Writing distributed applications with PHP

Andrey Hristov
100 DAYS – Software Projects

International PHP Conference 2004
Nov 10th, 2004

About me

- BSc in Computer Engineering
- Student at University of Applied Sciences, Stuttgart, Germany
- Practicing web programming since year 2000
- Author of pecl/stats
- Working for 100 Days, Germany

Survey

Have you used the following extensions with PHP :

- CORBA
- Java
- VL-SRM

Agenda

- CORBA
- Java
- COM
- VL-SRM
- Custom built solution

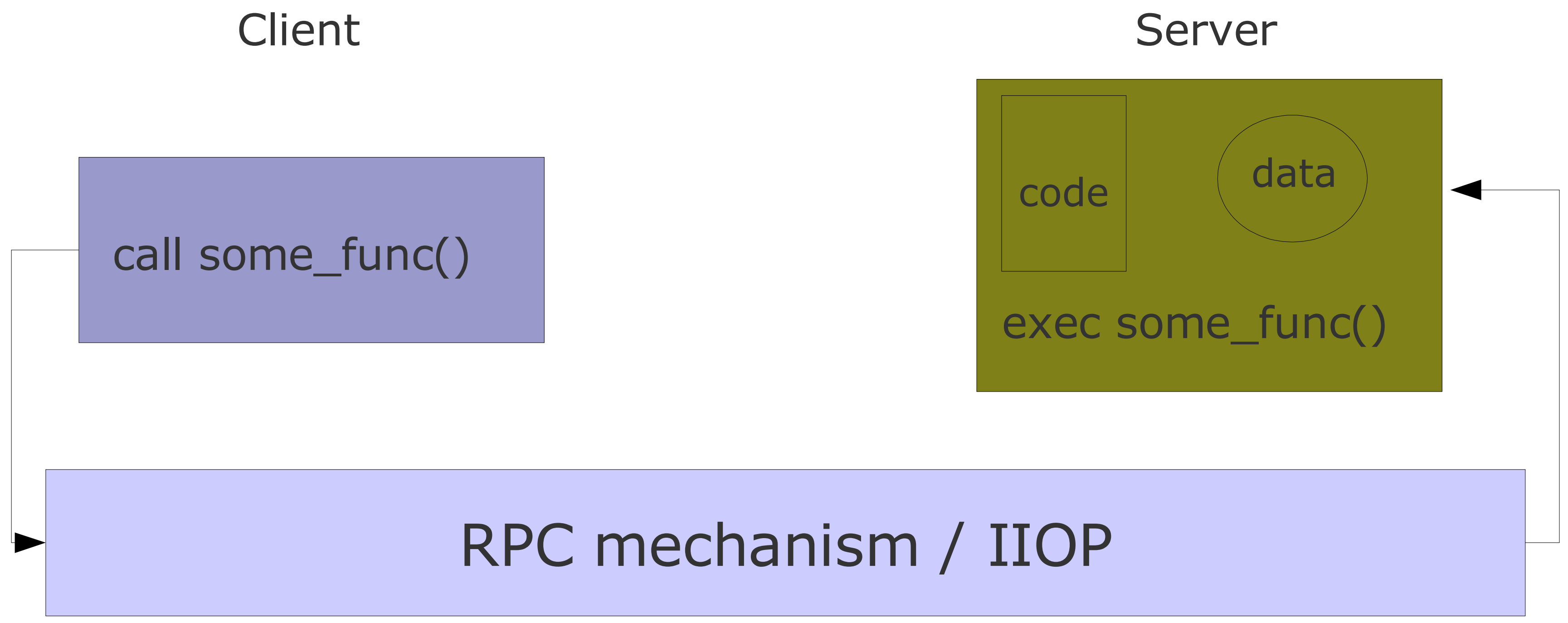
CORBA – introduction (1)

- Common Object Request Broker Architecture. Current revision 3.0.2 .
- Standard by OMG (Object Management Group). OMG is industrial consortium, about 500 members (2003). The same guys that push forward UML and MDA.
- High interoperability between different programming languages. Bindings exist for many languages (C, C++, Pascal, Java, COBOL, Perl, Python, PHP, Smalltalk etc.).
- CORBA aims building homogeneous applications on top of heterogeneous systems.

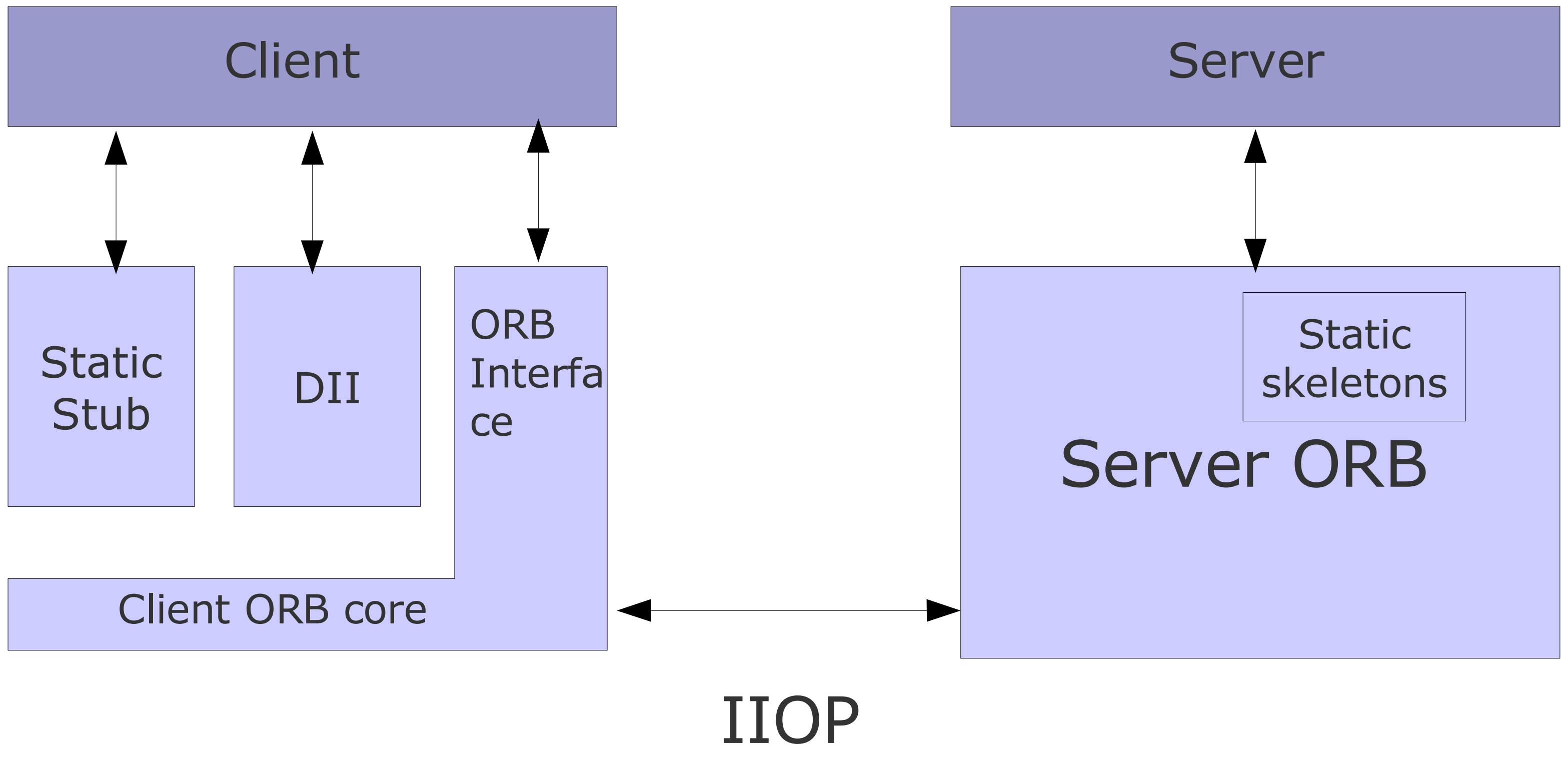
CORBA – introduction (2)

- CORBA helps integrating legacy code. Typical example is interfacing existing COBOL code.
- Basic terms :
  - CORBA-Object : an unit which has associated interface and servant (implementation).
  - Request : call of an operation of a specific object and return of result if available. Types are : synchron, deferred synchron (async) and oneway.
  - Client : An unit that makes requests.
  - Server : the place where objects live. Client and server are always relative to specific request!
  - IOR : Interoperable Object Reference

CORBA – simple model



CORBA - architecture



CORBA - IDL

- IDL stands for Interface Definition Language.
- Similar to C and influenced by it. Has preprocessor.
- Used to write the interface between the client and the server.
- In most cases the IDL file is compiled to *stubs* and *skeletons*.
- Stubs are used by the client (Proxy pattern).
- Skeletons implement the interface but contain no code. The programmer “fills” them with code.
- Supports following types: short, long, uint, ushort, float, double, char, string, boolean, octet, any (container), arrays (fixed), sequences (dynamic), structs, objects (CORBA!).
- Other elements: consts, interfaces, modules

CORBA – IDL example

```
module SuperDuperMod {
    typedef sequence<string> List;
    // Dictionary interface
    interface Dictionary {
        // store a key value pair into the dictionary.
        void put(in string key, in string val);
        // return the value of a key
        string get(in string key);
        // get more than one value at time
        List getList(in List keys);
    };
    // Counter interface
    interface Counter {
        void inc();
        void incByNumber(in long num);
        long get();
    };
};
```

PHP extensions for CORBA

- Satellite - deprecated.
- Universe – the successor of Satellite. Uses Mico ORB. No further development for 2 years. Clients and servers are possible in PHP.
- PHP-Orbit – New implementation. Last version from July'04. Only clients are possible. Uses ORBit (Gnome project's ORB).

PHP-Orbit example (client)

```
<?php
    dl('/usr/local/src/php-orbit-0.1.2-pre1/modules/orbit.so');
    function exception_handler($arg) {
        global $exception, $exception_self;
        if (substr($arg->_class, 0, 6) == "CORBA." && !$exception_self)
           die("\nCritical CORBA exception : {$arg->text} ({$arg->status})\n");
        else $exception = $arg;
    }

    orbit_exception_handler('exception_handler');
    orbit_load_idl('super_duper.idl');
    $dict = new CORBA(file_get_contents('dict.ref'));
    $dict->put("one", "ein");
    $dict->put("two", "zwei");
    $dict->put("three","drei");
    echo “Let's see what we have in the dictionary :\n”;
    var_dump($dict->get("two"));
    var_dump($dict->getList(array("two", "three")));

    $counter = new CORBA(file_get_contents('counter.ref'));
    echo “Let's do some counting :\n”;
    var_dump($counter->get(), $counter->inc(), $counter->get());
    var_dump($counter->incByNumber(14), $counter->get());
    print "Finished work\n";
?>
```

The server (CORBA::ORBIT)

```perl
#!/usr/bin/perl -w
use CORBA::ORBit idl =>[qw(super_duper.idl)];
use strict;
package Dictionary;
use base qw(POA_SuperDuperMod::Dictionary);
sub new {
  my $type = shift;
  $type = ref($type) || $type;
  my $self = {}; $self->{dict} = {};
  bless($self,$type);
  return $self;
}
sub put { my ($self,$key,$val) = @_;
  $self->{dict}->{$key} = $val;}
sub get { my ($self,$key) = @_;
  return $self->{dict}->{$key};}
sub getList {my ($self,$list) = @_;
  return [map {$self->{dict}->{$_}}@$list];
}
package Counter;
use base qw(POA_SuperDuperMod::Counter);
sub new {
  my $type = shift;$type = ref($type)||$type;
  my $self = {};$self->{cnt} = 0;
  bless($self,$type); return $self;}
sub get {my ($self)= @_;return $self->{cnt};}
sub inc {my ($self)= @_;$self->{cnt}++;}
sub incByNumber{
  my ($self,$n)= @_;$self->{cnt} += $n;}
```

```perl
package main;
use Error qw(:try);
#get our ORB
my $orb = CORBA::ORB_init("orbit-local-orb");
#get our POA so we can act as a server
my $poa =
  $orb->resolve_initial_references("RootPOA");

my $servD = new Dictionary();
my $servC = new Counter();

#Activate our objects with CORBA
my $idD = $poa->activate_object ($servD);
my $idC = $poa->activate_object($servC);
#get a corba object reference
my $refD = $orb->object_to_string (
  $poa->id_to_reference ($idD));
my $refC = $orb->object_to_string(
  $poa->id_to_reference ($idC));

open (OUT1, ">d.ref");open (OUT2, ">c.ref");
print OUT1 $refD;print OUT2 $refC;
close OUT1; close OUT2;

#start the poa manager
$poa->_get_the_POAManager->activate;
$orb->run ();
#never reached
exit(0);
```

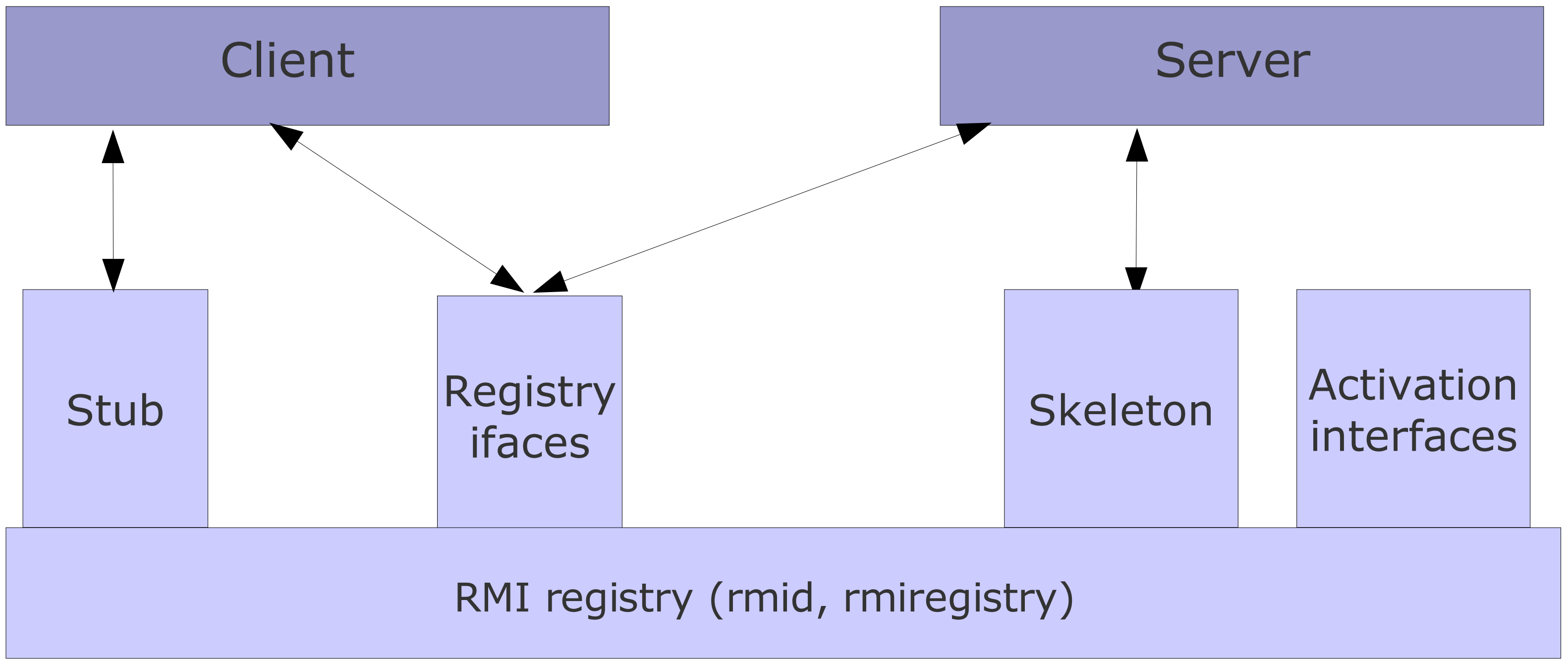
CORBA example output

```
andrey@vivaldi:/usr/local/src/php-orbit-0.1.2-pre1/test> php client_simple.php
Let's see what we have in the dictionary :
string(4) "zwei"
array(2) {
  [0]=>
  string(4) "zwei"
  [1]=>
  string(4) "drei"
}
Let's do some counting :
int(0)
NULL
int(1)
NULL
int(15)
Finished work
```

Java technologies

- PHP has an extension to interact with Java.
- This extension gives access to all Java APIs (calls are only by values).
- Another bridge is hosted at SF.net .
- Java APIs related to distributed programming are RMI and JMS.
- RMI is available since JDK 1.1, impoved in 1.2, supports IIOP since 1.3 (interoperability between CORBA and RMI). The protocol name is RMI-IIOP.
- The protocol is binary thus giving performance advantage.

RMI - architecture



Installing PHP-Java bridge (1)

- Download from SF.net (see resources).
- Be sure you have PHP 4 headers installed. PHP 5 will not work according to the author of the extension.
- phpize && make && make install
- *java.so*, *JavaBridge.class*, *libnatcJavaBridge.so* and also static flavours are somewhere in /usr/local/bin/php/extensions/....
- Put in your php.ini instruction for loading the ext. If having v4 and v5 better have 2 ini files.
  [java]
  extension=java.so

Installing PHP-Java bridge (2)

- Loading with dl() is possible but kills performance.
- The best practice is to start JavaBridge separately and putting this in you php.ini :

```
java.socketname=/var/run/.php-java-bridge_socket
```

  In this mode you can restart the JVM without restarting the PHP client. Nice for configurations with web server – no need to restart the server.
- Start your java binary with something like this :

```
/usr/lib/SunJava2-1.4.2/bin/java
-Djava.library.path=/usr/local/lib/php/extensions/no-debug-non-zts-20020429
-Djava.class.path=/usr/local/lib/php/extensions/no-debug-non-zts-20020429
-Djava.awt.headless=true
JavaBridge
/var/run/.php-java-bridge_socket
```

- -Djava.class.path needs to point to the directory where JavaBridge.class is installed. java.library.path points to where java.so is installed.

Things to know when working with Java

- Classes are instantiated with Java class:
  `$string = new Java('java.lang.String');`
- When the class has no public constructor the PHP variable is an object of class `java.lang.Class`.
- Calling static method is no different than calling normal methods.
- Java exceptions are PHP errors. A E_WARNING will be generated on exception.
- `java_last_exception_get()` tell you if there was an exception. Use `java_last_exception_clear()` to clear it.
- One can define his own error handler.

Accessing remote service over RMI

```
<?php
function gettime(){return array_sum(explode(' ', microtime()));}
function err_handler($errno, $errstr, $errfile, $errline) {
        if ($ex = java_last_exception_get())  echo $ex->toString()."\n";
        java_last_exception_clear();
}
set_error_handler("err_handler");
$class = new Java("java.rmi.Naming");
$calculator = $class->lookup("rmi://localhost/CalculatorService");
var_dump($calculator->sub(14,3));
$i = 0;
$start = gettime();
while ($i++ < 500) $calculator->sub($i, 20);
$end = gettime();
printf("Finished %d calls in %2.5f\n",
        $i-1, $end-$start);

//now let's generate an exception
var_dump($calculator->fakeMethod());
echo "\nOK\n";
?>
```

```
andrey@vivaldi:~/RMI> rmiregistry &
[1] 12079
andrey@vivaldi:~/RMI> java CalculatorServer &
[1] 12199
```

```
vivaldi:/src/php-java-bridge-1.0.6a # php test_new.php
float(11)
Finished 500 calls in 0.99310
bool(false)
string(41) "java.lang.NoSuchMethodException: nomethod"

OK
```

(D)COM

- (D)COM is Windows (tm) only. Proprietary model of Microsoft.
- PHP has an extension to access COM components.
- Using COM objects is like using Java objects from PHP.
- The extension exposes COM and VARIANT classes. VARIANT can be used to escape PHP's internal autoconversions. Unlike Java parameters are passed by reference.
- .NET extension is a wrapper around COM and can load assemblies.

(D)COM examples

```
<?php
  $obj = new COM("ProgId");
  $obj->method();
  $obj2 = new COM("ProgId2", "server.test.org");
  $obj2->method2();
?>

<?php
  $obj = new COM("ProgId", array(
          "Server" => "server.test.org",
          "Username" => "user",
          "Password" => "foo42",
          "Flags" => CTX_REMOTE_SERVER)
      );
  $obj->method();
?>

<?php
  $stack = new DOTNET("mscorlib", "System.Collections.Stack");
  $stack->Push(".Net");
  $stack->Push("Hello ");
  echo $stack->Pop() . $stack->Pop();
?>
```

SOAP

- Stands for Simple Object Access Protocol.
- One of the technologies behing the so popular web services.
- Uses XML as to transport data.
- The transport is not fixed to be HTTP but HTTP is commonly used. One-way messages are possible with SMTP underlying protocol.
- PHP 4 has an extension for SOAP but it is experimental and sometimes crashes. PEAR::SOAP is pure PHP implementation. PHP 5 SOAP module was built on the ground of the v4 one.
- Interoperability is very high, but performance is worse than other technology that uses binary protocol.
- WSDL is for SOAP what IDL is for CORBA.

VL-SRM

- SRM stands for Script Running Machine.
- Current version 0.7.0 (PHP 4 only)
- Developed by Derick Rethans.
- By using SRM daemon your objects live persistently in the memory. For the user they just like normal objects. Calls are transparent. Connection over UNIX or TCP/IP sockets. Only the instantiation differs (try Factory Pattern).
- For throughout description of SRM visit Derick's talk "Enterprise PHP Bananas" just after this one.

SRM example

```
<?php
  class foo {
    var $bar = '';

     function fubar() {
       $this->bar = 'foo';
       return func_get_args();
     }

     function echo_bar() {
       echo $this->bar;
     }
}
?>
```

```
<?php
    include '/usr/local/srm/banana/lib/foo.class.php';

    $f = new foo();
    $s = new SRM('127.0.0.1', 7777);
    //create the banana
    $f = new SRMApp($s, $f, 'key2');

    var_dump($f->fubar(1, 2, 'bar', 3, 4));

    /* Properties don't work yet */
    echo $f->bar;
?>
```

Create your own solution

- PHP 5 makes it easy to build stubs (proxies) in PHP. Just provide __get()/__set()/__call(). Probably PHP 4 can do the job but you need to overload() your classes.
- One needs a way to transport data over the network:
  - A messaging service like JMS built on top of a RDBMS or a product like memcached.
  - TCP/IP and/or UNIX sockets.
- By using scripts everything is under your control and less likely to have crashes. However the speed can be an issue.

pecl/memcache

- “memcached is a high-performance, distributed memory object caching system, generic in nature, but intended for use in speeding up dynamic web applications by alleviating database load”.
- memcached runs on every server in the web server farm. It is CPU lightweight and memory “hungry”. Good symbiosis with httpd.
- Limitation of pecl/memcache is that it does not support multiple servers but is in C. There are 2 other APIs with support.
- Extremely fast. >1000 reads/s (on 2200+ Athlon) of a simple variable.
- Works on *nix and Win32 (PHP4 and PHP5).
- Both procedural (PHP4) and OO (PHP5) API .
- Easy to install : “pear install memcache”

pecl/memcache example

```php
<?php
$mc = memcache_connect('localhost', 11211);
if ($mc) {
     $mc->set("num_key", 123);
     $mc->set("str_key", "String to store in memcached");
     $mc->set("arr_key", array('assoc'=>123, 345, 567));

     $object = new stdClass;
     $object->attribute = 'test';
     $mc->set("obj_key", $object);

     var_dump( $mc->get('str_key'),
               $mc->get('num_key'),
               $mc->get('obj_key'),
               $mc->get('arr_key')
          );
} else {
     echo "Connection to memcached failed";
}
?>
```

```
andrey@poohie:~> php example.php
string(28) "String to store in memcached"
string(3) "123"
object(stdClass)#3 (1) {
  ["attribute"]=>
  string(4) "test"
}
array(3) {
  ["assoc"]=>
  int(123)
  [0]=>
  int(345)
  [1]=>
  int(567)
}
```

pecl/memcache client-server

```
<?php
// client.php
function sum_matrix_by_rows($ar) {
  $dim = count($ar);
  $j=0;
  for ($i = $dim - 1; $i >= 0; --$i)
    $sum_matrix[$i] = array_sum($ar[$i]);
  return array_sum($sum_matrix);
}

if(!($mc=memcache_connect($argv[1],11211))){
  die("Connection failed");
}
while (1) {
  while (1) {
    if ($matrix = $mc->get("matrix")) break;
  }
  if (!is_array($matrix)) {
    $mc->set('matrix_sum', NULL);
  } else {
    echo microtime(1)." Served\n";
    $mc->set('matrix_sum',
           sum_matrix_by_rows($matrix));
    $mc->set('matrix', NULL);
  }
}
?>
```

```
<?php
// server.php
define('DIMENSION', $argv[1]);
function &generate_matrix() {
  static $ar = NULL;
  for ($i = DIMENSION-1; $i >= 0; --$i)
    for ($j = DIMENSION-1; $j >= 0; --$j)
      $ar[$i][$j] = 1;//rand(0, 9);
  return $ar;
}// generate matrix
if(!($mc = memcache_connect($argv[2],11211))){
  die("Connection to failed");
}
while (1) {
  $start = microtime(1);
  $mc->set("matrix", generate_matrix());
  while (1)
    if (($result = $mc->get("matrix_sum"))) {
      $mc->set('matrix_sum', NULL);
      break;
    }
  $end=microtime(1);
  printf("%1.6f\n", $end - $start);
  if ($result !== FALSE) {
    echo "Result is :";
    var_dump($result);
  }
}
?>
```

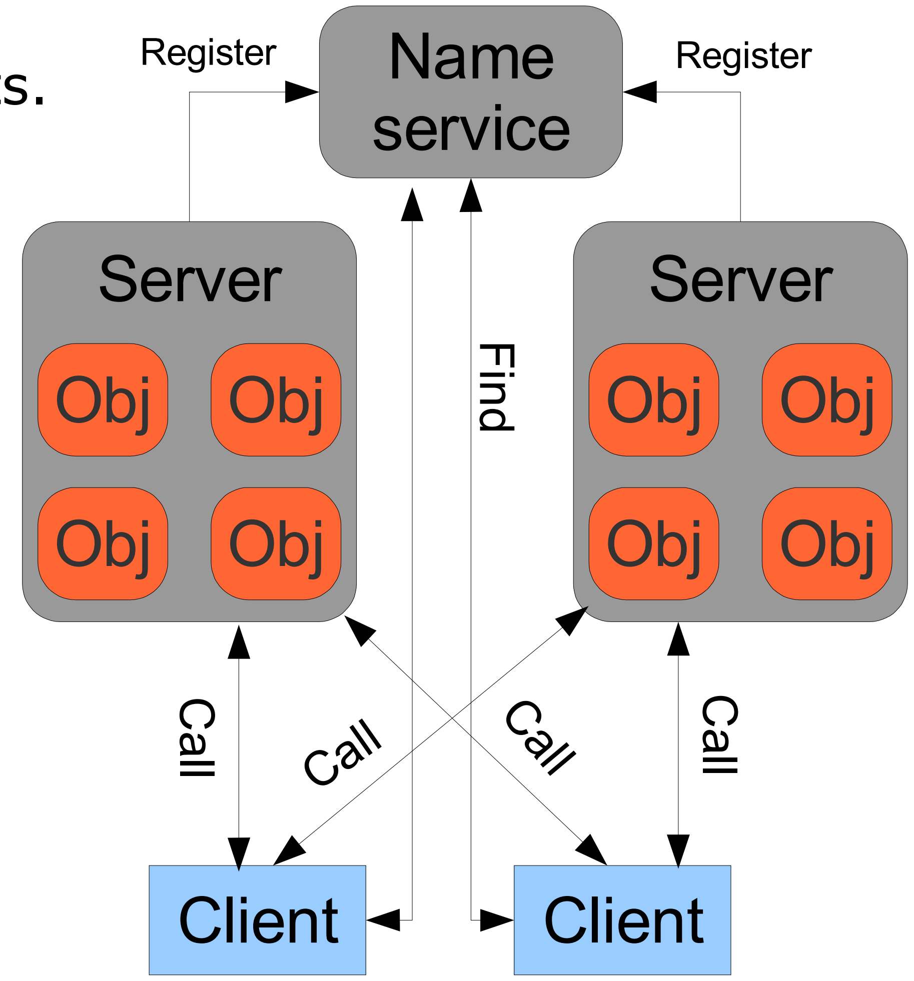
memcache c/s - output

```
andrey@vivaldi:~/test/phpconf> php memc_server.php 192.168.0.100
1099509195.2326 Served
1099509201.1265 Served
1099509201.1582 Served
1099509201.2026 Served
1099509201.2443 Served
1099509201.2839 Served
1099509201.3202 Served
```

```
andrey@vivaldi:~/test/phpconf> php memc_client.php 20 192.168.0.100
0.035353
Result is :string(1) "4"
0.034760
Result is :string(3) "400"
0.051271
Result is :string(3) "400"
0.040590
Result is :string(3) "400"
0.036784
Result is :string(3) "400"
0.034472
Result is :string(3) "400"
```

Sockets based RPC

- SRPC is an example of using sockets.
- PHP 5 based – relies on exceptions and stream sockets.
- There are 3 parts in the system :
  - Client
  - Server
  - Name Service (also a server)
- Steps (client):
  - Create an object – factory
  - Implicit lookup is made
  - Use the object
- Steps (server):
  - Create the server
  - Add servants
  - Register them



The heart of SRPC (1)

```
public /* protected */ function doGetRequest() {
  do {
    $read = array();
    foreach ($this->masterSocketPool as &$v) {
      $read[] = $v->getSocket();
    }
    $read[] = $this->serverSocket->getSocket();  // add the listener
    $mod_fd = stream_select($read, $_w = NULL, $_e = NULL, 15);
  } while ($mod_fd === 0);
  if ($mod_fd === FALSE) throw new Exception("Error occured while doing stream_select() ");
  foreach ($read as &$readableSocket) {
    if ($readableSocket === $this->serverSocket->getSocket()) {
      $conn = stream_socket_accept($this->serverSocket->getSocket());
      $this->masterSocketPool[(int) $conn] = new SocketModem($conn);
    } else {
      $this->socketData[(int) $readableSocket] =  $this->masterSocketPool[(int) $readableSocket]->read();
      $totalDataRead =   $this->masterSocketPool[(int) $readableSocket]->getTransactionReadLen();
      if ($totalDataRead === 0 || $totalDataRead === FALSE) {
        $this->unsetData($readableSocket);
      } else if ($totalDataRead === SocketModem::ST_SUCCESS) {
        $requestObj = unserialize($this->socketData[(int) $readableSocket]);
        if ($this->isDataCorrect($requestObj) === FALSE) {
          //corrupted data
          $this->unsetData($readableSocket);
          continue;
        }
        $requestObj->setCallerID((int) $readableSocket);
        unset($this->socketData[(int) $readableSocket]);
        return $requestObj;
      } else if ($totalDataRead === SocketModem::ST_PROTMSG) {
        $protMsg = $this->masterSocketPool[(int) $readableSocket]->getProtocolMessageData();
        if ($protMsg === 1024) // closing the socket with message
          $this->unsetData($readableSocket);
      }
    }// if
  }// for
}// doGetRequest
```

Questions?

I am reachable at
andrey.hristov_100days_de or **andrey_php_net**

Resources :
This presentation
http://andrey.hristov.com/projects/php_stuff/pres/
PHP-Orbit
http://www.saout.de/misc/
Universe
http://universe-phpext.sourceforge.net/
CORBA::ORBit
http://people.redhat.com/otaylor/corba/orbit.html
ORBit info
http://orbit-resource.sourceforge.net/
RMI over IIOP
http://java.sun.com/products/rmi-iiop/
PHP-Java bridge
http://php-java-bridge.sourceforge.net/