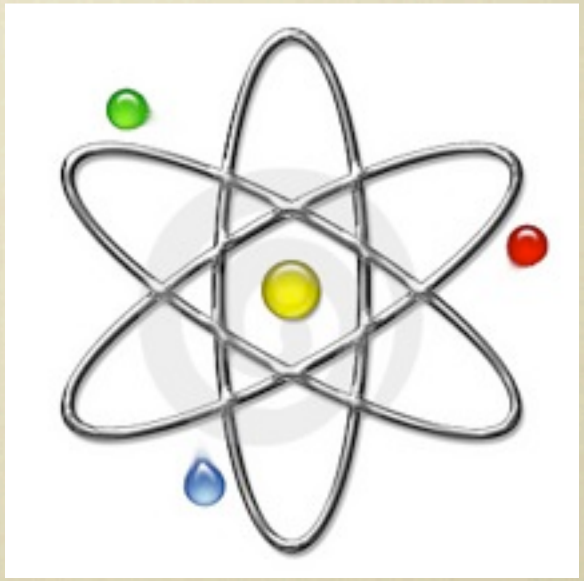
Example

```
$m = new Memcached();
$m->addServer('localhost', 11211);

$m->getDelayed(
   array('footer','header'), false, 'cb');

function cb(Memcached $m, $data) {
   //$data = array('key' => '...', 'value' => '...');
   layout::$data['key']($data['value']);
}
```

Callback will be called individually for every key

Atomic Counters

```
$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

```
$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.

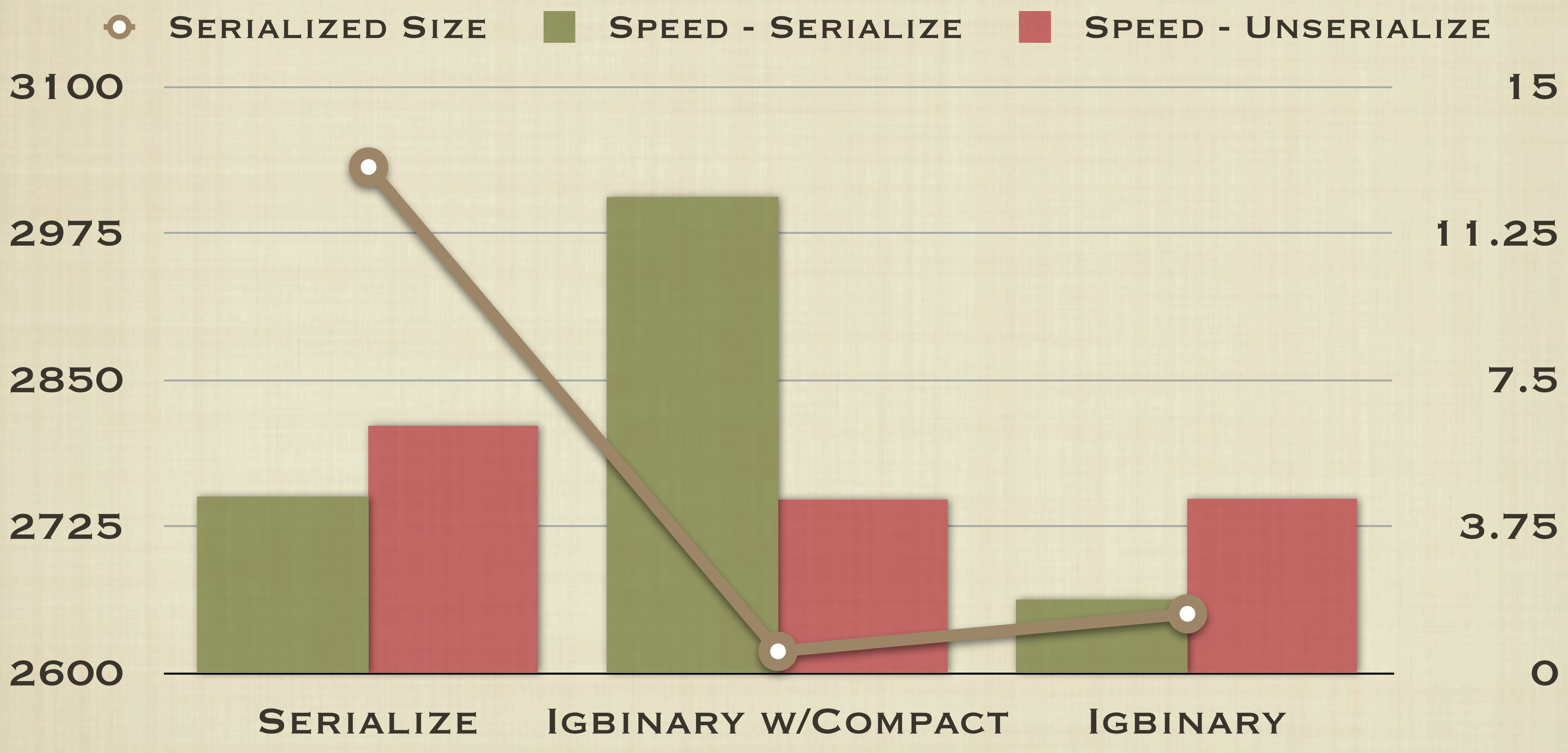
http://github.com/phadej/igbinary

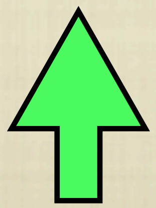
Enabling Igbinary

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
);
```

Igbinary speed test





Optimal

Utility Methods

```
$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

Installing Memcached

If you want the latest Memcached sources checkout github:

- http://github.com/andreiz/php-memcached
- http://github.com/tricky/php-memcached
- http://github.com/iliaal/php-memcached
- (and a bunch of others)

Memcached Session Handler

# Session settings

session.save_handler # set to “memcached

session.save_path # set to memcache host server:port

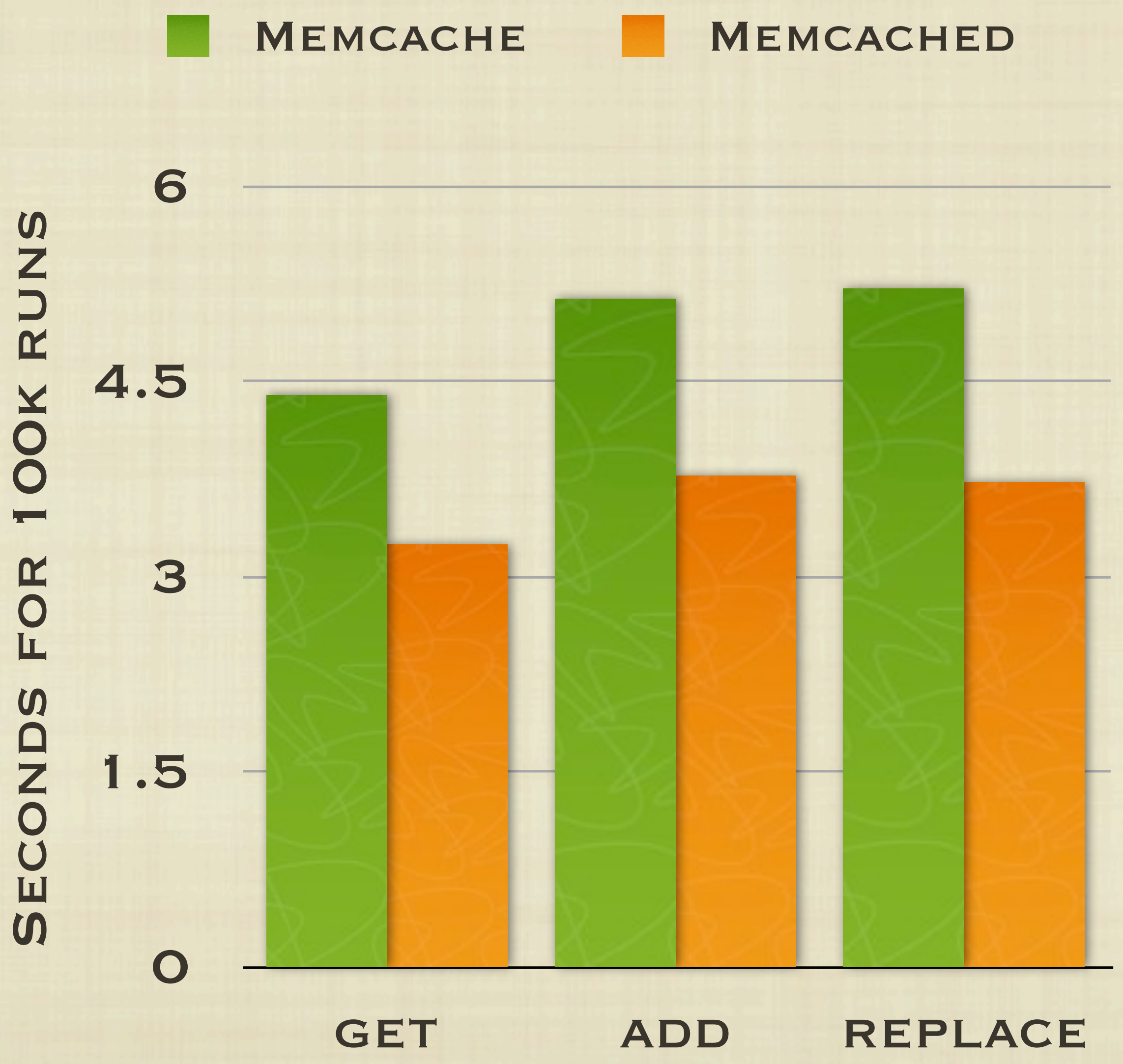
memcached.sess_prefix # Defaults to memc.sess.key.

# 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

Performance



Thank You For Listening

Slides will be available at http://ilia.ws

Please give me your feedback http://joind.in/2250