

PHP

Vancouver PHP User Group

April 17, 2003. Vancouver

Rasmus Lerdorf <rasmus@php.net>

<http://lerdorf.com/vpug.pdf>

Handling simple data coming from a form took something like this to do in C:

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <string.h>

#define ishex(x) (((x) >= '0' && (x) <= '9') || ((x) >= 'a' &&
                (x) <= 'f') || ((x) >= 'A' && (x) <= 'F'))

int htoi(char *s) {
    int    value;
    char   c;

    c = s[0];
    if(isupper(c)) c = tolower(c);
    value=(c >= '0' && c <= '9' ? c - '0' : c - 'a' + 10) * 16;

    c = s[1];
    if(isupper(c)) c = tolower(c);
    value += c >= '0' && c <= '9' ? c - '0' : c - 'a' + 10;

    return(value);
}

void main(int argc, char *argv[]) {
    char params, data, dest, s, *tmp;
    char name, age;

    puts("Content-type: text/html\r\n");
    puts("<html><head><title>Form Example</title></head>");
    puts("<body><h1>My Example Form</h1>");
    puts("<form action=\"form.cgi\" method=\"GET\">");
    puts("Name: <input type=\"text\" name=\"name\">");
    puts("Age: <input type=\"text\" name=\"age\">");
    puts("<br><input type=\"submit\">");
    puts("</form>");

    data = getenv("QUERY_STRING");
    if(data && *data) {
        params = data; dest = data;
        while(*data) {
            if(data=='+') dest=' ';
            else if(data == '%' && ishex((data+1))&&ishex(*(data+2))) {
                *dest = (char) htoi(data + 1);
                data+=2;
            } else dest = data;
            data++;
            dest++;
        }
        *dest = '\0';
        s = strtok(params, "&");
        do {
            tmp = strchr(s, '=');
            if(tmp) {
                *tmp = '\0';
                if(!strcmp(s, "name")) name = tmp+1;
                else if(!strcmp(s, "age")) age = tmp+1;
            }
        } while(s=strtok(NULL, "&"));

        printf("Hi s, you are s years old\n", name, age);
    }
    puts("</body></html>");
}
```

Perl became an obvious choice because it was made for text processing. The same thing in Perl using CGI.pm:

```
use CGI qw(:standard);
print header;
print start_html('Form Example'),
      hl('My Example Form'),
      start_form,
      "Name: ", textfield('name'),
      p,
      "Age: ", textfield('age'),
      p,
      submit,
      end_form;
if(param()) {
    print "Hi ",em(param('name')),
          "You are ",em(param('age')),
          " years old";
}
print end_html;
```

Much easier both to read and to write, at least to people with a bit of a programming background.

PHP has an HTML-centric approach. The same script in PHP became:

```
<html><head><title>Form Example</title></head>
<body><h1>My Example Form</h1>
<form action="form.phtml" method="POST">
Name: <input type="text" name="name">
Age: <input type="text" name="age">
<br><input type="submit">
</form>
<?if($name):?>
Hi <?echo $name?>, you are <?echo $age?> years old
<?endif?>
</body></html>
```

A block of raw HTML followed by the minimum amount of logic possible.

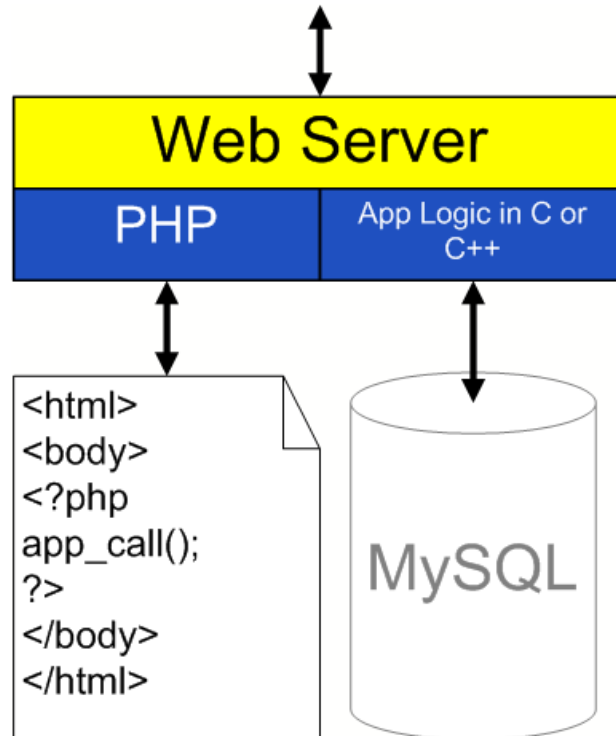
A good solution should

- o Have a shallow learning curve
- o Instant gratification
- o Build on what you know
- o Great documentation
- o Solve the simple problem easily
- o Eliminate tedium
- o Be able to solve even the most complex problem
- o Be secure
- o Use/borrow existing technology
- o Work everywhere

Bonus

- o Be Free
- o Teach the basics by not hiding the problem

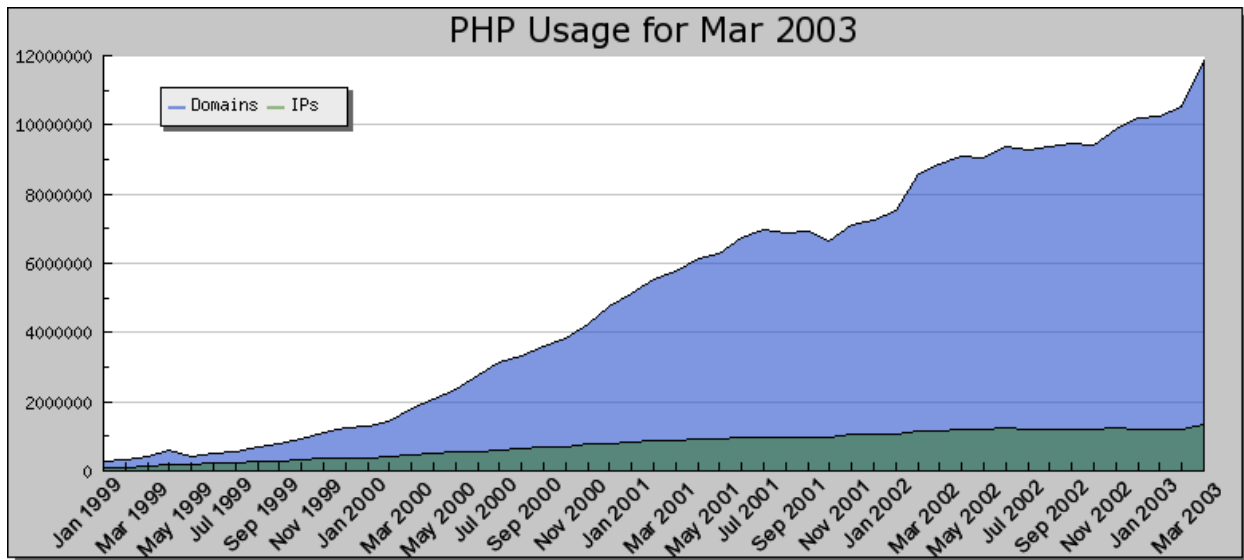
PHP was created as a framework for writing web applications in C or C++ and making it easy to expose the business logic of these applications to a powerful presentation-layer templating language.



Most people don't really use PHP this way. Over the years the templating language improved both in scope and performance to the point where entire web apps could be written in it.

March 2003 Netcraft Report

- o 40,100,739 Domains queried
- o 11,860,645 Domains. 1,316,288 IP addresses
- o PHP installed on 29.58% of all domains



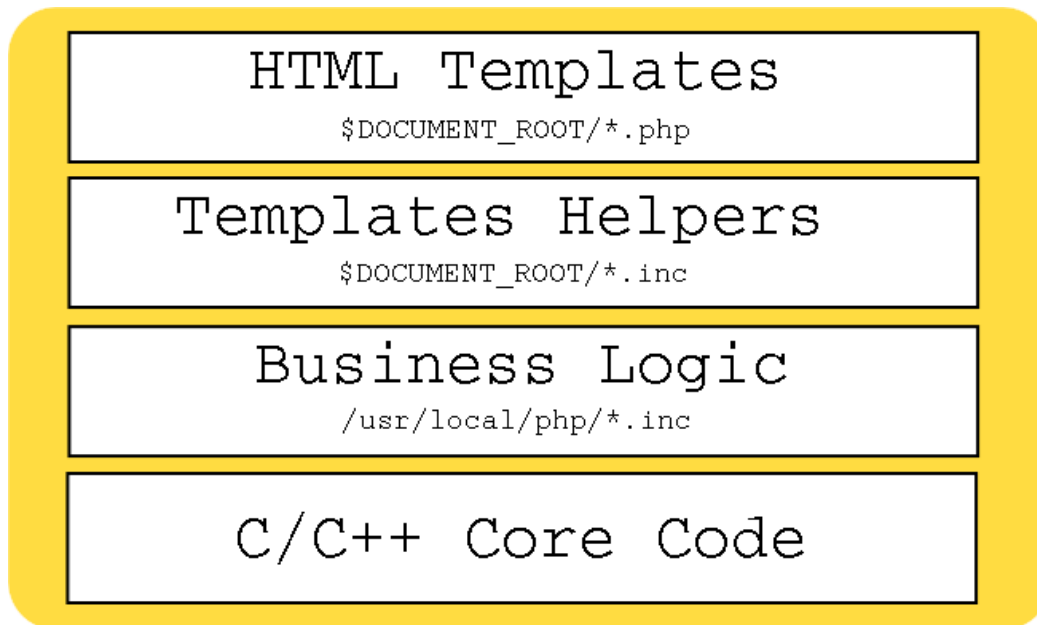
Source: Netcraft

March 2003 Apache Module Report

- o 6,134,492 Apache Servers surveyed
- o 3,107,624 (50.66%) PHP
- o 1,877,707 (30.61%) OpenSSL
- o 1,813,509 (29.56%) mod_ssl
- o 1,345,589 (21.93%) Frontpage
- o 1,297,683 (21.15%) mod_perl
- o 370,550 (6.04%) DAV
- o 321,928 (5.25%) mod_throttle
- o 175,924 (2.87%) AuthMySQL
- o 172,517 (2.81%) mod_jk
- o 170,581 (2.78%) mod_auth_pam

Source: SecuritySpace.com

A typical architecture for a PHP application. The template layer should have as little business logic as possible. As you go down you have less presentation and more business logic.



Guestbook Example

A very simple guestbook example to illustrate basic file handling.

```
<html><head><title>My Guestbook</title></head>
<body>
<h1>Welcome to my Guestbook</h1>
<h2>Please write me a little note below</h2>
<form action="<?="$PHP_SELF#results"?">" method="POST">
<textarea cols=40 rows=5 name=note wrap=virtual></textarea>
<input type=submit value=" Send it ">
</form>
<?if(isset($note)) {
    $fp = fopen("/tmp/notes.txt","a");
    fputs($fp,nl2br($note).'<br>');
    fclose($fp);
}
?><h2>The entries so far:</h2>
<? @ReadFile("/tmp/notes.txt") ?>
</body></html>
```

Output:

My Guestbook

Welcome to my Guestbook
Please write me a little note below

The entries so far:

PHP scripts that talk to databases all look similar to the code below. Connect to the database, select a database, send a query and loop through the results.

```
<?php
mysql_pconnect("db.server.com", "username", "password");
mysql_select_db("products");
$result = mysql_query("SELECT * FROM details");
if ($result) {
    echo "<TABLE>\n";
    echo "<TR><TH>Name</TH><TH>Description</TH></TR>\n";
    while ($a = mysql_fetch_array($result)) {
        echo "<TR><TD>$a[name]</TD>", "<TD>$a[descr]</TD></TR>";
    }
    echo "</TABLE>";
} else {
    echo "<P>Nothing to see here.";
}
?>
```

SQL'izing the Guestbook Example

Recall our file-driven guestbook example from earlier. We are going to convert this into an SQL-driven guestbook by first creating a database, then a schema for the table where we will store the data and then we will modify the code.

```
<h1>Welcome to my Guestbook</h1>
<h2>Please write me a little note below</h2>
<form action="<? echo $PHP_SELF?>" method="POST">
<textarea cols=40 rows=5 name="note"></textarea>
<input type="submit" value=" Send it ">
</form>
<?
if(isset($note)) {
    $fp = fopen("notes.txt","a");
    fputs($fp,nl2br($note)."<br>");
    fclose($fp);
}
?>
<h2>The entries so far:</h2>
<? @ReadFile("notes.txt") ?>
```

Create a database

```
mysqladmin create mydb
```

Create a Schema

```
CREATE TABLE comments (
    id int(8) DEFAULT '0' NOT NULL auto_increment,
    comment text,
    ts datetime,
    PRIMARY KEY (id)
);
```

SQL'izing the Guestbook Example

Here we add the necessary code to store our guestbook comments in an SQL database

```
<html><head><title>My Guestbook</title></head>
<body>
<h1>Welcome to my Guestbook</h1>
<h2>Please write me a little note below</h2>
<form action="<? echo "$PHP_SELF#results"?" method="POST">
<textarea cols=40 rows=5 name="note" wrap=virtual></textarea>
<input type="submit" value=" Send it ">
</form>
<?
    mysql_connect('localhost');
    mysql_select_db('mydb');
    if(isset($note)) {
        $ts = date("Y-m-d H:i:s");
        mysql_query("insert into comments values (0,'$note','$ts')");
    }
?>
<h2>The entries so far:</h2>
<? $result = mysql_query("select * from comments order by ts desc");
    while($row=mysql_fetch_row($result)) {
        echo $row[0] . " " . $row[1] . " " . $row[2] . "<br>\n";
    } ?>
</body></html>
```

Output:

My Guestbook

Welcome to my Guestbook
Please write me a little note below

The entries so far:

Creating a PNG with a TrueType font

```
<?
Header("Content-type: image/png");
$im = ImageCreate(630,80);
$blue = ImageColorAllocate($im,0x5B,0x69,0xA6);
$white = ImageColorAllocate($im,255,255,255);
$black = ImageColorAllocate($im,0,0,0);
ImageTTFText($im, 45, 0, 10, 57, $black, "CANDY", $text);
ImageTTFText($im, 45, 0, 6, 54, $white, "CANDY", $text);
ImagePNG($im);
?>

<IMG src="txt.php?text=<?echo urlencode($text)?>">
```


CreateFrom and Bounding Box Math

```
<?
Header("Content-type: image/png");
$font = 'phpi';
if(!$si) $si = 66;
$im = ImageCreateFromPNG('php-blank.png');
$tsize = imagettfbbox($si,0,$font,$text);
$dx = abs($tsize[2]-$tsize[0]);
$dy = abs($tsize[5]-$tsize[3]);
$x = ( imagesx($im) - $dx ) / 2;
$y = ( imagesy($im) - $dy ) / 2 + 3*$dy/4;
$blue = ImageColorAllocate($im,0x5B,0x69,0xA6);
$white = ImageColorAllocate($im,255,255,255);
$black = ImageColorAllocate($im,0,0,0);
ImageAlphaBlending($im,true);
ImageTTFText($im, $si, 0, $x, $y, $white, $font, $text);
ImageTTFText($im, $si, 0, $x+2, $y, $white, $font, $text);
ImageTTFText($im, $si, 0, $x, $y+2, $white, $font, $text);
ImageTTFText($im, $si, 0, $x+2, $y+2, $white, $font, $text);
ImageTTFText($im, $si, 0, $x+1, $y+1, $black, $font, $text);
ImagePNG($im);
?>

<IMG src="txt2.php?text=<?echo urlencode($text)?>&si=<?echo $si?>">

Text:   Size:
```

Built-in Fonts

GD comes with 5 built-in fonts. They aren't all that useful.

```
<?
$im = ImageCreate(175,125);
$white = ImageColorAllocate($im,255,255,255);
$black = ImageColorAllocate($im,0,0,0);
ImageString($im,1,10,20,"Font 1: ABCdef",$black);
ImageString($im,2,10,35,"Font 2: ABCdef",$black);
ImageString($im,3,10,53,"Font 3: ABCdef",$black);
ImageString($im,4,10,70,"Font 4: ABCdef",$black);
ImageString($im,5,10,90,"Font 5: ABCdef",$black);
ImageStringUp($im,5,150,118,"Vertical Text",$black);
Header('Content-Type: image/png');
ImagePNG($im);
?>
```

Output:

Font 1: ABCdef

Font 2: ABCdef

Font 3: ABCdef

Font 4: ABCdef

Font 5: ABCdef

Vertical Text

TrueType Fonts

You can use any TrueType Font that includes a Unicode mapping table. Fonts such as Wingdings will not work.

```
<?
$im = ImageCreate(600,7150);
$white = ImageColorAllocate($im,255,255,255);
$black = ImageColorAllocate($im,0,0,0);
$dir = opendir('/usr/share/fonts/truetype');
$y=30;
while($file = readdir($dir)) {
    if(substr($file, strrpos($file, '.')) == '.ttf') {
        ImageString($im, 5, 5, $y-20, substr($file, 0, -4), $black);
        ImageTTFText($im, 30, 0, 100, $y, $black, substr($file, 0, -4), "ABCdéf123");
        $y+=40;
    }
}
Header('Content-Type: image/png');
ImagePNG($im);
?>
```

Output:

abalc ABCdéf123
 a_d_mono ABCD □ 123
 Adresack ABCD □ 123
 aggstock ABCd f123
 ariblk **ABCdéf123**
 arnari ABCdéf123
 arnar ABCdéf123
 bkant ABCdéf123
 Bookosbi **ABCdéf123**
 bookosb **ABCdéf123**
 Bookosi ABCdéf123
 Bookos ABCdéf123
 calist ABCdéf123
 comicbd **ABCdéf123**
 comic ABCdéf123
 coprgrtb **ABCDÉF 123**
 coprgrtl ABCDÉF 123
 courbd **ABCdéf123**
 courbi **ABCdéf123**
 couri ABCdéf123
 cour ABCdéf123
 dc_sans **A B C 'D** □ **'F** □ □ □
 dc_serif **A B C 'D** □ **'F** □ □ □
 dilate a b c d e f 1 2 3
 dirtydoz **ABC D □ 123**
 Dotmatrix ABCd é f 1 2 3
 dronecat **QBCD** □ □ □ □
 earth a b c d e f 1 2 3
 Eklekti0 ABCDÉF123
 electroh ΔΒΓϵ □ 123
 Eroded2020 a b c d e f 1 2 3

Reading EXIF Headers from a JPEG

```
<?php
$data = exif_read_data('presentations/slides/intro/img_resize.jpg');
foreach($data as $key=>$val) {
    if(is_array($val)) {
        foreach($val as $k=>$v) {
            echo $key."[$k]: $v<br />\n";
        }
    } else
        echo "$key: " .@substr($val,0,40)."<br />\n";
}
?>
```

Output:

```
FileName: img_resize.jpg
FileDateTime: 1027351588
FileSize: 669158
FileType: 2
MimeType: image/jpeg
SectionsFound: ANY_TAG, IFD0, THUMBNAIL, EXIF
COMPUTED[html]: width="1536" height="1024"
COMPUTED[Height]: 1024
COMPUTED[Width]: 1536
COMPUTED[IsColor]: 1
COMPUTED[ByteOrderMotorola]: 0
COMPUTED[ApertureFNumber]: f/4.0
COMPUTED[FocusDistance]: 1.07m
COMPUTED[Thumbnail.FileType]: 8
COMPUTED[Thumbnail.MimeType]: image/tiff
COMPUTED[Thumbnail.Height]: 64
COMPUTED[Thumbnail.Width]: 96
Make: Eastman Kodak Company
Model: KODAK DC265 ZOOM DIGITAL CAMERA (V01.00)
Orientation: 1
XResolution: 150/1
YResolution: 150/1
ResolutionUnit: 2
YCbCrPositioning: 1
Exif_IFD_Pointer: 190
THUMBNAIL[ImageWidth]: 96
THUMBNAIL[ImageLength]: 64
THUMBNAIL[BitsPerSample]: Array
THUMBNAIL[Compression]: 1
THUMBNAIL[PhotometricInterpretation]: 2
THUMBNAIL[StripOffsets]: 1748
THUMBNAIL[Orientation]: 1
THUMBNAIL[SamplesPerPixel]: 3
THUMBNAIL[RowsPerStrip]: 64
THUMBNAIL[StripByteCounts]: 18432
THUMBNAIL[XResolution]: 72/1
THUMBNAIL[YResolution]: 72/1
THUMBNAIL[PlanarConfiguration]: 1
THUMBNAIL[ResolutionUnit]: 2
ExposureTime: 1/250
FNumber: 400/100
ExifVersion: 0200
DateTimeOriginal: 1999:01:31 04:17:59
ComponentsConfiguration: *
CompressedBitsPerPixel: 24/10
ShutterSpeedValue: 800/100
ApertureValue: 400/100
ExposureBiasValue: 0/100
MaxApertureValue: 300/100
```

SubjectDistance: 107/100
MeteringMode: 2
LightSource: 0
Flash: 1
FocalLength: 80000/10000
MakerNote: * Eastman Kodak Company
FlashPixVersion: 0100
ColorSpace: 1
ExifImageWidth: 1536
ExifImageLength: 1024

Fetching an embedded thumbnail

```
<?  
Header('Content-type: image/tiff');  
echo exif_thumbnail('p0004557.jpg');  
>
```

A PDF Invoice

```

<?php
$pdf = pdf_new();
pdf_open_file($pdf);
pdf_set_info($pdf, "Author", "Rasmus Lerdorf");
pdf_set_info($pdf, "Title", "Sample Invoice");
pdf_set_info($pdf, "Creator", "See Author");
pdf_set_info($pdf, "Subject", "Sample Invoice");

$sizes = array('a4'=>'595x842', 'letter'=>'612x792', 'legal'=>'612x1008');

if(!isset($type)) $type='letter';
list($x,$y) = explode('x',$sizes[$type]);

$itemes = array(array('Our special low-cost widget that does everything','299.99'),
                array('Our special high-cost widget that does more','1899'),
                array('A blue widget','29.95'),
                array('And a red widget','49.95'),
                array('A yellow widget that makes noise','49.9'),
                array('And one that doesn\'t','999.95'),
                );

pdf_begin_page($pdf, $x, $y);

$im = pdf_open_jpeg($pdf, "php-big.jpg");
pdf_place_image($pdf, $im, 5, $y-72, 0.5);
pdf_close_image ($pdf,$im);

pdf_set_value($pdf, 'textrendering', 0); // fill

pdf_set_font($pdf, "Helvetica" , 12, winansi);
pdf_show_xy($pdf, 'Generic Evil Company Inc.',145,$y-20);
pdf_continue_text($pdf, '123 Main Street');
pdf_continue_text($pdf, 'Dark City, CA 98765');

pdf_set_font($pdf, "Helvetica" , 10, winansi);
pdf_show_xy($pdf, 'Helpless Customer Ltd.',20,$y-100);
pdf_continue_text($pdf, '2 Small Street');
pdf_continue_text($pdf, 'Little Town, ID 56789');

pdf_set_font($pdf, "Helvetica" , 10, winansi);
pdf_show_xy($pdf, 'Terms: Net 30',150,$y-100);
pdf_continue_text($pdf, 'PO #: 12345');

pdf_set_font($pdf, "Helvetica-Bold" , 30, winansi);
pdf_show_xy($pdf, " I N V O I C E ", $x-250,$y-112);

pdf_setcolor($pdf,'fill','gray',0.9,0,0,0);
pdf_rect($pdf,20,80,$x-40,$y-212);
pdf_fill_stroke($pdf);

$offset = 184; $i=0;
while($y-$offset > 80) {
    pdf_setcolor($pdf,'fill','gray',($i%2)?0.8:1,0,0,0);
    pdf_setcolor($pdf,'stroke','gray',($i%2)?0.8:1,0,0,0);
    pdf_rect($pdf,21,$y-$offset,$x-42,24);
    pdf_fill_stroke($pdf);
    $i++; $offset+=24;
}

pdf_setcolor($pdf,'fill','gray',0,0,0,0);
pdf_setcolor($pdf,'stroke','gray',0,0,0,0);
pdf_moveto($pdf, 20,$y-160);
pdf_lineto($pdf, $x-20,$y-160);

```

```

pdf_stroke($pdf);

pdf_moveto($pdf, $x-140,$y-160);
pdf_lineto($pdf, $x-140,80);
pdf_stroke($pdf);

pdf_set_font($pdf, "Times-Bold" , 18, winansi);
pdf_show_xy($pdf, "Item",30,$y-150);
pdf_show_xy($pdf, "Price", $x-100,$y-150);

pdf_set_font($pdf, "Times-Italic" , 15, winansi);

$offset = 177;
foreach($items as $item) {
    pdf_show_xy($pdf, $item[0],30,$y-$offset);
    pdf_show_boxed($pdf, '$'.number_format($item[1],2), $x-55, $y-$offset, 0, 0,
'right');
    $offset+=24;
    $total += $item[1];
}

pdf_set_font($pdf, "Times-Bold" , 17, winansi);
$offset+=24;
pdf_show_xy($pdf, 'Total',30,$y-$offset);
pdf_show_boxed($pdf, '$'.number_format($total,2), $x-55, $y-$offset, 0, 0,
'right');

pdf_end_page($pdf);
pdf_close($pdf);

$data = pdf_get_buffer($pdf);
header('Content-type: application/pdf');
header("Content-disposition: inline; filename=invoice.pdf");
header("Content-length: " . strlen($data));
echo $data;
?>

```

See <http://www.opaque.net/ming/>

```
<?
    $s = new SWFShape();
    $fp = fopen('php-big.jpg','r');
    $jpg = new SWFBitmap($fp);
    $w = $jpg->getWidth(); $h = $jpg->getHeight();

    $f = $s->addFill($jpg);
    $f->moveTo(-$w/2, -$h/2);
    $s->setRightFill($f);

    $s->movePenTo(-$w/2, -$h/2);
    $s->drawLine($w, 0);
    $s->drawLine(0, $h);
    $s->drawLine(-$w, 0);
    $s->drawLine(0, -$h);

    $p = new SWFSprite();
    $i = $p->add($s);

    for($step=0; $step<360; $step+=2) {
        $p->nextFrame();
        $i->rotate(-2);
    }

    $m = new SWFMovie();
    $i = $m->add($p);
    $i->moveTo(230,120);
    $m->setRate(100);
    $m->setDimension($w1.8, $h1.8);

    header('Content-type: application/x-shockwave-flash');
    $m->output();
?>
```

Output:

Flash + RSS/XML

```

<?php
require 'XML/RSS.php';

$r =& new XML_RSS('slashdot.rdf');
$r->parse();

$allItems = $r->getItems();
$itemCount = count($allItems);
$width = 1000;
$m = new SWFMovie();
$m->setDimension($width, 70);
$m->setBackground(0xcf, 0xcf, 0xcf);

$f = new SWFFont("../../../fonts/Techno.fdb");

$hit = new SWFShape();
$hit->setRightFill($hit->addFill(0,0,0));
$hit->movePenTo(-($width/2), -30);
$hit->drawLine($width, 0);
$hit->drawLine(0, 60);
$hit->drawLine(-$width, 0);
$hit->drawLine(0, -60);
$x = 0;

// build the buttons
foreach($allItems as $Item) {

    $title = $Item['title'];
    $link = $Item['link'];

    // get the text
    $t = new SWFText();
    $t->setFont($f);
    $t->setHeight(50);
    $t->setColor(0,0,0);
    $t->moveTo(-$f->getWidth($title)/2, 25);
    $t->addString($title);

    // make a button
    $b[$x] = new SWFButton();
    $b[$x]->addShape($hit, SWFBUTTON_HIT);
    $b[$x]->addShape($t, SWFBUTTON_OVER | SWFBUTTON_UP | SWFBUTTON_DOWN);
    $b[$x++]->addAction(new SWFAction("getURL('$link','_new');"), SWFBUTTON_MOUSEUP);
}

// display them
for($x=0; $x<$itemCount; $x++) {

    $i = $m->add($b[$x]);
    $i->moveTo($width/2,30);

    for($j=0; $j<=30; ++$j) {
        $i->scaleTo(sqrt(sqrt($j/30)));
        $i->multColor(1.0, 1.0, 1.0, $j/30);
        $m->nextFrame();
    }

    for($j=0; $j<=30; ++$j) {
        $i->scaleTo(sqrt(sqrt(1+($j/30))));
        $i->multColor(1.0, 1.0, 1.0, (30-$j)/30);
        $m->nextFrame();
    }
}

```



```
    $m->remove($i);  
}  
header('Content-type: application/x-shockwave-flash');  
$m->output();  
?>
```

Output:

Super-cool Dynamic Image Generator

Want to be cooler than all your friends? Well here it is!

First, set up an ErrorDocument 404 handler for your images directory.

```
<Directory /home/doc_root/images>
  ErrorDocument 404 /images/generate.php
</Directory>')
```

Then generate.php looks like this:

```
<?php
$filename = basename($_SERVER['REDIRECT_URL']);
if(preg_match('/^([\^?])([\^?])([\^_?])\.(.?)$/', $filename, $reg)) {
    $type = $reg[1];
    $text = $reg[2];
    $rgb = $reg[3];
    $ext = $reg[4];
}

if(strlen($rgb)==6) {
    $r = hexdec(substr($rgb,0,2));
    $g = hexdec(substr($rgb,2,2));
    $b = hexdec(substr($rgb,4,2));
} else $r = $g = $b = 0;

switch(strtolower($ext)) {
    case 'jpg':
        Header("Content-Type: image/jpg");
        break;
    case 'png':
    case 'gif': / We don't do gif - send a png instead /
        Header("Content-Type: image/png");
        break;
    default:
        break;
}

switch($type) {
    case 'solid':
        $im = imagecreatetruecolor(80,80);
        $bg = imagecolorallocate($im, $r, $g, $b);
        imagefilledrectangle($im,0,0,80,80,$bg);
        break;
    case 'button':
        $ssi = 32; $font = "php";
        $im = imagecreatefrompng('blank_wood.png');
        $tsize = imagettfbbox($ssi,0,$font,$text);
        $dx = abs($tsize[2]-$tsize[0]);
        $dy = abs($tsize[5]-$tsize[3]);
        $x = ( imagesx($im) - $dx ) / 2;
        $y = ( imagesy($im) - $dy ) / 2 + $dy;
        $white = ImageColorAllocate($im,255,255,255);
        $black = ImageColorAllocate($im,$r,$g, $b);
        ImageTTFText($im, $ssi, 0, $x, $y, $white, $font, $text);
        ImageTTFText($im, $ssi, 0, $x+2, $y, $white, $font, $text);
        ImageTTFText($im, $ssi, 0, $x, $y+2, $white, $font, $text);
        ImageTTFText($im, $ssi, 0, $x+2, $y+2, $white, $font, $text);
        ImageTTFText($im, $ssi, 0, $x+1, $y+1, $black, $font, $text);
        break;
}
Header("HTTP/1.1 200 OK");
$dest_file = dirname($_SERVER['SCRIPT_FILENAME']).'/'.$filename;
switch(strtolower($ext)) {
```

```
case 'png':
case 'gif':
    @ImagePNG($im,$dest_file);
    ImagePNG($im);
    break;
case 'jpg':
    @ImageJPEG($im,$dest_file);
    ImageJPEG($im);
    break;
}
?>
```

The URL, http://localhost/images/button_test_000000.png produces this image:



\$PATH_INFO is your friend when it comes to creating clean URLs. Take for example this URL:

```
http://www.company.com/products/routers
```

If the Apache configuration contains this block:

```
<Location "/products">  
    ForceType application/x-httpd-php  
</Location>
```

Then all you have to do is create a PHP script in your DOCUMENT_ROOT named 'products' and you can use the \$PATH_INFO variable which will contain the string, '/routers', to make a DB query.

Apache's ErrorDocument directive can come in handy. For example, this line in your Apache configuration file:

```
ErrorDocument 404 /error.php
```

Can be used to redirect all 404 errors to a PHP script. The following server variables are of interest:

- o \$REDIRECT_ERROR_NOTES - File does not exist: /docroot/bogus
- o \$REDIRECT_REQUEST_METHOD - GET
- o \$REDIRECT_STATUS - 404
- o \$REDIRECT_URL - /docroot/bogus

Don't forget to send a 404 status if you choose not to redirect to a real page.

```
<? Header('HTTP/1.0 404 Not Found'); ?>
```

Interesting uses

- o Search for closest matching valid URL and redirect
- o Use attempted url text as a DB keyword lookup
- o Funky caching

An interesting way to handle caching is to have all 404's redirected to a PHP script.

```
ErrorDocument 404 /generate.php
```

Then in your generate.php script use the contents of \$REDIRECT_URI to determine which URL the person was trying to get to. In your database you would then have fields linking content to the URL they affect and from that you should be able to generate the page. Then in your generate.php script do something like:

```
<?php
    $s = $REDIRECT_URI;
    $d = $DOCUMENT_ROOT;
    // determine requested uri
    $uri = substr($s, strpos($s,$d) + strlen($d)+1);
    ob_start(); // Start buffering output
    // ... code to fetch and output content from DB
    $data = ob_get_contents();
    $fp = fopen("$DOCUMENT_ROOT/$uri", 'w');
    fputs($fp,$data);
    fclose($fp);
    ob_end_flush(); // Flush and turn off buffering
?>
```

So, the way it works, when a request comes in for a page that doesn't exist, generate.php checks the database and determines if it should actually exist and if so it will create it and respond with this generated data. The next request for that same URL will get the generated page directly. So in order to refresh your cache you simply have to delete the files.

Don't use a regex if you don't have to

PHP has a rich set of string manipulation functions - use them!

```
BAD: <? $new = ereg_replace("-", "_", $str); ?>
```

```
GOOD:<? $new = str_replace("-", "_", $str); ?>
```

```
BAD: <? preg_match('/(\\..*?)$/', $str, $reg); ?>
```

```
GOOD:<? substr($str, strrpos($str, '.')); ?>
```

Use References if you are passing large data structs around to save memory

There is a tradeoff here. Manipulating references is actually a bit slower than making copies of your data, but with references you will be using less memory. So you need to determine if you are cpu or memory bound to decide whether to go through and look for places to pass references to data instead of copies.

Use Persistent Database connections

Some database are slower than others at establishing new connections. The slower it is, the more of an impact using persistent connections will have. But, keep in mind that persistent connections will sit and tie up resources even when not in use. Watch your resource limits as well. For example, by default Apache's

Using MySQL? Check out `mysql_unbuffered_query()`

Use it exactly like you would `mysql_query()`. The difference is that instead of waiting for the entire query to finish and storing the result in the client API, an unbuffered query makes results available to you as soon as possible and they are not allocated in the client API. You potentially get access to your data quicker, use a lot less memory, but you can't use `mysql_num_rows()` on the result resource and it is likely to be slightly slower for small selects.

Hey Einstein!

Don't over-architect things. If your solution seems complex to you, there is probably a simpler and more obvious approach. Take a break from the computer and go out into the big (amazingly realistic) room and think about something else for a bit.

Check these

- o Don't fetch 10,000 rows and only use 5, use a LIMIT clause
- o Make sure you have keys on the right columns
- o --log-slow-queries[=file_name] can help here
- o So can --log-long-format to show non-indexed queries
- o Use mysqldumpslow to look at these

A non-indexed query

```
<?
mysql_connect('localhost');
mysql_select_db('foo');
$result=mysql_query(
    "select * from users where ts>0");
?>
```

And an indexed query

```
<?
$result = mysql_query(
    "select * from users where id='rasmus'");
echo "<pre>";
system('mysqldumpslow');
echo "</pre>\n";
?>
```

Output:

```
Count: 19  Time=0.00s (0s)  Lock=0.00s (0s)  Rows=2.0 (38),
nobody[nobody]@localhost
  select id, email,
  date_format(ts,'S') as d
  from users order by ts

Count: 18  Time=0.00s (0s)  Lock=0.00s (0s)  Rows=2.0 (36),
nobody[nobody]@localhost
  select * from users

Count: 1  Time=0.00s (0s)  Lock=0.00s (0s)  Rows=2.0 (2), nobody[nobody]@localhost
  select * from comments

Count: 3  Time=0.00s (0s)  Lock=0.00s (0s)  Rows=2.0 (6), nobody[nobody]@localhost
  select * from users where ts>N

Count: 18  Time=0.00s (0s)  Lock=0.00s (0s)  Rows=2.0 (36),
nobody[nobody]@localhost
  select * from users order by id

Count: 13  Time=0.00s (0s)  Lock=0.00s (0s)  Rows=2.7 (35),
nobody[nobody]@localhost
  select * from comments order by ts desc
```


User Defined Functions

You can execute PHP code as a user defined function inside MySQL.

Loading the PHP UDF library

```
CREATE FUNCTION php RETURNS STRING SONAME 'myphp.so';
```

Hello World

```
mysql> SELECT php('hello, world!');
+-----+
| php('hello, world!') |
+-----+
| hello, world!        |
+-----+
1 row in set (0.36 sec)
```

```
mysql> SELECT php('strftime("%c")');
+-----+
| php('strftime("%c")') |
+-----+
| Thu Oct 24 18:12:04 2002 |
+-----+
1 row in set (0.36 sec)
```

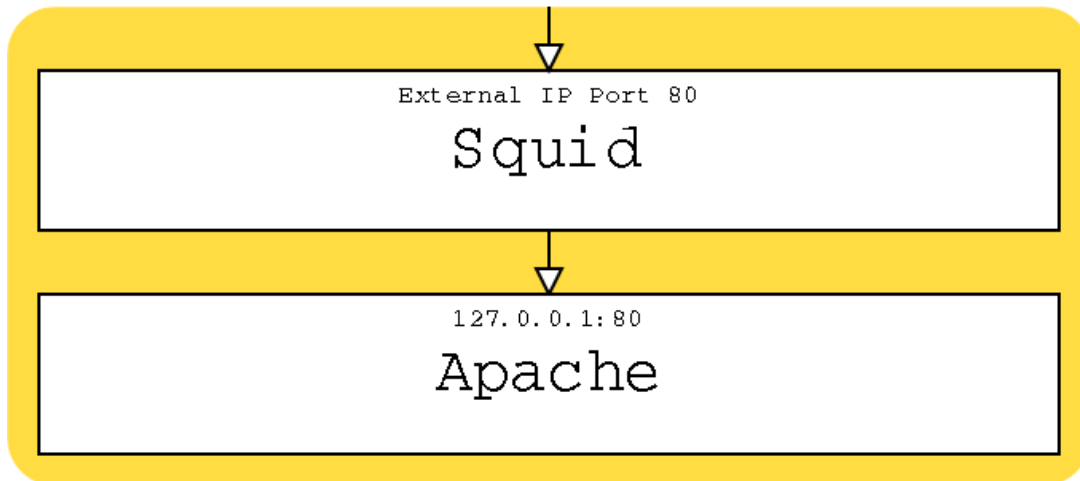
Applying PHP on a selected column

```
mysql> CREATE TABLE test (s varchar(255));
mysql> insert into test values ('iguana'),('turtle'),('aardvark');
mysql> select s,php('strlen($argv[1]'),'s) FROM test;
+-----+-----+
| s          | php('strlen($argv[1]'),'s) |
+-----+-----+
| iguana    | 6                            |
| turtle    | 6                            |
| aardvark  | 8                            |
+-----+-----+
3 rows in set (1.07 sec)
```

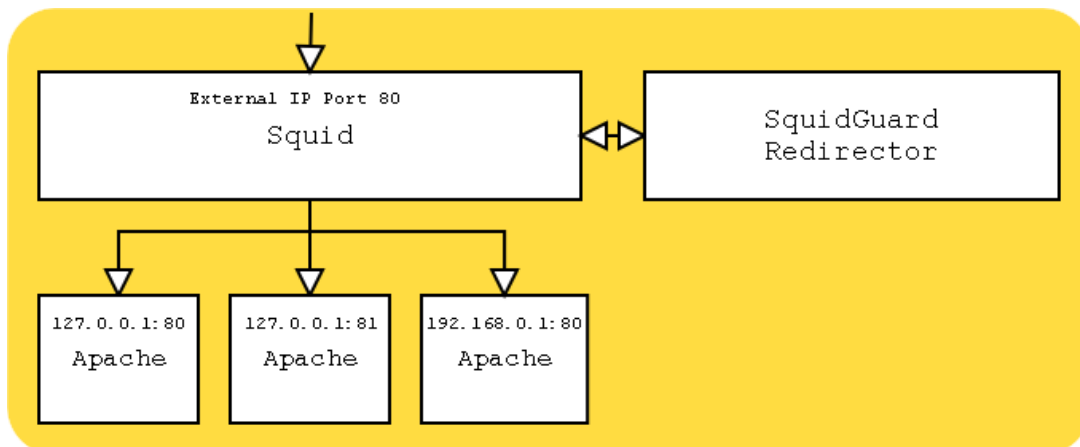
Oh my!

```
mysql> create table code (name varchar(255), code mediumblob);
mysql> insert into code values ('capitalize','strtoupper($argv[1])');
mysql> select php(code,s) FROM test,code WHERE code.name = 'capitalize';
+-----+
| php(code,s) |
+-----+
| IGUANA      |
| TURTLE      |
| AARDVARK    |
+-----+
```

For really busy sites, a reverse proxy like Squid is magical! Either run it as a single-server accelerator:



Or as a front-end cache to a number of local or remote servers:



Note:

Watch out for any use of `$REMOTE_ADDR` in your PHP scripts. Use `$HTTP_X_FORWARDED_FOR` instead.

Make it listen to port 80 on our external interface:

```
http_port 198.186.203.51:80
```

If we don't do cgi-bin stuff, comment these out:

```
#acl QUERY urlpath_regex cgi-bin  
#no_cache deny QUERY
```

If we have plenty of RAM, bump this up a bit:

```
cache_mem 16MB  
maximum_object_size 14096 KB
```

Specify where to store cached files (size in Megs, level 1 subdirs, level 2 subdirs)

```
cache_dir ufs /local/squid/cache 500 16 256
```

Get rid of the big store.log file:

```
cache_store_log none
```

Set our SNMP public community string:

```
acl snmppublic snmp_community public
```

Get rid of "allow all" and use list of hosts we are blocking (1 ip per line):

```
#http_access allow all  
acl forbidden src "/local/squid/etc/forbidden"  
http_access allow !forbidden
```

Set user/group squid should run as:

```
cache_effective_user squid  
cache_effective_group daemon
```

Single-server reverse proxy setup (set up Apache to listen to port 80 on the loopback):

```
httpd_accel_host 127.0.0.1  
httpd_accel_port 80  
httpd_accel_single_host on  
httpd_accel_uses_host_header on
```

Only allow localhost access through snmp:

```
snmp_access allow snmppublic localhost
```

There are two primary ways to configure Squid to be a load balancer.

Private /etc/hosts

The easiest is to simply list multiple ips for the `httpd_accel_host` in your `/etc/hosts` file and Squid will automatically round-robin across the backend servers.

Use `cache_peer`

This is more complex. Squid has advanced support for communicating with other caches. It just so happens that this communication happens over HTTP so you can set Squid up to treat the web servers you wish to load balance as peer caches. The configuration would look something like this:

```
httpd_accel_host www.visible-domain.com
httpd_accel_uses_host_header on
never_direct allow all
cache_peer server1 parent 80 0 no-query round-robin
cache_peer server2 parent 80 0 no-query round-robin
cache_peer server3 parent 80 0 no-query round-robin
```

`no-query` in the above tells Squid not to try to send ICP (Internet Cache Protocol) requests to the Apache servers. You could turn on the echo port on each server and redirect these ICP requests to there which would make this system automatically detect a server that is down.

Avoiding Redirectors

It is generally a good idea to avoid external redirectors. A lot of things can be done directly in Squid's config file. For example:

```
acl domain1 dstdomain www.domain1.com
acl domain2 dstdomain www.domain2.com
acl domain3 dstdomain www.domain3.com
cache_peer_access server1 allow domain1
cache_peer_access server2 allow domain2
cache_peer_access server3 allow domain3
cache_peer_access server1 deny all
cache_peer_access server2 deny all
cache_peer_access server3 deny all
```

This would configure Squid to send requests for pages on certain domains to certain backend servers. These backend servers could actually be aliases to different ips on the same server if you wanted to run multiple Apache instances on different ports on the same box.

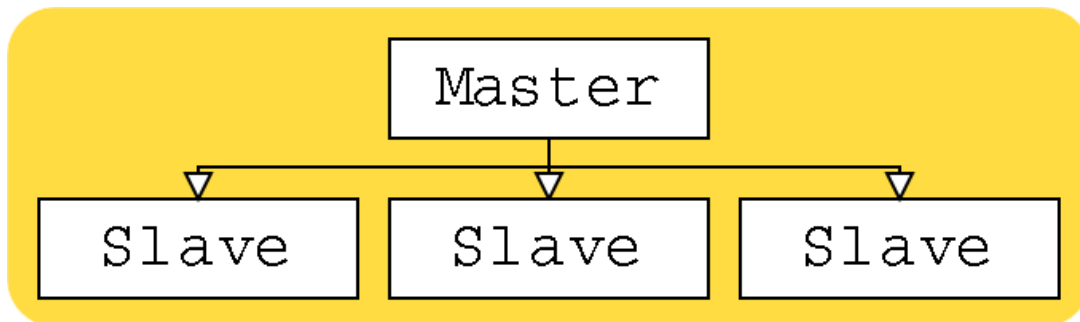
The Easy (And Expensive) Solution

You can buy dedicated boxes that do all sorts of advanced load balancing for you. Some popular ones are:

- o Cisco CSS
- o Foundry ServerIron
- o Intel Netstructure
- o F5 BIG-IP

If you are on Linux you could also try LVS. See www.LinuxVirtualServer.org.

As of version 3.23.15 (try to use 3.23.29 or later), MySQL supports one-way replication. Since most web applications usually have more reads than writes, an architecture which distributes reads across multiple servers can be very beneficial.



In typical MySQL fashion, setting up replication is trivial. On your master server add this to your "my.cnf" file:

```
[mysqld]
log-bin
server-id=1
```

And add a replication user id for slaves to log in as:

```
GRANT FILE ON . TO repl%"%" IDENTIFIED BY 'foobar';
```

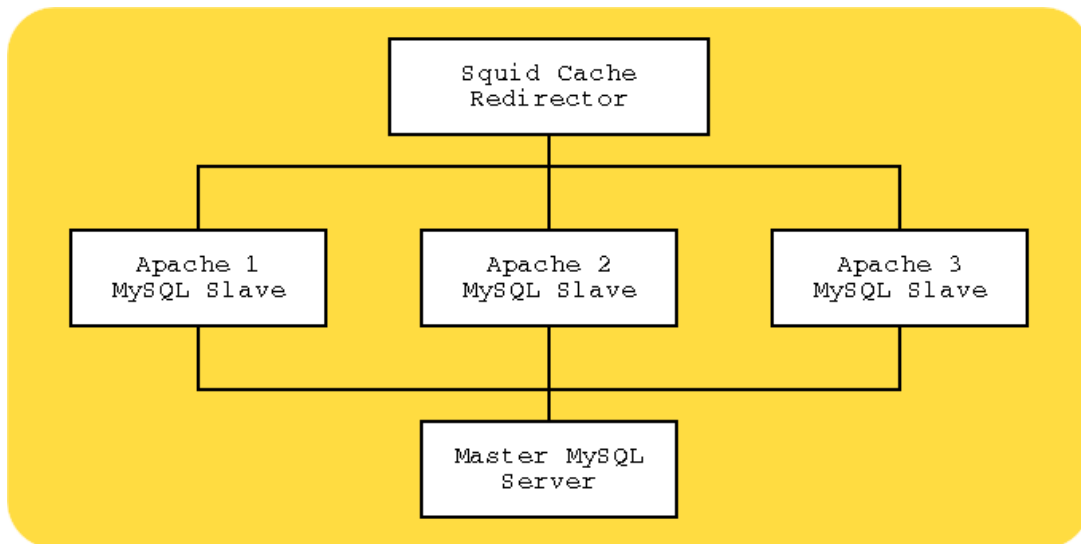
If you are using MySQL 4.0.2 or later, replace FILE with REPLICATION SLAVE in the above. Then on your slave servers:

```
[mysqld]
set-variable = max_connections=200
log-bin
master-host=192.168.0.1
master-user=repl
master-password=foobar
master-port=3306
server-id=2
```

Make sure each slave has its own unique server-id. And since these will be read-only slaves, you can start them with these options to speed them up a bit:

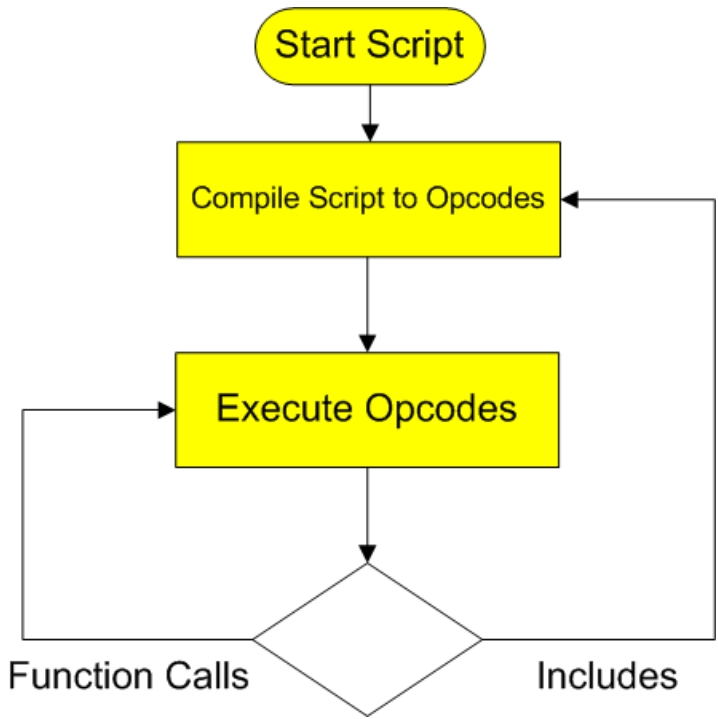
```
--skip-bdb --low-priority-updates
--delay-key-write-for-all-tables
```

Stop your master server. Copy the table files to each of your slave servers. Restart the master, then start all the slaves. And you are done. Combining MySQL replication with a Squid reverse cache and redirector and you might have an architecture like this:

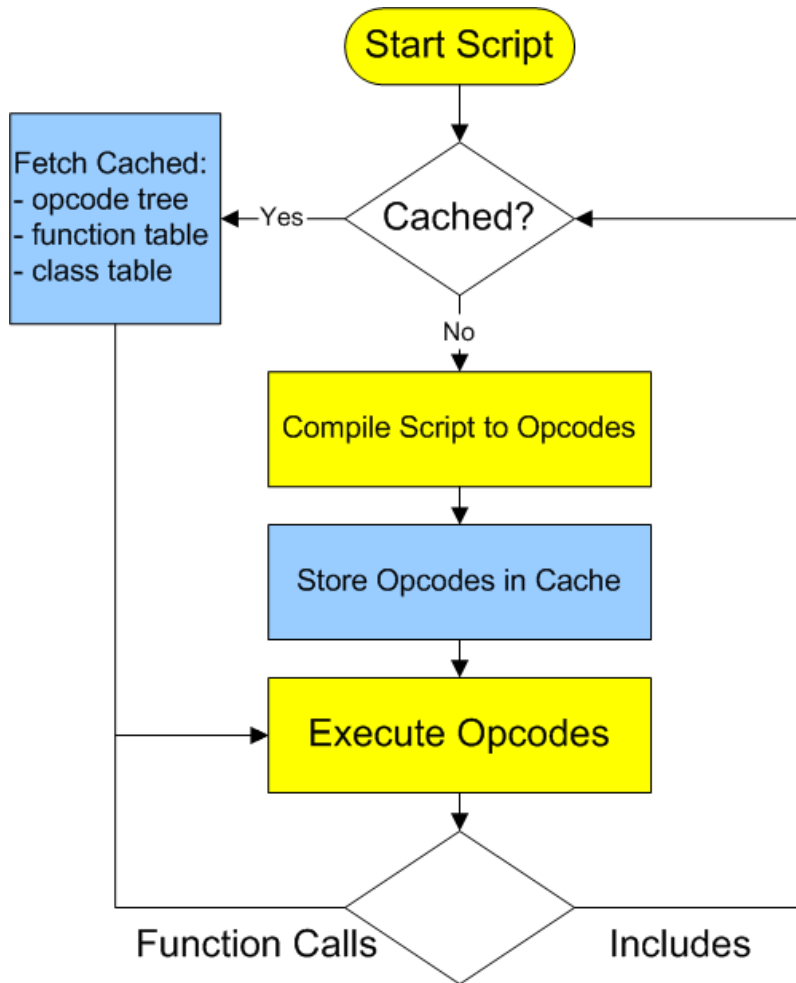


You would then write your application to send all database writes to the master server and all reads to the local slave. It is also possible to set up two-way replication, but you would need to supply your own application-level logic to maintain atomicity of distributed writes. And you lose a lot of the advantages of this architecture if you do this as the writes would have to go to all the slaves anyway.

Standard PHP



PHP with an Opcode Cache



There are a number of them out there.

- o APC - open source
- o IonCube Accelerator - free, but closed source
- o Zend Cache - commercial

Installation

Typically very trivial. For IonCube, for example, in your php.ini add:

```
zend_extension = "/usr/local/lib/php/php_accelerator_1.3.3r2.so"
```

So why isn't this just built into standard PHP?

This gets asked often, and although not a good answer, the answer is that since it wasn't in PHP from the start a number of competing plugin cache systems were developed and choosing one over another at this point would effectively chop these projects, both commercial and free, off at their knees. This way they can compete and innovate against each other at the cost of duplicated effort.

Let's have a look at how we might benchmark and subsequently tune a PHP server.

<http://apc.communityconnect.com/sources/apc-cvs.tar.gz>

<http://www.ioncube.com>

```
CPU: Pentium III 700MHz  
RAM: 128M
```

Our test script includes 2 files, defines 3 functions, loops a bit and outputs about 6k of data.

For testing I am using `http_load` from `acme.com` and the default `SendBufferSize`

```
# To change this on Linux, cat a larger number  
# into /proc/sys/net/core/wmem_max in your  
# httpd startup script  
SendBufferSize 65535
```

An http_load run

```
% http_load -parallel 5 -fetches 5000 url.txt
5000 fetches, 5 max parallel, 2.954e+07 bytes, in 12.5082 seconds
5908 mean bytes/connection
399.739 fetches/sec, 2.36166e+06 bytes/sec
msecs/connect: 0.242907 mean, 9.246 max, 0.195 min
msecs/first-response: 10.7645 mean, 130.136 max, 2.114 min
HTTP response codes:
code 200 -- 5000
```

No Opcode cache	254 requests/sec
With APC2 cache	399 requests/sec
With IonCube (phpa) cache	350 requests/sec

It is also interesting to look at what your server is doing while you hammer it with http_load:

Load without opcode cache

procs				memory			swap		io			system			cpu	
r	b	w	swpd	free	buff	cache	si	so	bi	bo	in	cs	us	sy	id	
11	0	0	90000	3844	4928	73152	0	0	0	0	2854	450	87	13	0	
11	0	1	90000	5280	4944	73128	0	8	0	160	2934	450	88	12	0	
13	0	0	90000	3928	4944	73164	0	0	0	0	3002	595	87	13	0	
12	0	1	90000	3892	4944	73184	0	0	0	0	2928	515	82	18	0	
9	0	0	90000	3896	4944	73200	0	0	0	0	2970	602	79	21	0	
12	0	1	90000	3840	4952	73220	0	0	0	12	2948	517	80	20	0	
11	0	1	90000	3956	4952	73240	0	0	0	0	2916	522	91	9	0	
9	0	1	90000	3952	4952	73256	0	0	0	0	2996	547	89	11	0	
11	0	1	90000	3916	4952	73276	0	0	0	0	2949	534	81	19	0	
13	0	1	90000	3896	4952	73296	0	0	0	0	2886	516	86	14	0	
11	0	1	90000	3876	4960	73312	0	0	0	12	2887	547	90	10	0	
11	0	1	90000	3864	4960	73332	0	0	0	0	2931	531	85	15	0	
12	0	1	90000	3844	4960	73348	0	0	0	0	3013	579	89	11	0	
12	0	1	90000	3952	4960	73368	0	0	0	0	2999	508	87	13	0	
12	0	1	90000	3928	4968	73388	0	0	0	12	3022	600	79	21	0	

Load with APC2 cache

procs				memory			swap		io			system			cpu	
r	b	w	swpd	free	buff	cache	si	so	bi	bo	in	cs	us	sy	id	
11	0	0	90000	3904	4712	72212	0	0	0	0	4494	596	78	22	0	
10	0	1	90000	5280	4680	72160	0	0	0	0	4422	596	77	23	0	
11	0	1	90000	5176	4688	72192	0	0	0	12	4500	791	85	15	0	
9	0	0	90000	5224	4688	72092	0	0	0	0	4541	702	79	21	0	
11	0	1	90000	5152	4688	72120	0	0	0	0	4495	885	88	12	0	
10	0	1	90000	5216	4688	72020	0	0	0	0	4513	674	73	27	0	
11	0	1	90000	5132	4688	72048	0	0	0	0	4477	794	82	18	0	
11	0	1	90000	5200	4688	71952	0	0	0	128	4476	701	80	20	0	
10	0	1	90000	5296	4688	71852	0	0	0	0	4486	770	76	24	0	
11	0	1	90000	5224	4688	71880	0	0	0	0	4524	698	82	18	0	
11	0	0	90000	5144	4688	71908	0	0	0	0	4481	788	78	22	0	
11	0	1	90000	5196	4688	71808	0	0	0	0	4504	684	74	26	0	
11	0	1	90000	5116	4688	71836	0	0	0	128	4463	792	80	20	0	
10	0	1	90000	5172	4688	71740	0	0	0	0	4539	681	80	20	0	

```
12 0 1 90000 5228 4688 71640 0 0 0 0 4503 747 74 26 0
```

Load with IonCube cache

procs				memory			swap		io			system			cpu	
r	b	w	swpd	free	buff	cache	si	so	bi	bo	in	cs	us	sy	id	
11	0	1	86084	5196	6284	54104	0	0	0	4	4078	632	75	25	0	
12	0	1	86084	5132	6284	54132	0	0	0	0	4024	593	78	22	0	
12	0	1	86084	5204	6284	54052	32	4	32	4	3996	705	66	34	0	
12	0	1	86084	5180	6292	54076	0	0	0	140	3979	678	74	26	0	
11	0	1	86084	5148	6292	54104	0	0	0	0	3980	740	70	30	0	
10	0	1	86084	5256	6292	54000	0	0	0	0	3945	660	82	18	0	
10	0	1	86084	5228	6292	54024	0	0	0	0	4025	663	75	25	0	
12	0	1	86084	5200	6292	54052	0	0	0	0	3967	692	82	18	0	
11	0	1	86084	5180	6292	54076	0	0	0	0	3994	688	76	24	0	
11	0	1	86084	5156	6292	54100	0	0	0	0	4023	641	71	29	0	
11	0	1	86144	5264	6292	54012	0	0	0	0	3993	719	78	22	0	
10	0	1	86144	5232	6292	54036	0	0	0	0	3967	655	80	20	0	
12	0	1	86144	5208	6292	54060	0	0	0	0	3946	732	78	22	0	
11	0	0	86144	5180	6292	54088	0	0	0	128	4011	637	76	24	0	
10	0	1	86144	3856	6292	54112	0	0	0	0	3983	664	80	20	0	
11	0	1	86160	3976	6292	54080	0	28	0	28	3932	639	80	20	0	

Our benchmark test was deliberately designed to have quite a bit of PHP processing and not a whole lot of output. 6k is somewhat small for a web page. If instead we have a whole lot of output, chances are we will be io-bound instead of cpu-bound. If you are io-bound, there is little sense in optimizing at the PHP level.

Evidence of an io-bound test

procs				memory			swap		io			system			cpu	
r	b	w	swpd	free	buff	cache	si	so	bi	bo	in	cs	us	sy	id	
4	0	0	94928	5204	2228	29812	0	0	0	0	4813	1660	19	24	58	
7	0	0	94928	5200	2228	29812	0	0	0	0	4828	1614	7	33	60	
6	0	0	94928	5176	2248	29812	0	0	4	24	4823	1688	13	20	68	
7	0	0	94928	5172	2248	29816	0	0	0	0	4815	1670	8	38	54	
5	0	0	94928	5168	2248	29816	0	0	0	0	4840	1647	12	23	66	
5	0	0	94928	5168	2248	29816	0	0	0	0	4821	1663	12	28	60	
5	0	0	94928	5164	2248	29820	0	0	0	0	4843	1664	11	25	64	
5	0	0	94928	5152	2256	29820	0	0	0	12	4791	1687	9	26	65	
4	0	0	94936	5276	2256	29820	0	8	0	8	4853	1665	14	25	62	
6	0	0	94936	5284	2256	29824	0	0	0	0	4805	1670	7	28	65	
6	0	0	94936	5268	2256	29824	0	0	0	0	4823	1676	8	25	67	
6	0	0	94936	5244	2256	29824	24	0	24	0	4803	1700	9	32	59	
5	0	0	94936	5228	2264	29824	0	0	0	12	4865	1762	12	23	66	

Things to try if you are io-bound

```
[php.ini]
output_handler = ob_gzhandler
```

```
[httpd.conf]
LoadModule gzip_module lib/apache/mod_gzip.so
```

include_path

```
include_path = "/usr/share/pear:."
```

```
<?php
    include './template_helpers.inc';
    include 'business_logic.inc';
?>
```

open_basedir

```
open_basedir = "/usr/share/htdocs/:/usr/share/pear/"
```

open_basedir checking adds some extra syscalls to every file operation. It can be useful, but it is rather expensive, so turn it off if you don't think you need it.

variables_order

```
variables_order = "GPC"
```

```
<?php
    echo $_SERVER['DOCUMENT_ROOT'];
    echo getenv('DOCUMENT_ROOT');
?>
```

PHP 4.3 w/ APC2 and EGPCS	399 requests/sec
PHP 4.3 w/ APC2 and GPC	513 requests/sec

Why Profile?

Because your assumptions of how things work behind the scenes are not always correct. By profiling your code you can identify where the bottlenecks are quantitatively.

How?

PEAR/Pecl to the rescue!

```
www:~> pear install apd
downloading apd-0.4pl.tgz ...
...done: 39,605 bytes
16 source files, building
running: phpize
PHP Api Version      : 20020918
Zend Module Api No   : 20020429
Zend Extension Api No: 20021010
building in /var/tmp/pear-build-root/apd-0.4pl
running: /tmp/tmpFrFlAqf/apd-0.4pl/configure
running: make
apd.so copied to /tmp/tmpFrFlAqf/apd-0.4pl/apd.so
install ok: apd 0.4pl
```

Woohoo!

```
www:~> pear info apd
About apd-0.4pl
=====
```

Package	apd	
Summary	A full-featured engine-level profiler/debugger	
Description	APD is a full-featured profiler/debugger that is loaded as a zend_extension. It aims to be an analog of C's gprof or Perl's Devel::DProf.	
Maintainers	George Schlossnagle <george@omniti.com> (lead)	
Version	0.4pl	
Release Date	2002-11-25	
Release License	PHP License	
Release State	stable	
Release Notes	Fix for pre-4.3 versions of php	
Last Modified	2002-12-02	

```
www:~> pear config-show
Configuration:
=====
```

PEAR executables directory	bin_dir	/usr/local/bin
PEAR documentation directory	doc_dir	/usr/local/lib/php/docs
PHP extension directory	ext_dir	/usr/local/lib/php/extensions/no-debug-non-zts-20020429
PEAR directory	php_dir	/usr/local/lib/php
PEAR Installer cache directory	cache_dir	/tmp/pear/cache
PEAR data directory	data_dir	/usr/local/lib/php/data
PEAR test directory	test_dir	/usr/local/lib/php/tests
Cache TimeToLive	cache_ttl	<not set>
Preferred Package State	preferred_state	stable
Unix file mask	umask	18

Debug Log Level	verbose	1
HTTP Proxy Server Address	http_proxy	<not set>
PEAR server	master_server	pear.php.net
PEAR password (for maintainers)	password	<not set>
PEAR username (for maintainers)	username	<not set>


```
www:~> cd /usr/local/lib/php
www:/usr/local/lib/php> ln -s extensions/no-debug-non-zts-20020429/apd.so apd.so
```

Then in your php.ini file:

```
zend_extension = "/usr/local/lib/php/apd.so"
apd.dumpdir = /tmp
```

It isn't completely transparent. You need to tell the profiler when to start profiling. At the top of a script you want to profile, add this call:

```
<?php
apd_set_pprof_trace();
?>
```

The use the command-line tool called pprofp:

```
www: ~> pprofp
pprofp <flags> <trace file>
Sort options
-a          Sort by alphabetic names of subroutines.
-l          Sort by number of calls to subroutines
-m          Sort by memory used in a function call.
-r          Sort by real time spent in subroutines.
-R          Sort by real time spent in subroutines (inclusive of child calls).
-s          Sort by system time spent in subroutines.
-S          Sort by system time spent in subroutines (inclusive of child
calls).
-u          Sort by user time spent in subroutines.
-U          Sort by user time spent in subroutines (inclusive of child calls).
-v          Sort by average amount of time spent in subroutines.
-z          Sort by user+system time spent in subroutines. (default)

Display options
-c          Display Real time elapsed alongside call tree.
-i          Suppress reporting for php builtin functions
-O <cnt>   Specifies maximum number of subroutines to display. (default 15)
-t          Display compressed call tree.
-T          Display uncompressed call tree.
```

```
% ls -latr /tmp/pprofp.*
-rw-r--r--  1 nobody  nobody      16692 Dec  3 01:19 /tmp/pprof.04545
```

```
% pprofp -z /tmp/pprof.04545
Trace for /home/rasmus/phpweb/bm.php
Total Elapsed Time = 0.01
Total System Time = 0.00
Total User Time = 0.01
```

Name	Real %Time (excl/cumm)	User (excl/cumm)	System (excl/cumm)	Calls	secs/ call	cumm s/call	Memory Usage
100.0	0.01	0.01	0.00	2	0.0050	0.0050	3488
include							
0.0	0.00	0.00	0.00	1	0.0000	0.0000	48 c
0.0	0.00	0.00	0.00	21	0.0000	0.0000	1168 b
0.0	0.00	0.00	0.00	1	0.0000	0.0000	56 a
0.0	0.00	0.00	0.00	1	0.0000	0.0000	0
main							
0.0	0.00	0.00	0.00	1	0.0000	0.0000	1080
range							

```
Trace for /home/rasmus/phpweb/index.php
```

Total Elapsed Time = 0.69
 Total System Time = 0.01
 Total User Time = 0.08

Name	Real (excl/cumm)	User (excl/cumm)	System (excl/cumm)	Calls	secs/ call	cumm s/call	Memory Usage
require_once	33.3 0.11 0.13	0.02 0.03	0.01 0.01	7	0.0043	0.0057	298336
feof	22.2 0.02 0.02	0.02 0.02	0.00 0.00	183	0.0001	0.0001	-33944
define	11.1 0.01 0.01	0.01 0.01	0.00 0.00	3	0.0033	0.0033	-14808
fgetc	11.1 0.04 0.04	0.01 0.01	0.00 0.00	182	0.0001	0.0001	112040
getimagesize	11.1 0.25 0.25	0.01 0.01	0.00 0.00	6	0.0017	0.0017	3768
sprintf	11.1 0.01 0.01	0.01 0.01	0.00 0.00	55	0.0002	0.0002	2568
printf	0.0 0.00 0.00	0.00 0.00	0.00 0.00	7	0.0000	0.0000	-136
htmlspecialchars	0.0 0.00 0.00	0.00 0.00	0.00 0.00	1	0.0000	0.0000	136
mirror_provider_url	0.0 0.00 0.00	0.00 0.00	0.00 0.00	1	0.0000	0.0000	-16
spacer	0.0 0.00 0.00	0.00 0.00	0.00 0.00	7	0.0000	0.0000	112
delim	0.0 0.00 0.00	0.00 0.00	0.00 0.00	10	0.0000	0.0000	-552
mirror_provider	0.0 0.00 0.00	0.00 0.00	0.00 0.00	1	0.0000	0.0000	112
print_link	0.0 0.00 0.00	0.00 0.00	0.00 0.00	20	0.0000	0.0000	-624
have_stats	0.0 0.00 0.00	0.00 0.00	0.00 0.00	1	0.0000	0.0000	24
make_submit	0.0 0.00 0.00	0.00 0.00	0.00 0.00	1	0.0000	0.0000	-72
strchr	0.0 0.00 0.00	0.00 0.00	0.00 0.00	2	0.0000	0.0000	112
filesize	0.0 0.08 0.08	0.00 0.00	0.00 0.00	2	0.0000	0.0000	168
commonfooter	0.0 0.00 0.00	0.00 0.00	0.00 0.00	1	0.0000	0.0000	-16
download_link	0.0 0.00 0.11	0.00 0.00	0.00 0.00	2	0.0000	0.0000	0
make_image	0.0 0.00 0.25	0.00 0.01	0.00 0.00	6	0.0000	0.0017	208

New General Features

- o Get rid of stdio completely
- o Stream servers and contexts
- o ADT (Advanced Data Types) extension
- o Mono Extension
- o RPC Extension
- o Bundled Curl support

New OO Features

- o Better Object Overloading
- o Namespaces
- o try/catch exceptions
- o __construct() and __destruct() methods
- o Private, Protected and Public methods and properties
- o Static methods and properties
- o Final properties
- o Constant properties
- o __autoload
- o Abstract classes
- o Interfaces
- o class type hinting
- o SPL

New Engine Features

- o Opcode overloading
- o Ability to add new opcodes

Home Page: <http://www.php.net>

Manual: <http://php.net/manual>

Tutorial: <http://php.net/tut.php>

Books: <http://php.net/books.php>

Index

The Good Old Days	2
The Perl alternative	3
The PHP Approach	4
Solution	5
The PHP Way	6
PHP Usage Growth	7
App Architecture	8
A Simple Guestbook	9
SQL Example	10
DB-driven Guestbook	11
DB-driven Guestbook	12
GD 1/2	13
ImageColorAt	14
GD 1/2	15
Text	16
TTF Text	17
EXIF	19
PDFs on-the-fly	21
Ming-Flash	23
More Ming	24
Cool!	26
\$PATH_INFO	28
ErrorDocument	29
Funky Caching	30
Optimization	31
Optimizing MySQL Usage	32
UDF	33
Squid	34
Squid Configuration	35
Load Balancing with Squid	36
Load Balancing	37
MySQL Replication	38
PHP Opcode Caches	40
PHP Opcode Caches	41
PHP Opcode Caches	42
Benchmarking	43
Benchmarking Results	44
Tuning	46
Profiling PHP	47
Profiling PHP	49
PHP5	51
Resources	52