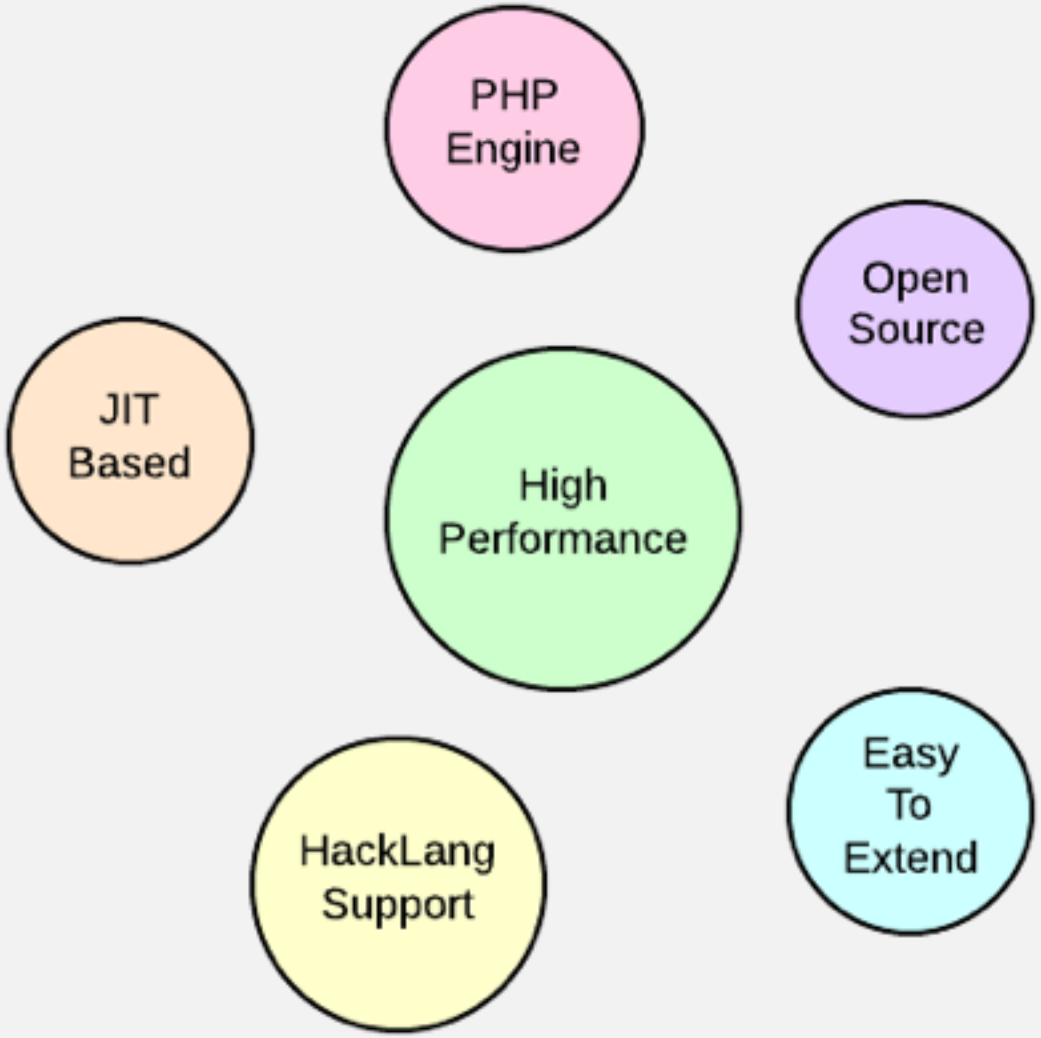
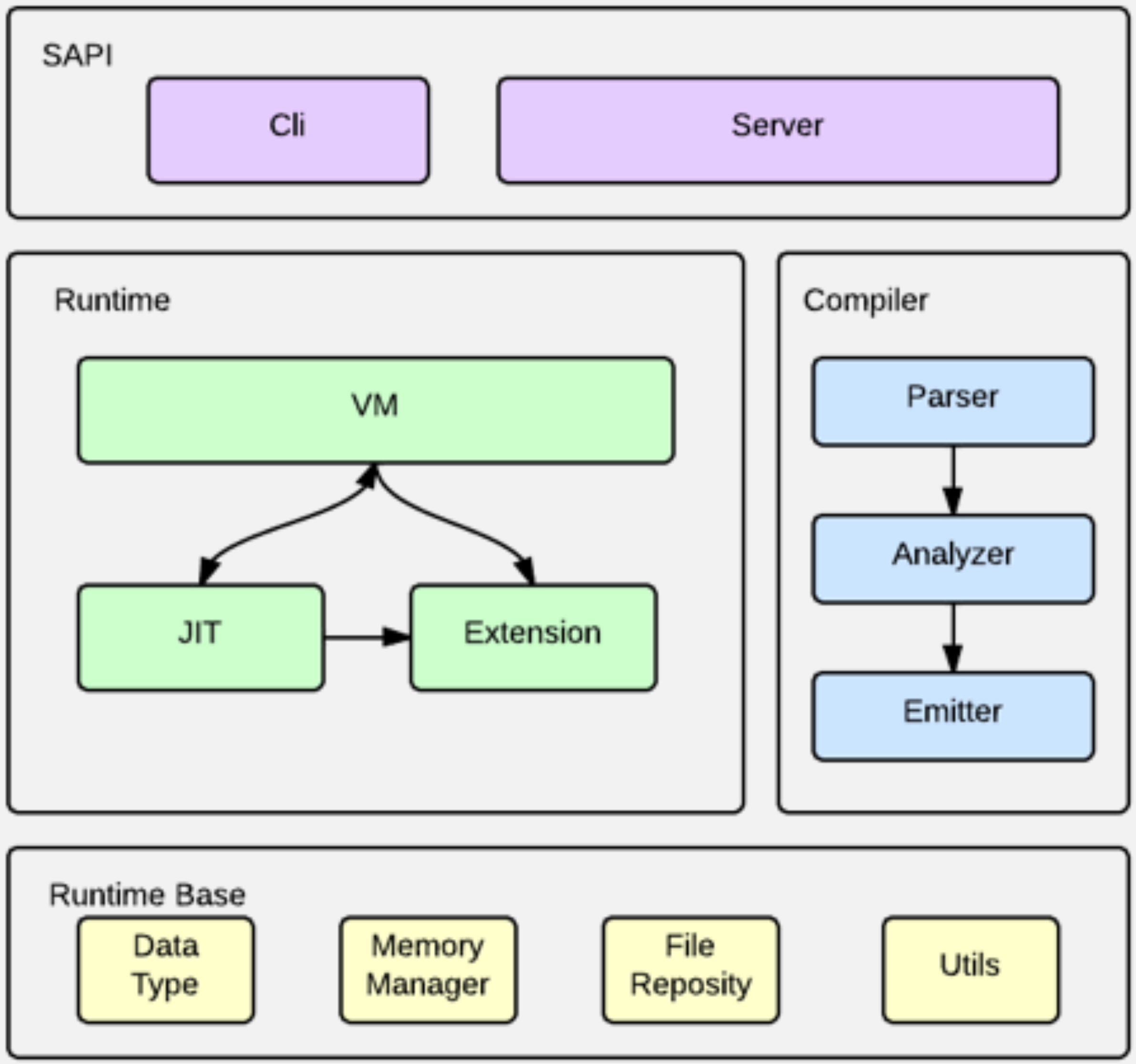
# Introduction to HHVM

Weibing Wang
2014-05

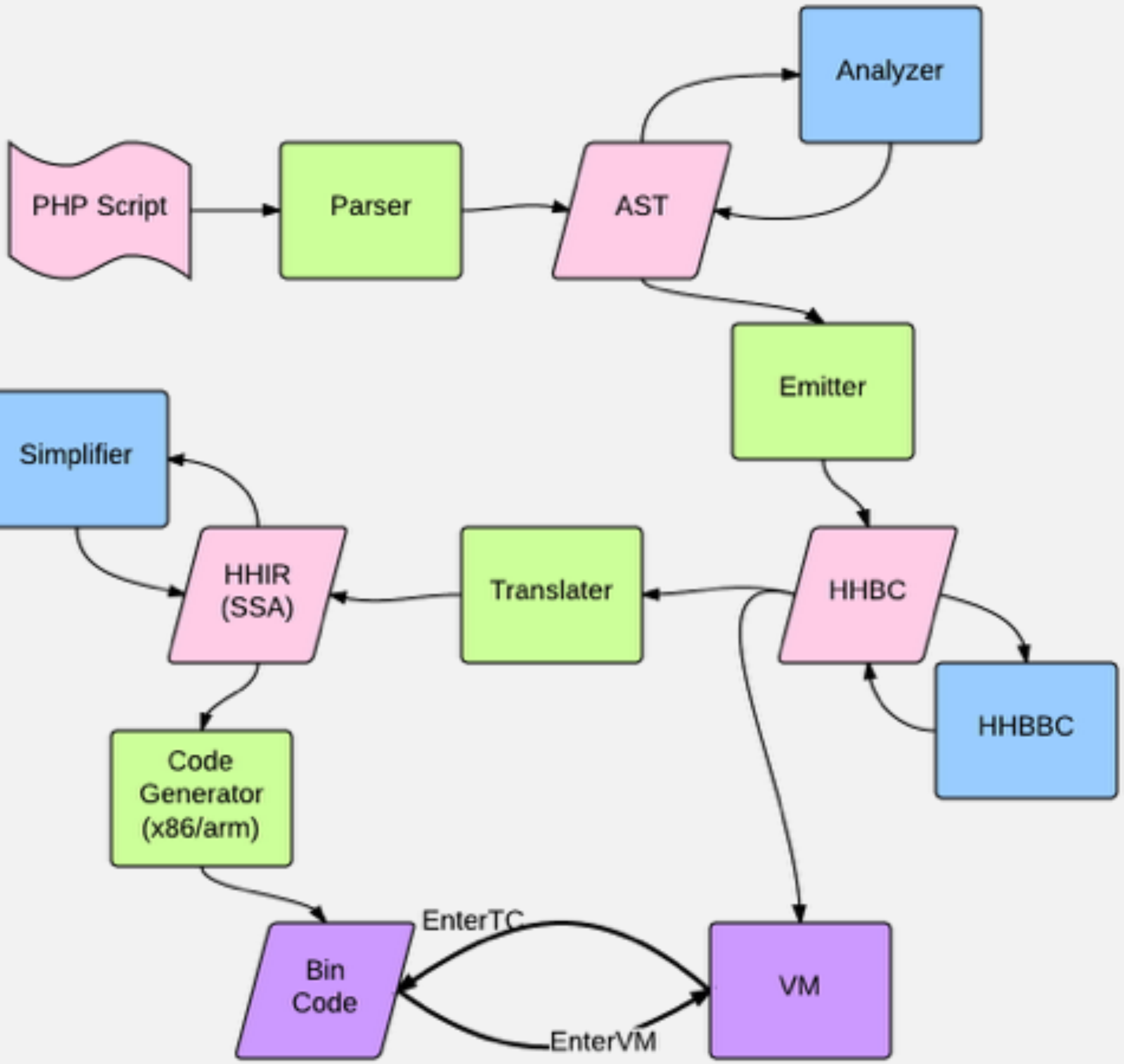
## What is HHVM


PHP Engine
Open Source
JIT Based
High Performance
HackLang Support
Easy To Extend

# HHVM Architecture

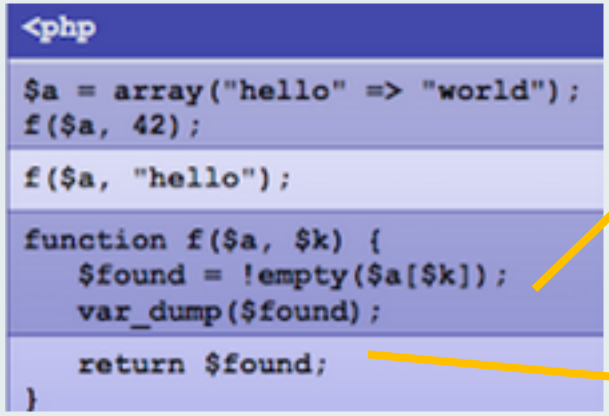


# HHVM Data Flow

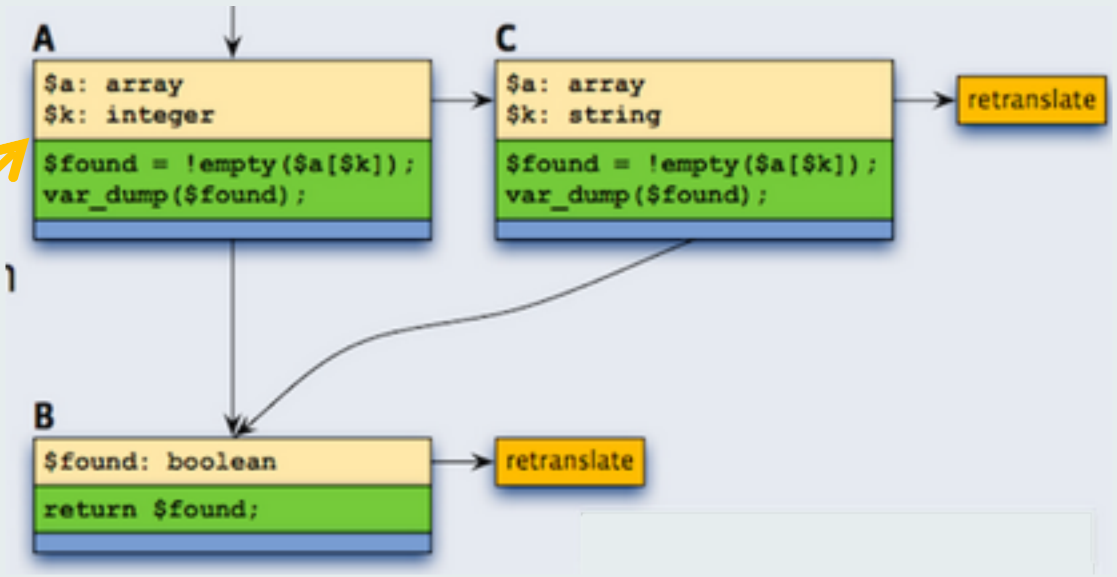


## Why is HHVM fast?

# Type Inference



```
<php
$a = array("hello" => "world");
f($a, 42);
f($a, "hello");
function f($a, $k) {
    $found = !empty($a[$k]);
    var_dump($found);
    return $found;
}
```



```
A
$a: array
$k: integer
$found = !empty($a[$k]);
var_dump($found);

C
$a: array
$k: string
$found = !empty($a[$k]);
var_dump($found);

B
$found: boolean
return $found;
```

# Just In Time Compiler (JIT)

PHP Code

```
<?php
function addPositive($arr) {
    $n = count($arr);
    $sum = 0;
    for ($i = 0; $i < $n; $i++) {
        $elem = $arr[$i];
        if ($elem > 0) {
            $sum = $sum + $elem;
        }
    }
    return $sum;
}
```

HHBC

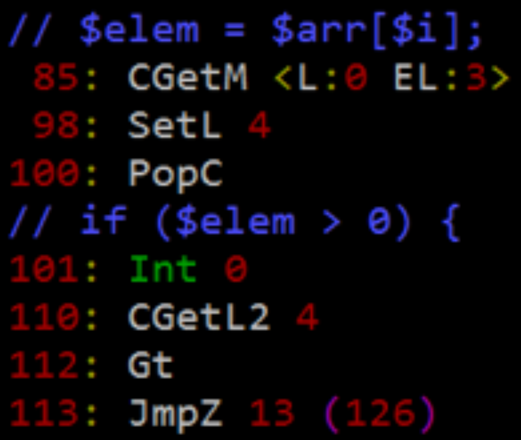
```
// $elem = $arr[$i];
 85: CGetM <L:0 EL:3>
 98: SetL 4
100: PopC
// if ($elem > 0) {
101: Int 0
110: CGetL2 4
112: Gt
113: JmpZ 13 (126)
```

ASM

```
  cmpl  $0xa, 0xc(%rbx)
  jnz 0x276004b2
  cmpl  $0xc, -0x44(%rbp)
  jnle 0x276004b2
101: SetL 4
103: PopC
  movq  (%rbx), %rax
  movq  -0x50(%rbp), %r13
104: Int 0
  xor %ecx, %ecx
113: CGetL2 4
  mov %rax, %rdx
  movl  $0xa, -0x44(%rbp)
  movq  %rax, -0x50(%rbp)
  add $0x10, %rbx
  cmp %rcx, %rdx
115: Gt
116: JmpZ 13 (129)
  jle 0x7608200
```

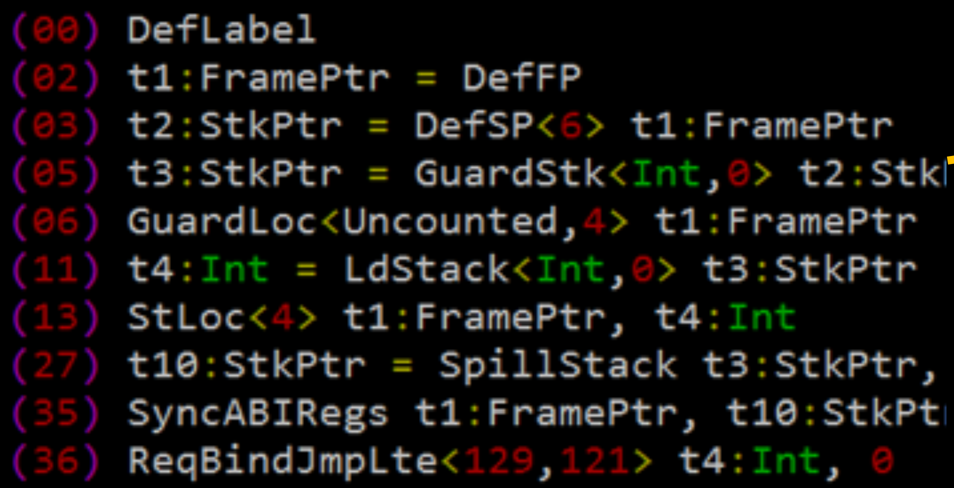
# SSA Based IR Optimization

HHBC



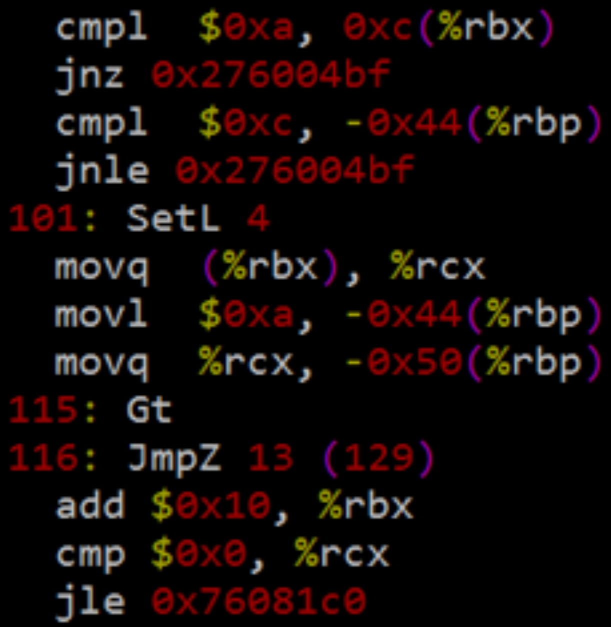
```
// $elem = $arr[$i];
 85: CGetM <L:0 EL:3>
 98: SetL 4
100: PopC
// if ($elem > 0) {
101: Int 0
110: CGetL2 4
112: Gt
113: JmpZ 13 (126)
```

Part of HHIR



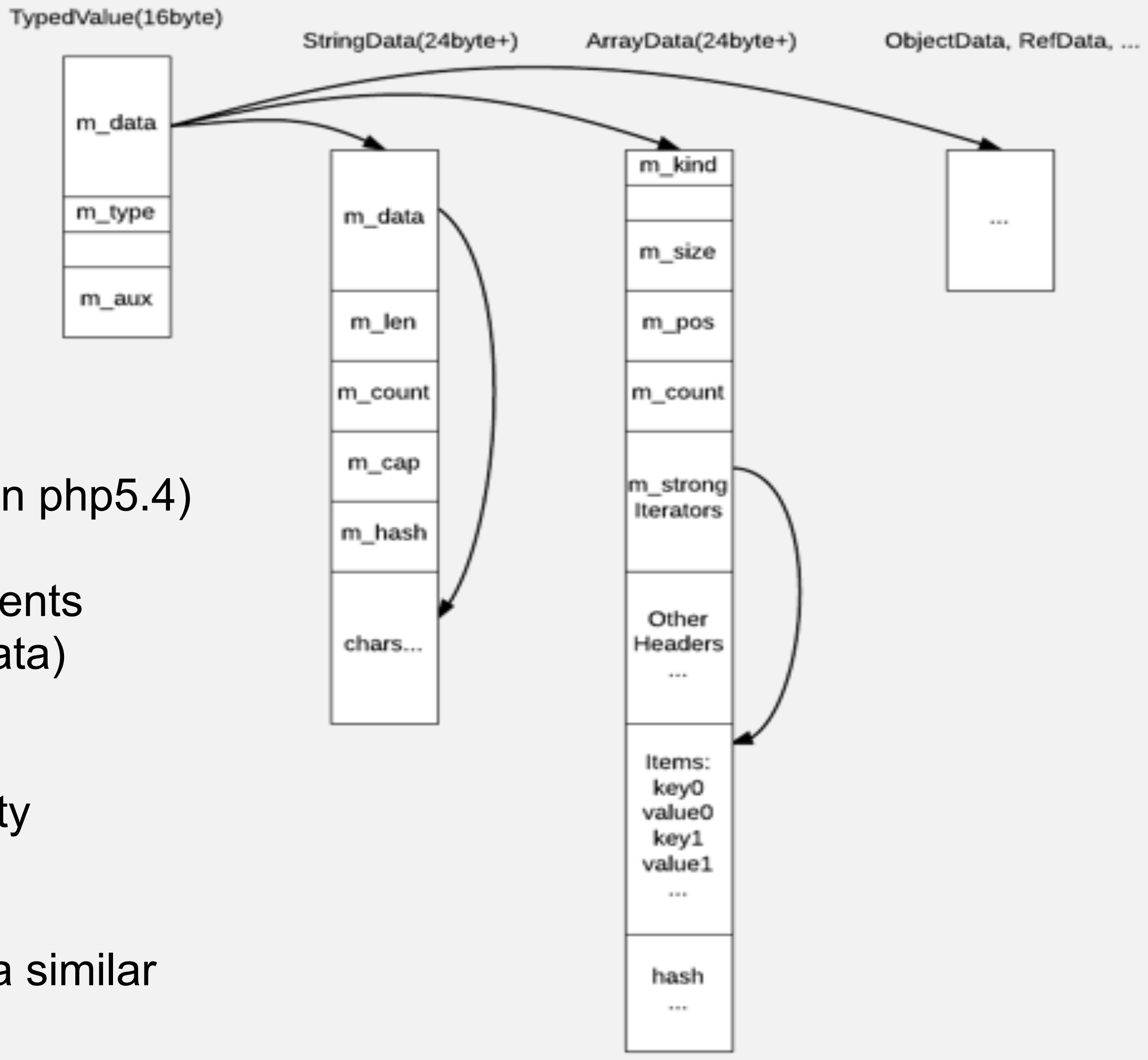
```
(00) DefLabel
(02) t1:FramePtr = DefFP
(03) t2:StkPtr = DefSP<6> t1:FramePtr
(05) t3:StkPtr = GuardStk<Int,0> t2:Stk
(06) GuardLoc<Uncounted,4> t1:FramePtr
(11) t4:Int = LdStack<Int,0> t3:StkPtr
(13) StLoc<4> t1:FramePtr, t4:Int
(27) t10:StkPtr = SpillStack t3:StkPtr,
(35) SyncABIRegs t1:FramePtr, t10:StkPt
(36) ReqBindJmpLte<129,121> t4:Int, 0
```

ASM after HHIR optimization (13 -> 10 instructions)



```
   cmpl  $0xa, 0xc(%rbx)
   jnz 0x276004bf
   cmpl  $0xc, -0x44(%rbp)
   jnle 0x276004bf
101: SetL 4
   movq  (%rbx), %rcx
   movl  $0xa, -0x44(%rbp)
   movq  %rcx, -0x50(%rbp)
115: Gt
116: JmpZ 13 (129)
   add $0x10, %rbx
   cmp $0x0, %rcx
   jle 0x76081c0
```

# Less Memory Usage



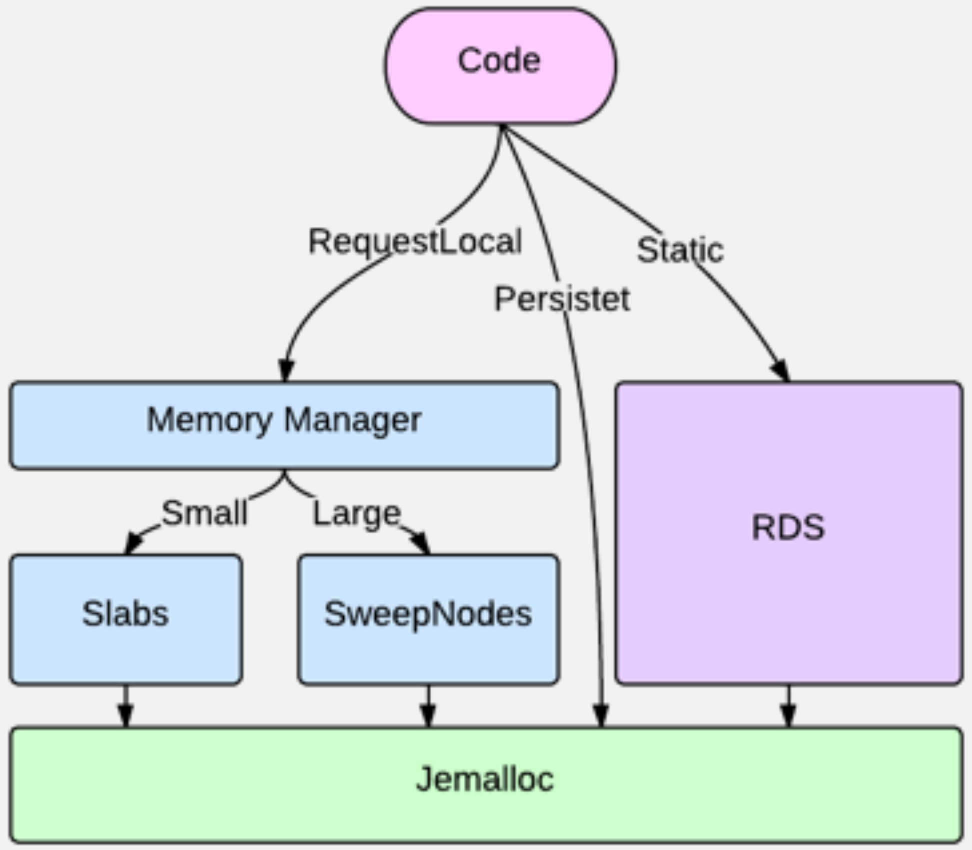
- Smaller Data Size
  (16 byte vs 32 byte in php5.4)

- Less Memory Fragments
  (StringData, ArrayData)

- Less Memory
  => Better code locality
  => Less cpu

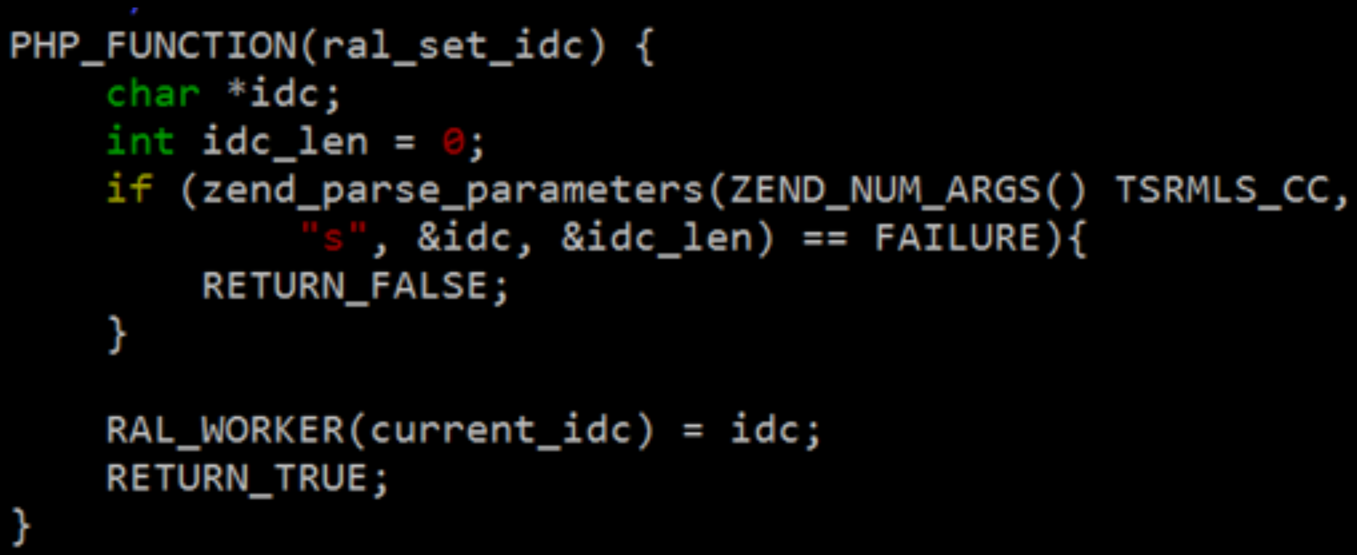
- phpng (php5.7) use a similar layout for zval

# Efficient Memory Manager



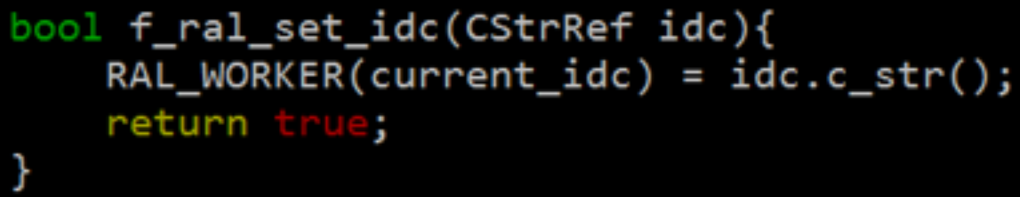
## Simpler Extension API

Zend
Extension



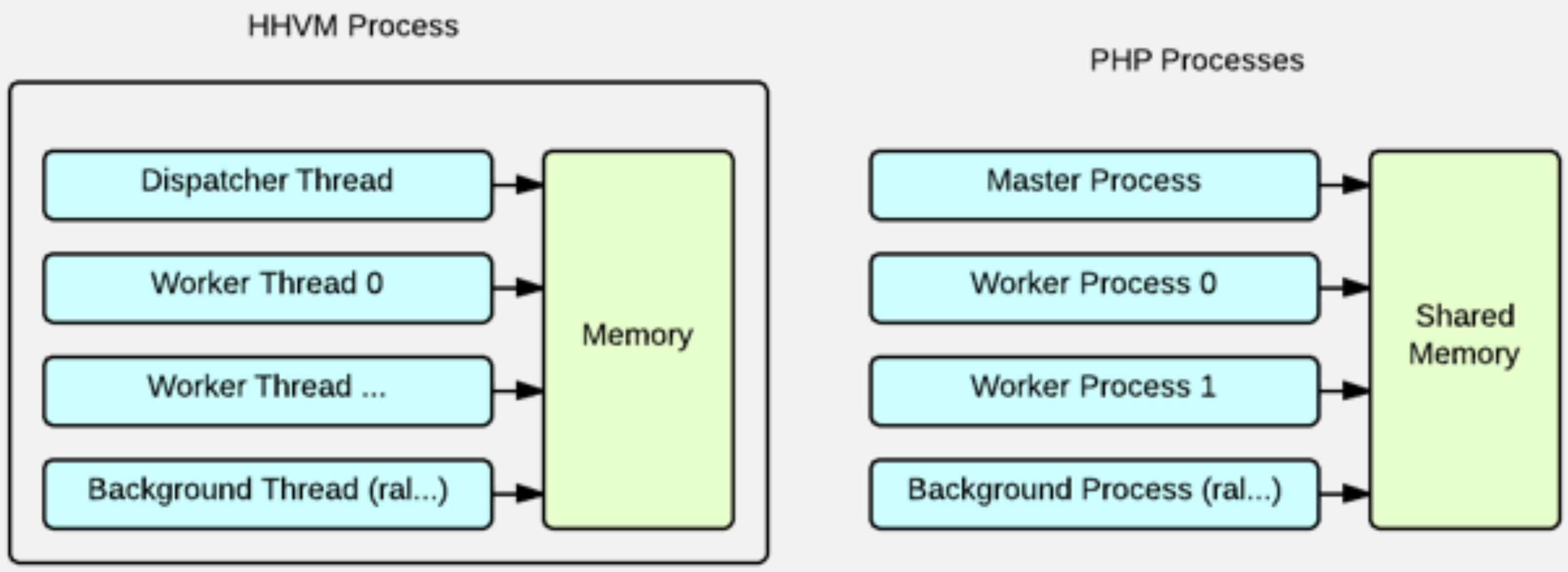
```
PHP_FUNCTION(ral_set_idc) {
    char *idc;
    int idc_len = 0;
    if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC,
            "s", &idc, &idc_len) == FAILURE){
        RETURN_FALSE;
    }

    RAL_WORKER(current_idc) = idc;
    RETURN_TRUE;
}
```

HHVM
Extension



```
bool f_ral_set_idc(CStrRef idc){
    RAL_WORKER(current_idc) = idc.c_str();
    return true;
}
```

# Single Process Architecture



- Advantage
  - Share data structure between different threads (eg. apc)
  - Share file descriptors (eg. connect pool)

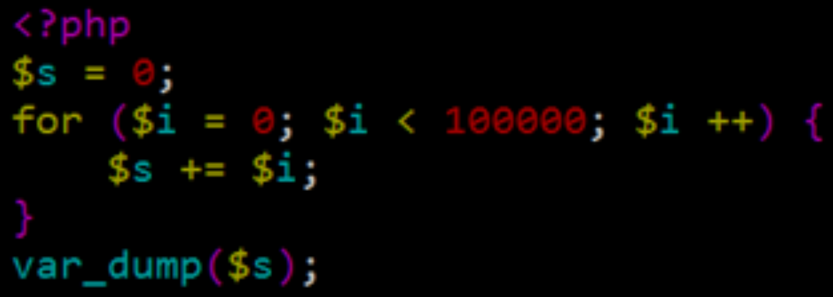
- Disadvantage
  - Thread safety, extra cost for lock/unlock
  - Memory leak

## More...

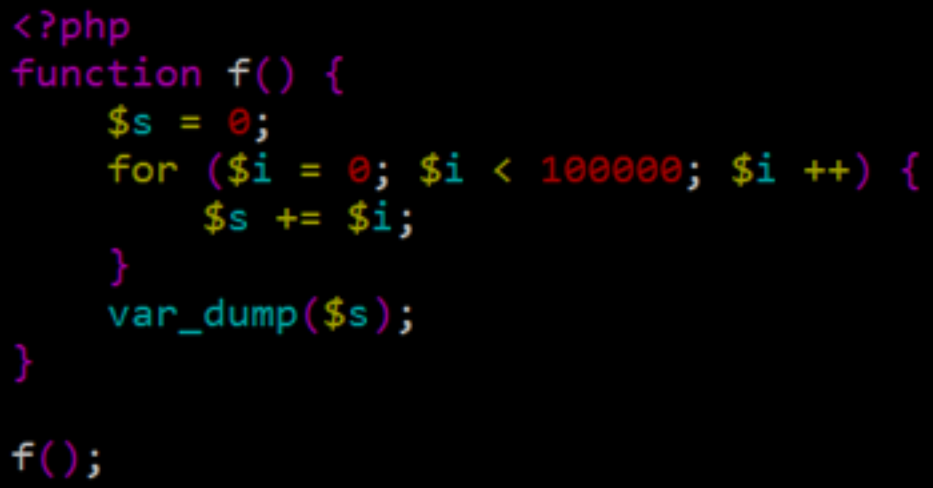
- Other optimizations
  - inline hottest builtin function (eg. count/strlen)
  - use newest pcre (jit for regexp)
  - use gcc 4.8
  - ...
- Ongoing optimizations
  - HHBBC (Bytecode to Bytecode Compiler)
  - Region Compiler
  - ARM64
  - Prototype LLVM integration
  - ...

## HHVM Coding Tips

## Keep Hot Code Out of Global Scope

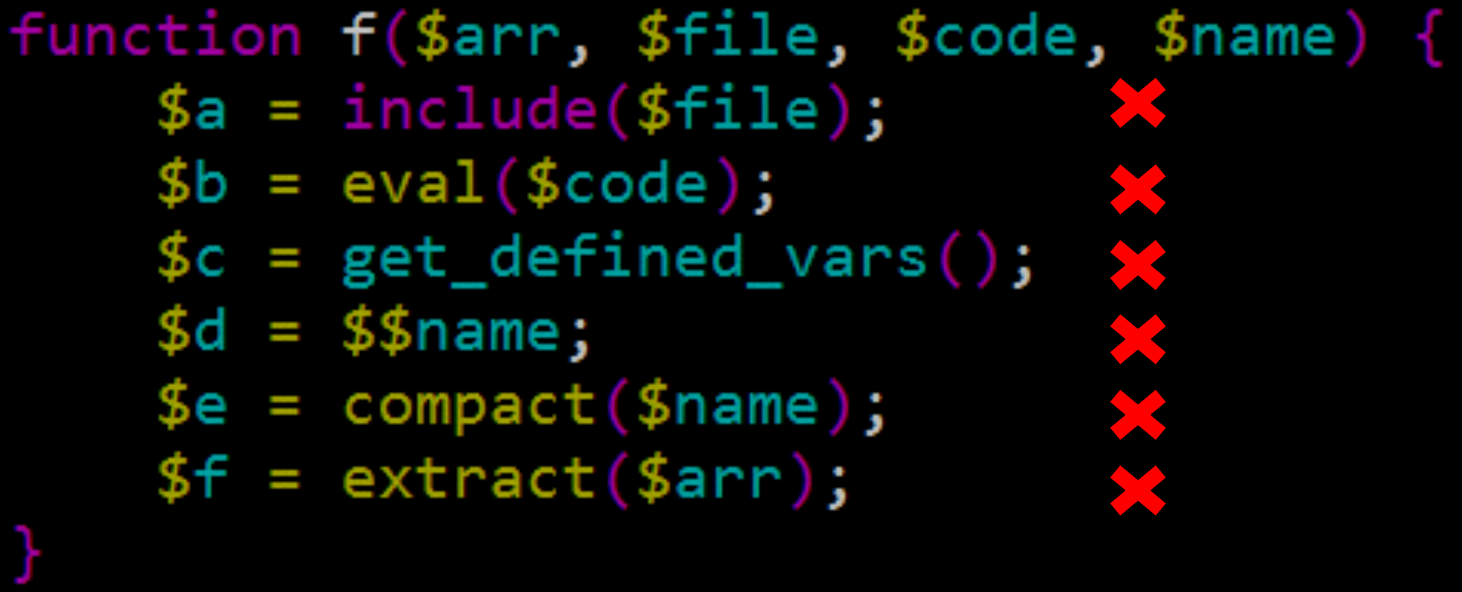


```php
<?php
$s = 0;
for ($i = 0; $i < 100000; $i ++) {
    $s += $i;
}
var_dump($s);
```



```php
<?php
function f() {
    $s = 0;
    for ($i = 0; $i < 100000; $i ++) {
        $s += $i;
    }
    var_dump($s);
}

f();
```

## Avoid Using Dynamic Functionalities



```
function f($arr, $file, $code, $name) {
    $a = include($file);           ✖
    $b = eval($code);              ✖
    $c = get_defined_vars();       ✖
    $d = $$name;                   ✖
    $e = compact($name);           ✖
    $f = extract($arr);            ✖
}
```

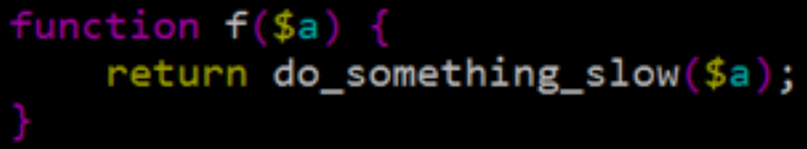
## Declare Properties

```
function f($arr) {
    $arr['key1'] = g();
    h($arr['key2']);
}
```

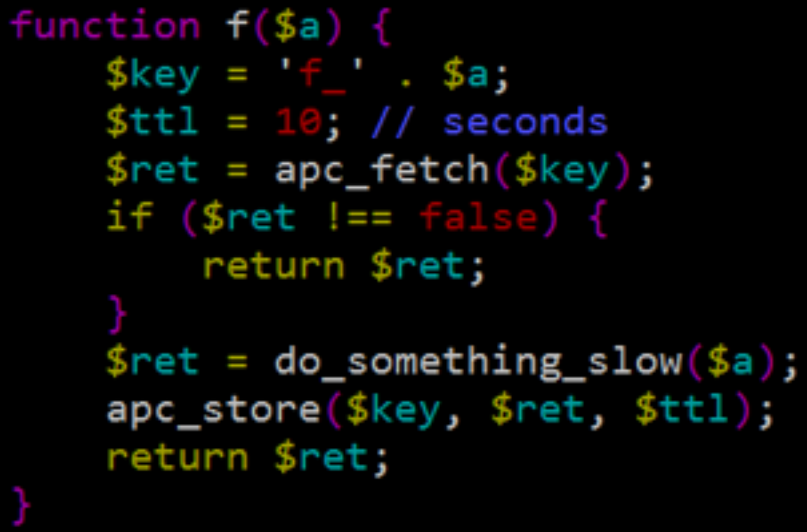
↓

```
class A {
    public $key1;
    public $key2;
}

function f(A $a) {
    $a->key1 = g();
    h($a->key2);
}
```

## Use APC to cache static data



```
function f($a) {
    return do_something_slow($a);
}
```



```
function f($a) {
    $key = 'f_' . $a;
    $ttl = 10; // seconds
    $ret = apc_fetch($key);
    if ($ret !== false) {
        return $ret;
    }
    $ret = do_something_slow($a);
    apc_store($key, $ret, $ttl);
    return $ret;
}
```

## HHVM OP Tips

## Useful Configure Options

- Server.Port
- Server.ThreadCount
- Server.RequestTimeoutSeconds
- Server.RequestMemoryMaxBytes
- AdminServer.Port
- Log.File
- ResourceLimit.MaxRSS
- Debug.CoreDumpReport

## Useful Admin Server Command

- stop
- check-health
- status.html
- vm-tcspace
- jemalloc-stats