Paul Biggar, Edsko de Vries and David Gregg
Department of Computer Science and Statistics
Trinly College Dublin
SAC '09: 11th March. 2009

Introduction to phc Challenges to compilation phc solution: use the C API Speedup

## A Practical Solution for Scripting Language Compilers

Paul Biggar, Edsko de Vries and David Gregg

Department of Computer Science and Statistics Trinity College Dublin

SAC '09: 11th March, 2009

2009-03-24

A Practical Solution for Scripting Language Compilers └─Outline



Introduction to phc Challenges to compilation phc solution: use the C API Speedup

#### Outline

- Introduction to phc
- Challenges to compilation
- phc solution: use the C API
- 4 Speedup

2009-03-24

	A Practical Solution for Scripting Language Compilers
ı	
	1
)	Sneak peak
•	

Sneak peak

J Problem: Scripting larguages present funique" problems
(in pactice)

J Soldon: Re-leue as much of the Canonical Implementation
as possible.

Introduction to phc Challenges to compilation phc solution: use the C API Speedup

### Sneak peak

- Problem: Scripting languages present "unique" problems (in practice)
- Solution: Re-use as much of the *Canonical Implementation* as possible.

A Practical Solution for Scripting Language Compilers
Loutline

Outline

Outline to part

Outline to part

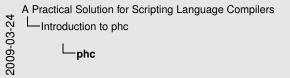
Outline to the comparison

Outli

Introduction to phe Challenges to compilation phc solution: use the C API Speedup

#### Outline

- 1 Introduction to phc
- Challenges to compilation
- 3 phc solution: use the C AF
- 4 Speeduj



phc

Ahaad of time compiler for PHP

Ahayad of time compiler for PHP

Ahaya / http://phpcompiler.org

BSD komse

phc

1. BSD licence useful since its easy to extend

Ahead-of-time compiler for PHP

Introduction to phc

Speedup

Challenges to compilation

phc solution: use the C API

- http://phpcompiler.org
- BSD license

A Practical Solution for Scripting Language Compilers

Challenges to compilation

Outline

Outline

Outline

Introduction to phe Challenges to compilation phe solution: use the C API Speedup

#### Outline

- 1 Introduction to pho
- 2 Challenges to compilation
- 3 phc solution: use the C AF
- 4 Speeduj

A Practical Solution for Scripting Language Compilers

Challenges to compilation

Undefined Language Semantics

Undefined Language Semantics

The PPP group claim that they have the final asy in the approximation of PPP. The groups is approximation in a proper validation rate. There are all thirties are proper validation rate. There are all thirties implementations by a final dumin to secondarily displayed they do not be comparable physical displayed to the control of PPP and the secondarily all the secondarily displayed they do not be controlled to the controlled to t

Introduction to phc
Challenges to compilation
phc solution: use the C API
Speedup

#### **Undefined Language Semantics**

The PHP group claim that they have the final say in the specification of PHP. This group's specification is an implementation, and there is no prose specification or agreed validation suite. There are alternate implementations [...] that claim to be compatible (they don't say what this means) with some version of PHP.

D. M. Jones. Forms of language specification: Examples from commonly used computer languages. ISO/IEC JTC1/SC22/OWG/N0121, February 2008.

A Practical Solution for Scripting Language Compilers

Challenges to compilation

Batteries included

1. all written in C. not PHP

2009-03-2

- 2. Mike Furr earlier: 1000 methods/classes in C
- 3. 4870 functions. 1000 methods



Introduction to phc
Challenges to compilation
phc solution: use the C API
Speedup

#### Batteries included

```
abs()
                                 apc load constants()
                                                                                            array values()
                                                                 array intersect()
acos (
                                 apc sma info()
                                                                 array intersect assoc()
                                                                                           array walk()
acosh(
                                 apc store()
                                                                 array intersect key()
                                                                                            array walk recursive()
                                                                 array intersect uassoc() ArrayIterator::current()
addcslashes()
                                 and breakpoint()
                                                                 array intersect ukey()
                                                                                            ArrayIterator::kev()
addslashes()
                                 apd callstack()
                                 apd clunk()
                                                                 array key exists()
                                                                                            ArrayIterator::next()
aggregate(
aggregate info()
                                 and continue()
                                                                 array keys()
                                                                                            ArrayIterator::rewind()
aggregate methods()
                                 apd croak()
                                                                 array map()
                                                                                            ArravIterator::seek()
                                                                 array_merge()
                                                                                            ArrayIterator::valid()
aggregate methods by list()
                                 apd dump function table()
aggregate methods by regexp()
                                 apd dump persistent resources() array merge recursive(
                                                                                            ArrayObject:: construct()
aggregate properties()
                                 apd dump regular resources()
                                                                 array multisort()
                                                                                            ArrayObject::append()
                                apd echo()
                                                                 array pad()
                                                                                            ArrayObject::count()
aggregate properties by list()
aggregate properties by regexp() and get active symbols()
                                                                 array pop()
                                                                                            ArrayObject::getIterator()
aggregation info()
                                 apd set pprof trace()
                                                                 array product()
                                                                                            ArrayObject::offsetExists()
apache child terminate()
                                 and set session()
                                                                 array push()
                                                                                            ArrayObject::offsetGet()
apache get modules()
                                 and set session trace()
                                                                 array rand()
                                                                                            ArrayObject::offsetSet()
apache get version()
                                 apd set socket session trace()
                                                                array reduce()
                                                                                            ArrayObject::offsetUnset()
apache getenv(
                                 array()
                                                                 array reverse(
                                                                                            arsort()
apache lookup uri()
                                 array change key case()
                                                                 array search()
                                                                                            ascii2ebcdic()
                                                                                            asin()
apache note()
                                 array chunk()
                                                                 array shift()
                                 array combine()
                                                                 array slice()
                                                                                            asinh()
apache request headers()
apache reset timeout()
                                 array count values()
                                                                 array splice()
                                                                                            asort()
apache response headers()
                                 array diff()
                                                                 array sum()
                                                                                            aspell check()
apache setenv()
                                 array diff assoc()
                                                                 array udiff()
                                                                                            aspell check raw()
apc add()
                                 array diff key()
                                                                 array udiff assoc()
                                                                                            aspell new()
apc cache info()
                                 array diff uassoc()
                                                                 array udiff uassoc()
                                                                                            aspell_suggest()
                                 array diff ukey()
apc clear cache()
                                                                 array uintersect()
                                                                                            assert()
apc compile file()
                                 array fill()
                                                                 array uintersect assoc() assert options()
apc define constants()
                                 array fill keys()
                                                                 array uintersect uassoc() atan()
                                 array filter()
apc delete()
                                                                 array unique()
                                                                                            atan2()
apc fetch()
                                 array flip()
                                                                 array unshift()
                                                                                            atanh()
```

Jeff Atwood, Coding Horror, May 20th, 2008 http://www.codinghorror.com/blog/archives/001119.html

Trinity College Dublin

A Practical Solution for Scripting Language Compilers

Challenges to compilation

Change between releases



Introduction to phc
Challenges to compilation
phc solution: use the C API
Speedup

#### Change between releases

```
<?php
  var_dump (0x9fa0ff0b);
?>
```

```
PHP 5.2.1 (32-bit)
```

int(2147483647)

#### PHP 5.2.3 (32-bit)

float(2678128395)

A Practical Solution for Scripting Language Compilers

Challenges to compilation

Run-time code generation



- 1. scripting langs are typically made for interpreters
- 2. can do source inclusion at compile time
- 3. same mechanism for plugins

Introduction to phc
Challenges to compilation
phc solution: use the C API
Speedup

#### Run-time code generation

```
<?php
  eval ($arqv[1]);
?>
<?php
  include ("mylib.php");
  . . .
  include ("plugin.php");
  . . .
?>
```

A Practical Solution for Scripting Language Compilers

—phc solution: use the C API

—Outline

Outline

Introduction to phc Challenges to compilation phc solution: use the C API Speedup

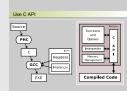
#### Outline

- 1 Introduction to pho
- Challenges to compilation
- 3 phc solution: use the C API
- Speeduj

A Practical Solution for Scripting Language Compilers

—phc solution: use the C API

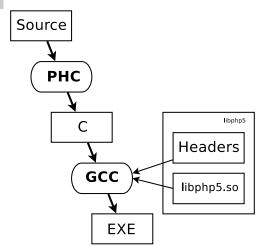
—Use C API

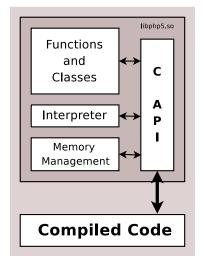


Introduction to phc Challenges to compilation phc solution: use the C API Speedup

#### Use C API

- 1. RTCG
- 2. Functions
- 3. Changes between releases: also use C API at compile-time





A Practical Solution for Scripting Language Compilers

\_\_phc solution: use the C API

—More detail

PHP zval
Python PyObject
Ruby VALUE
Lua TValue

H. Mammad and R. brasilmoty, C AP's in extension and extensible Straighgas, Journal of Universal Computer Science, 13(6):539–653, 2007.

Introduction to phc Challenges to compilation phc solution: use the C API Speedup

#### More detail

PHP	zval
Python	PyObject
Ruby	VALUE
Lua	TValue

H. Muhammad and R. Ierusalimschy. C APIs in extension and extensible languages. Journal of Universal Computer Science, 13(6):839–853, 2007.

- 1. C API is just zval + macros and functions
- 2. Use (target) PHP's C API at run-time



A Practical Solution for Scripting Language Compilers

—phc solution: use the C API

—Applicability

Applicability

- Everything

- Parl

- Parl

- Ruly

- Td - I zook

Introduction to phc Challenges to compilation phc solution: use the C API Speedup

#### Applicability

- Everything
  - Perl
  - PHP
  - Ruby
    - Tcl I think



A Practical Solution for Scripting Language Compilers

—phc solution: use the C API

—Applicability

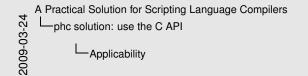
Applicability

Severything
Property
Applicability
Applicab

Introduction to phc
Challenges to compilation
phc solution: use the C API
Speedup

#### **Applicability**

- Everything
  - Perl
  - PHP
  - Ruby
  - Tcl I think
- Except specification
  - Lua
  - Python



Applicability

- Evryting
- Put
- Put
- Put
- Put
- Put
- Put
- Tal-Inse.
- Except operations
- Last
- Fylian
- Name

Introduction to phc Challenges to compilation phc solution: use the C API Speedup

#### **Applicability**

- Everything
  - Perl
  - PHP
  - Ruby
  - Tcl I think
- Except specification
  - Lua
  - Python
- Not at all
  - Javascript

```
2009-03-24
```

A Practical Solution for Scripting Language Compilers

—phc solution: use the C API

—Simple listings: \$i = 0

```
Simple listings: $i = 0

// 51 = 0;

eash Published (scook_st, *1*, 5863374, pub);

physical scook (pub);

physical scook (pub);

physical scook (pub);
```

Introduction to phc Challenges to compilation phc solution: use the C API Speedup

#### Simple listings: \$i = 0

```
// $i = 0;
{
   zval* p_i;
   php_hash_find (LOCAL_ST, "i", 5863374, p_i);
   php_destruct (p_i);
   php_allocate (p_i);
   ZVAL_LONG (*p_i, 0);
}
```

```
2009-03-24
```

A Practical Solution for Scripting Language Compilers

Lphc solution: use the C API

-Example: \$i = 0

#### Example: \$i = 0

(F. 18 - 19)

If (Semi\_s) = 100.5

Impl\_s : 10 [instructions\_prod\_prod\_s

Impl\_s : 10 [instructions\_prod\_prod\_s

Impl\_s : 10 [instructions\_prod\_prod\_s

Impl\_s : 10 [instructions\_prod\_s

Impl\_s : 10 [instruction

Introduction to phc Challenges to compilation phc solution: use the C API Speedup

#### Example: \$i = 0

```
// $i = 0;
  if (local i == NULL)
    local_i = EG (uninitialized_zval_ptr);
    local_i->refcount++;
  zval **p lhs = &local i;
  zval *value;
  if ((*p_lhs)->is_ref)
    // Always overwrite the current value
    value = *p_lhs;
    zval_dtor (value);
  else
    ALLOC_INIT_ZVAL (value);
    zval_ptr_dtor (p_lhs);
    *p_lhs = value;
  ZVAL_LONG (value, 0);
```

```
2009-03-24
```

A Practical Solution for Scripting Language Compilers

—phc solution: use the C API

—Example: \$i = \$j

Example: Si = Sj

The state of the state of

Introduction to phc Challenges to compilation phc solution: use the C API Speedup

#### Example: \$i = \$j

```
if (local_i -- NULL)
 local_i = EG (uninitialized_zval_ptr);
 local_i->refcount++;
zval **p_lhs = &local_i;
zval *rhs;
if (local_j -- NULL)
 rhs - EG (uninitialized_zval_ptr);
else
 rhs = local_i;
if (*p_lhs !- rhs)
 if ((*p_lhs)->is_ref)
   zval_dtor (*p_lhs);
   (*p_lhs)->value = rhs->value;
   (*p_lhs) ->type = rhs->type;
   zval_copy_ctor (*p_lhs);
  else
   zval_ptr_dtor (p_lhs);
   if (rhs->is_ref)
     *p_lhs = zvp_clone_ex (rhs);
   else
     rhs->refcount++;
     *p_lhs = rhs;
```

A Practical Solution for Scripting Language Compilers

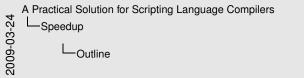
—phc solution: use the C API

—Example: printf (\$f)

Introduction to phc Challenges to compilation phc solution: use the C API Speedup

#### Example: printf (\$f)

Example: printf (\$f)





#### Outline

- Introduction to pho
- Challenges to compilation
- 3 phc solution: use the C AF
- 4 Speedup



#### Original Speed-up

- 1. Why is experiemental evaluation a speedup?
- 2. That's an interesting result. Shouldnt compilers always be faster!!!
- 3. PHP's interpreter isnt slowed by interpreter loop. Rather its the level of dynamicism.

0.1x

(10 times slower than the PHP interpreter)

2009-03-24

1. each statement is pretty high level

```
Introduction to phe
Challenges to compilation
phe solution: use the C API
Speedup
```

#### The problem with copies

```
<?php
 for (\$i = 0; \$i < \$n; \$i++)
    $str = $str . "hello";
?>
<?php
 for ($i = 0; $i < $n; $i++)
    $T = $str . "hello";
    str = sT;
?>
```

# A Practical Solution for Scripting Language Compilers \_\_Speedup \_\_Optimization



- 1. We dont need to know how to fold constants we just pass it off to PHP's eval
- 2. PHP implements this
- 3. function cant change afte first invocation dont need lookup-cache of inline cache or polymorphic inline cache

Introduction to phc Challenges to compilation phc solution: use the C API Speedup

#### Optimization

#### Constant folding

```
<?php
...
$T = "5" + true;
...
?>

<?php
...
$T = 6;
...
?>
```

```
A Practical Solution for Scripting Language Compilers

—Speedup

—Optimization
```



- We dont need to know how to fold constants we just pass it off to PHP's eval
- 2. PHP implements this
- 3. function cant change afte first invocation dont need lookup-cache of inline cache or polymorphic inline cache

- Constant folding
- Constant pooling

```
<?php
  $sum = 0;
  for ($i = 0; $i < 10; $i=$i+1)
  {
    $sum .= "hello";
  }
}
</pre>
```



- We dont need to know how to fold constants we just pass it off to PHP's eval
- 2. PHP implements this
- 3. function cant change afte first invocation dont need lookup-cache of inline cache or polymorphic inline cache

- Constant folding
- Constant pooling
- Function caching

```
// printf ($f);
static php_fcall_info printf_info;
{
   php_fcall_info_init ("printf", &printf_info);
   php_hash_find (
      LOCAL_ST, "f", 5863275, &printf_info.params);
   php_call_function (&printf_info);
}
```

```
A Practical Solution for Scripting Language Compilers

Speedup

Optimization
```

```
Optimization

Sociate tiding
Constant points
C
```

- We dont need to know how to fold constants we just pass it off to PHP's eval
- 2. PHP implements this
- 3. function cant change afte first invocation dont need lookup-cache of inline cache or polymorphic inline cache

- Constant folding
- Constant pooling
- Function caching
- Pre-hashing

```
// $i = 0;
{
  zval* p_i;
  php_hash_find (LOCAL_ST, "i", 5863374, p_i);
  php_destruct (p_i);
  php_allocate (p_i);
  ZVAL_LONG (*p_i, 0);
}
```

```
A Practical Solution for Scripting Language Compilers

—Speedup

—Optimization
```



- We dont need to know how to fold constants we just pass it off to PHP's eval
- 2. PHP implements this
- 3. function cant change afte first invocation dont need lookup-cache of inline cache or polymorphic inline cache

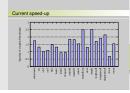
- Constant folding
- Constant pooling
- Function caching
- Pre-hashing
- Symbol-table removal

```
// $i = 0;
{
   php_destruct (local_i);
   php_allocate (local_i);
   ZVAL_LONG (*local_i, 0);
}
```



A Practical Solution for Scripting Language Compilers

\_\_Speedup



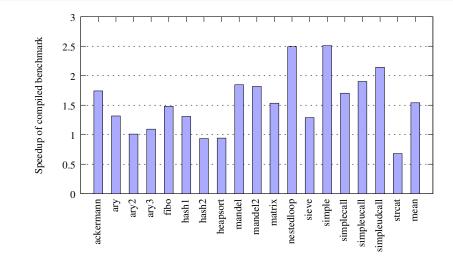
1. Explain how to read graph

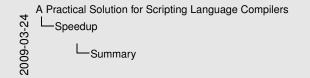
-Current speed-up

- 2. Much better than 0.1x
- 3. C compiler: be 5x faster
- 4. PHP 40x-70x slower



#### Current speed-up





Summary

Sosting language note new problems for complete

Solidation. Revise entiring numbers

Signature and the solidation of the solidat

Introduction to phc
Challenges to compilation
phc solution: use the C API
Speedup

#### Summary

- Scripting languages pose new problems for compilers
- Solution: Re-use existing run-time
  - Speed-ups of 1.5x
  - Future work: Precise optimization required for speed
- http://phpcompiler.org