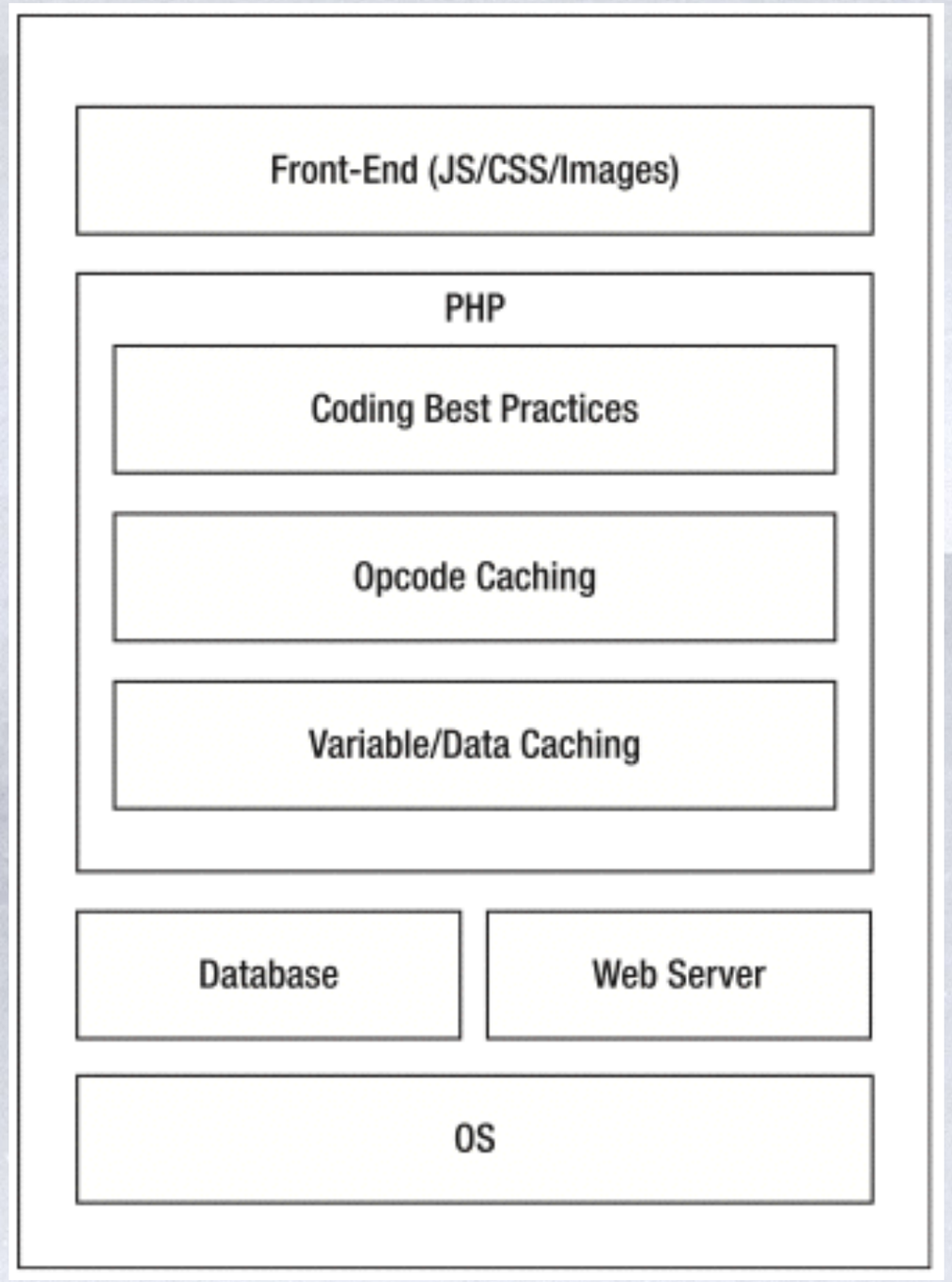
Lecture 11
Performance

Play

Outline

- PHP Application Performance
- Large Scale Web Application

The PHP Application Stack



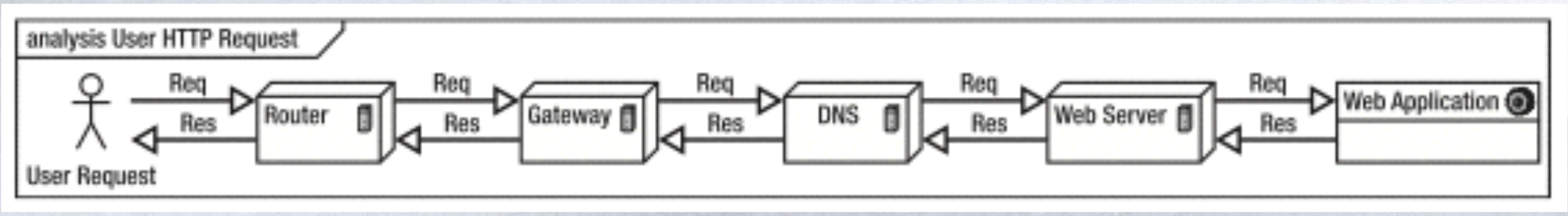
Benchmarking Utilities

- Apache Bench
  - ab utility bundled with Apache
- Siege
  - http://www.joedog.org/JoeDog/Siege
- http_load (Excellent for latency tests)
  - http://www.acme.com/software/http_load/

Affecting Your Benchmark Figures

- Geographical location
- Network issues
- Response size
- Code processing
- Browser behavior
- Web server configuration

HTTP request lifecycle

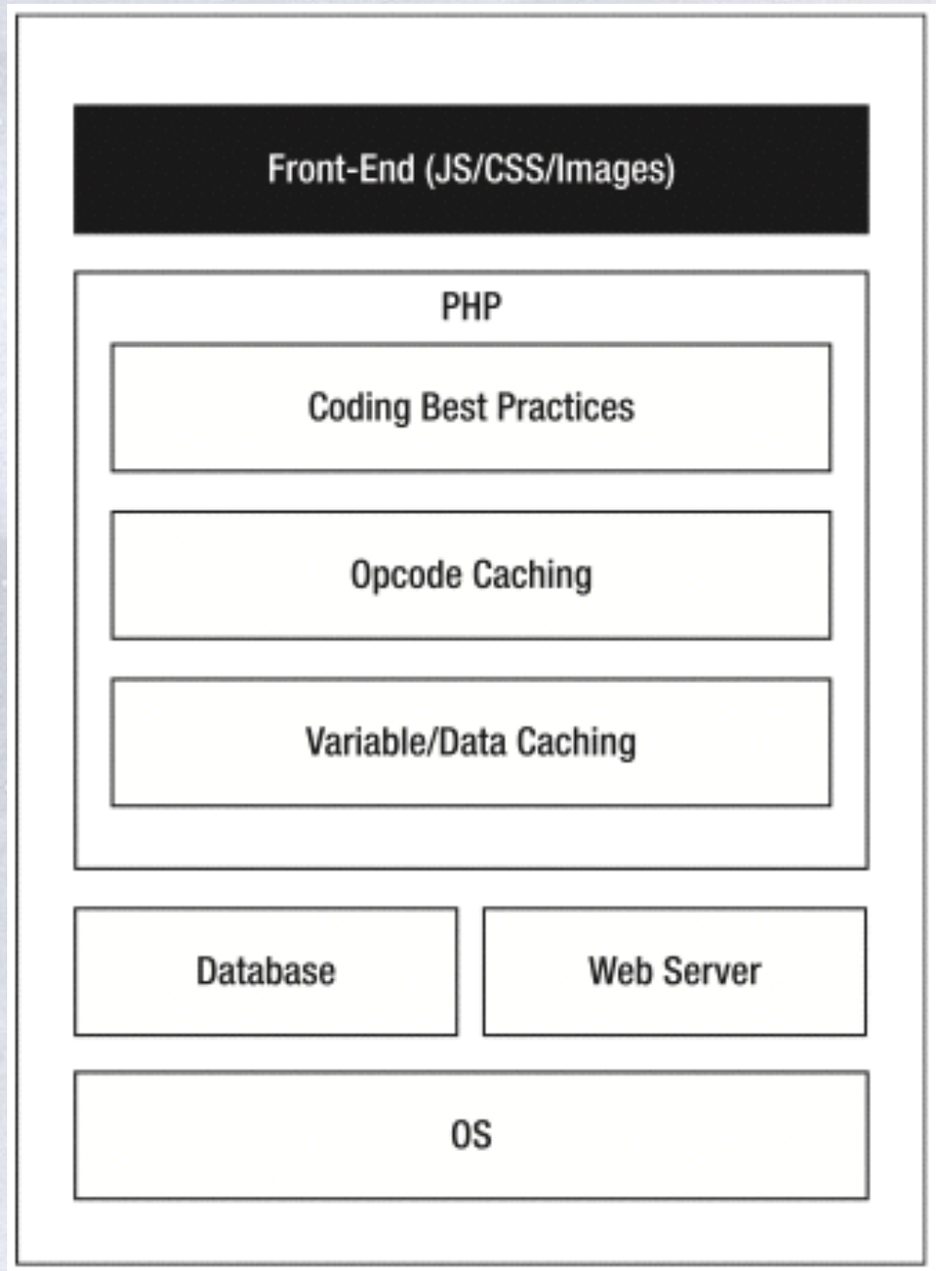


网站性能优化思路

网页通过HTTP获得,在浏览器解析,那么
网页性能优化思路:

Improving Client Download and Rendering Performance

- Firebug, YSlow, and Page Speed
  - An in-depth look at the response from a web server
  - Profile front-end logic within your JavaScript
  - An itemized list of resources that the browser will fetch
  - The length of time the browser took to fetch and receive the resource
  - Suggestions on how to optimize the response
- YUI Compressor, Closure Compiler, and Smush.it
  - optimize the response



YSlow

- YSlow v2's 22 web optimization rules cover the following:
  - CSS optimization
  - Image optimization
  - JavaScript optimization
  - Server optimization

CSS Optimization Rules

- Place the CSS styles at the top of the HTML document.
- Avoid certain CSS expressions.
- Minify the CSS files.

Image Optimization Rules

- Use desired image sizes instead of resizing within HTML using width and height attributes.
- Create sprites when possible.

JavaScript Optimization

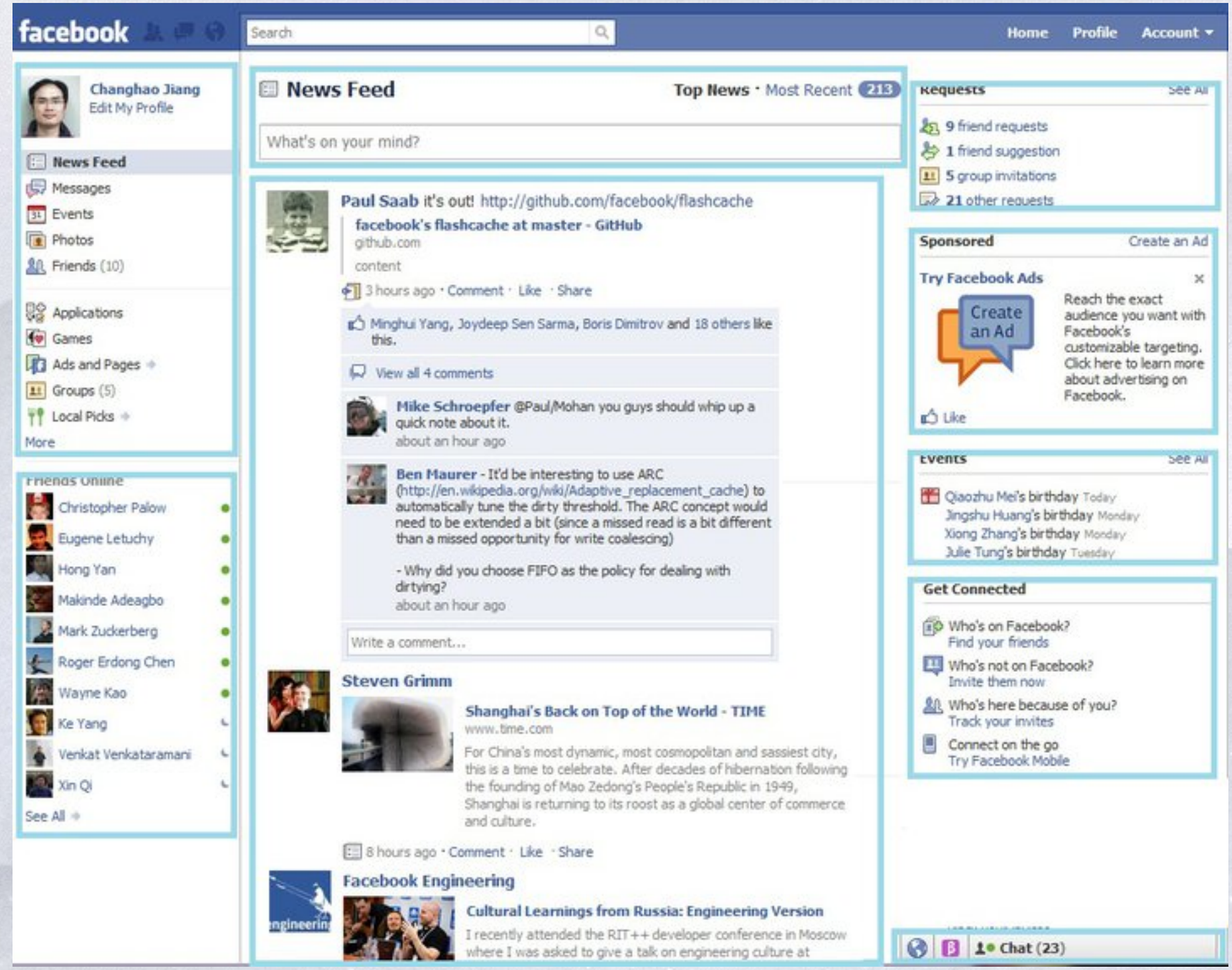
- Place JavaScript at the bottom of the HTML.
- Minify JavaScript.
- Make JavaScript external.

Server Optimization

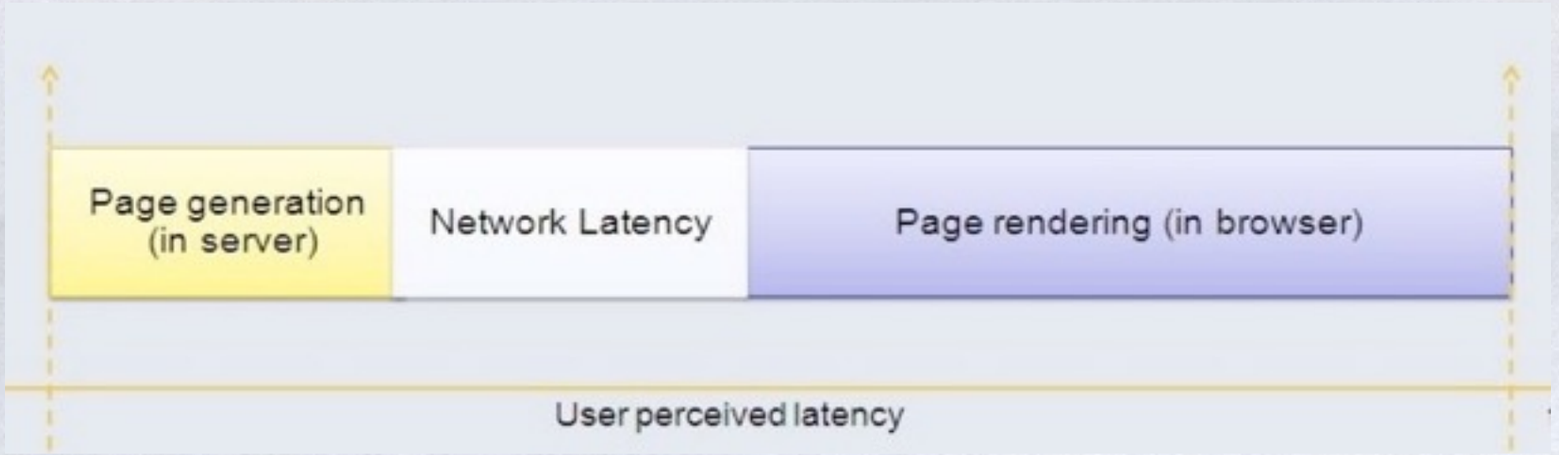
- Whether the server utilizes Gzip/bzip2 compression
- Whether DNS lookups are reduced
- Whether ETags are implemented

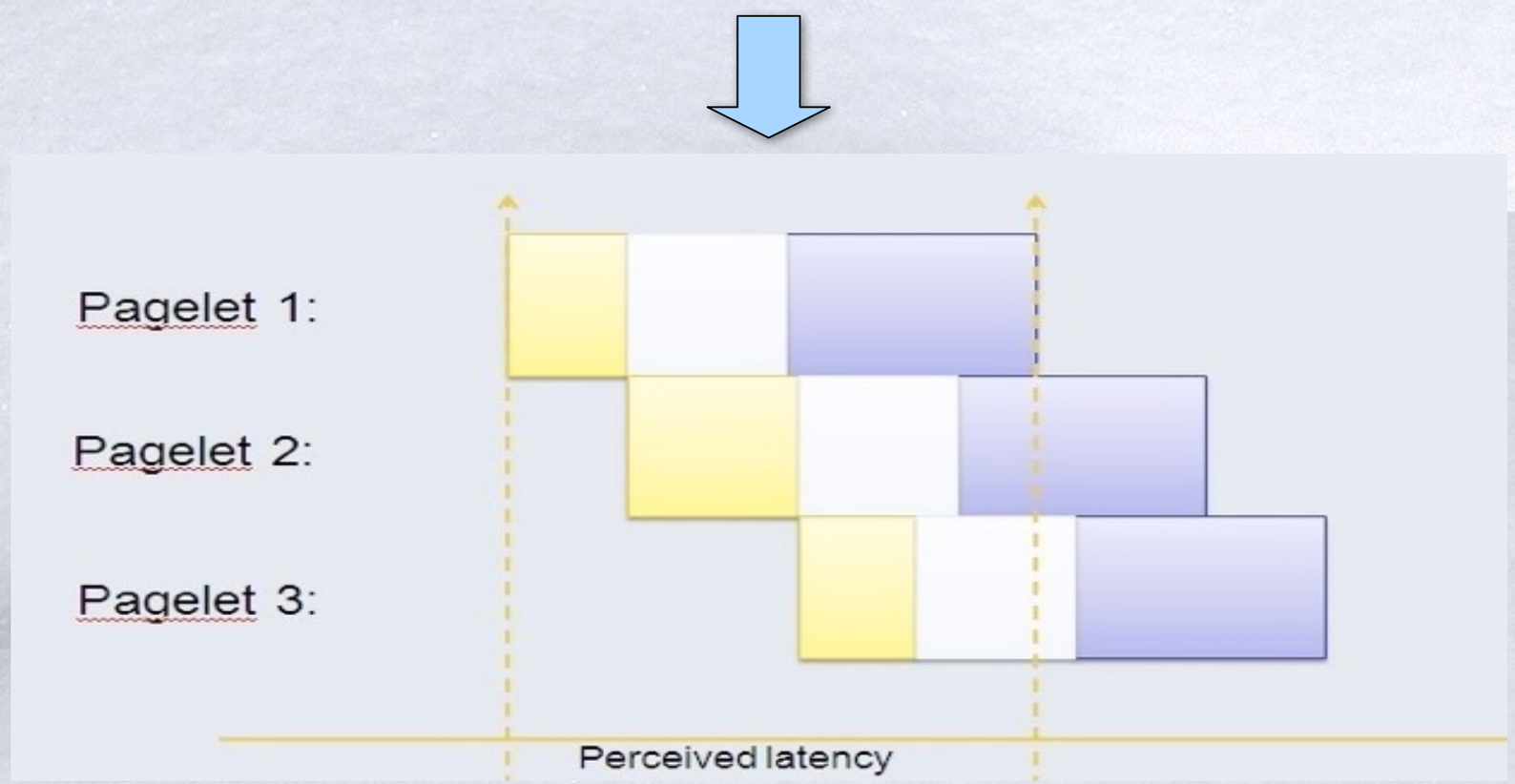
BigPipe

- 重新设计的Web服务处理过程
- 利用流水线思想降低网页的用户感知延迟
- AJAX模块化方式的性能加强版

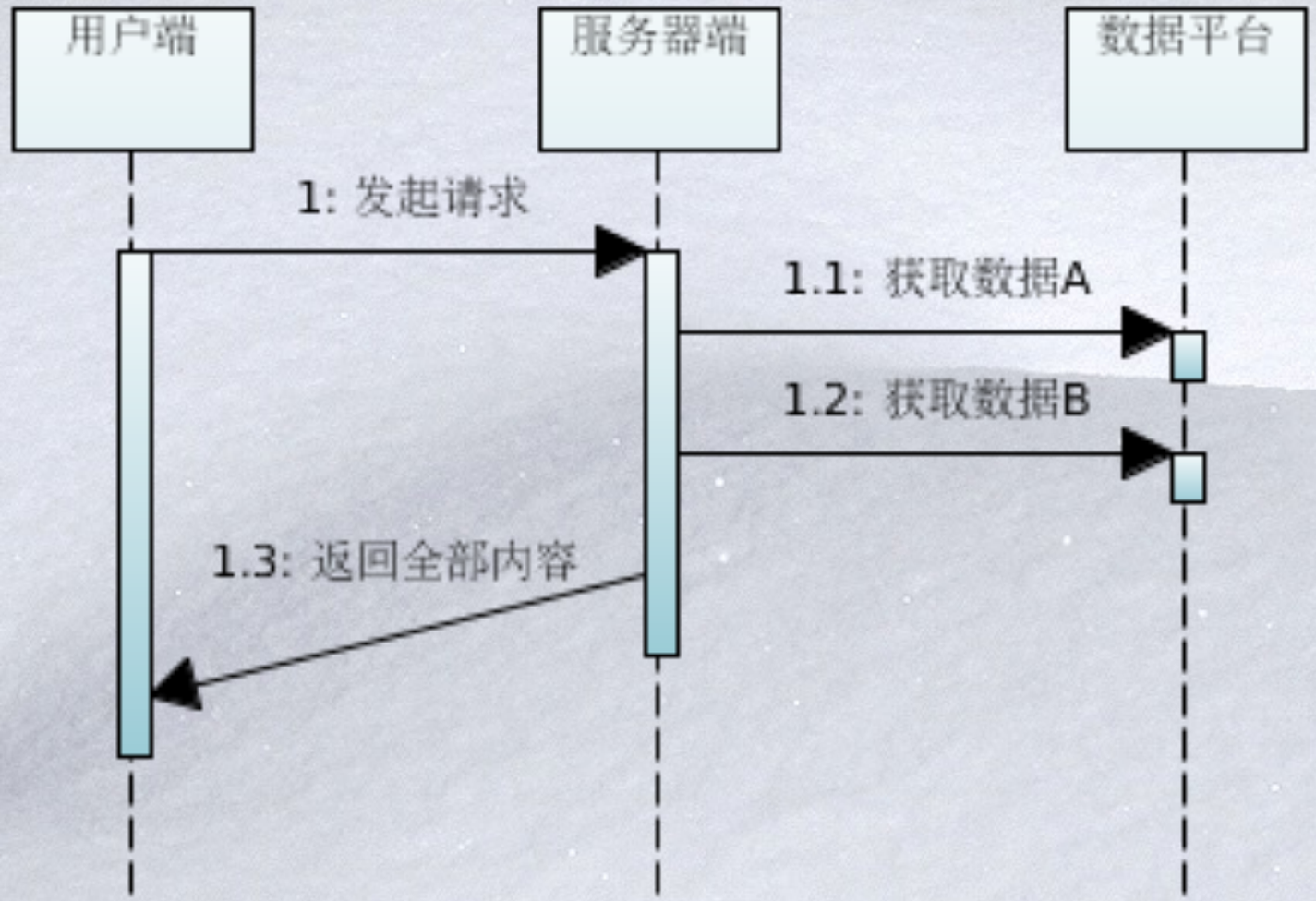

facebook
Search
Home
Profile
Account
Changhao Jiang
Edit My Profile
News Feed
Messages
Events
Photos
Friends (10)
Applications
Games
Ads and Pages
Groups (5)
Local Picks
More
Friends Online
Christopher Palow
Eugene Letuchy
Hong Yan
Makinde Adeagbo
Mark Zuckerberg
Roger Erdong Chen
Wayne Kao
Ke Yang
Venkat Venkataramani
Xin Qi
See All
News Feed
Top News · Most Recent 213
What's on your mind?
Paul Saab it's out! http://github.com/facebook/flashcache
facebook's flashcache at master - GitHub
github.com
content
3 hours ago · Comment · Like · Share
Minghui Yang, Joydeep Sen Sarma, Boris Dimitrov and 18 others like this.
View all 4 comments
Mike Schroepfer @Paul/Mohan you guys should whip up a quick note about it.
about an hour ago
Ben Maurer - It'd be interesting to use ARC (http://en.wikipedia.org/wiki/Adaptive_replacement_cache) to automatically tune the dirty threshold. The ARC concept would need to be extended a bit (since a missed read is a bit different than a missed opportunity for write coalescing)
- Why did you choose FIFO as the policy for dealing with dirtying?
about an hour ago
Write a comment...
Steven Grimm
Shanghai's Back on Top of the World - TIME
www.time.com
For China's most dynamic, most cosmopolitan and sassiest city, this is a time to celebrate. After decades of hibernation following the founding of Mao Zedong's People's Republic in 1949, Shanghai is returning to its roost as a global center of commerce and culture.
8 hours ago · Comment · Like · Share
Facebook Engineering
Cultural Learnings from Russia: Engineering Version
I recently attended the RIT++ developer conference in Moscow where I was asked to give a talk on engineering culture at
Requests
See All
9 friend requests
1 friend suggestion
5 group invitations
21 other requests
Sponsored
Create an Ad
Try Facebook Ads
Create an Ad
Reach the exact audience you want with Facebook's customizable targeting. Click here to learn more about advertising on Facebook.
Like
Events
See All
Qiaozhu Mei's birthday Today
Jingshu Huang's birthday Monday
Xiong Zhang's birthday Monday
Julie Tung's birthday Tuesday
Get Connected
Who's on Facebook?
Find your friends
Who's not on Facebook?
Invite them now
Who's here because of you?
Track your invites
Connect on the go
Try Facebook Mobile
Chat (23)

概念

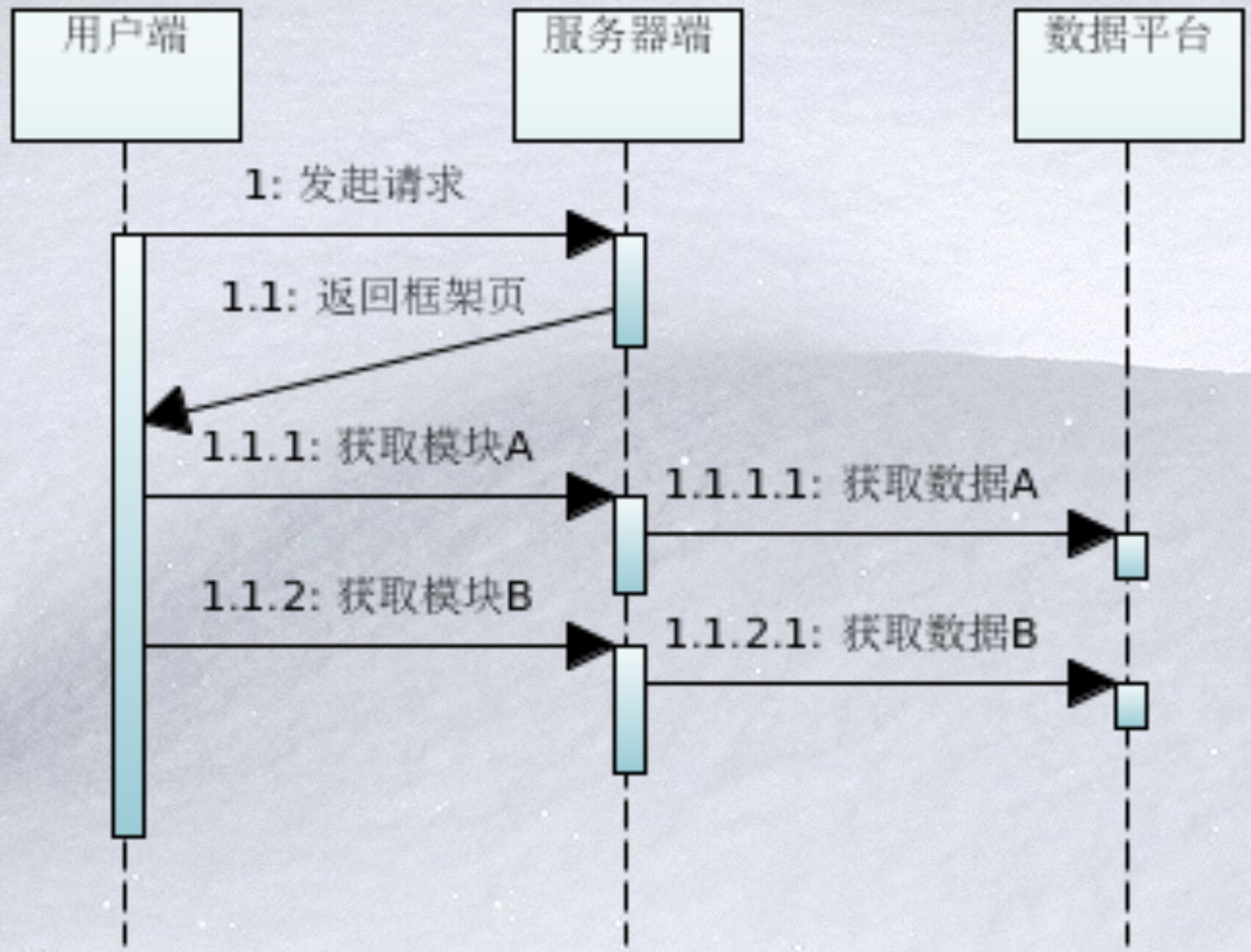

Page generation (in server)
Network Latency
Page rendering (in browser)
User perceived latency


Pagelet 1:
Pagelet 2:
Pagelet 3:
Perceived latency

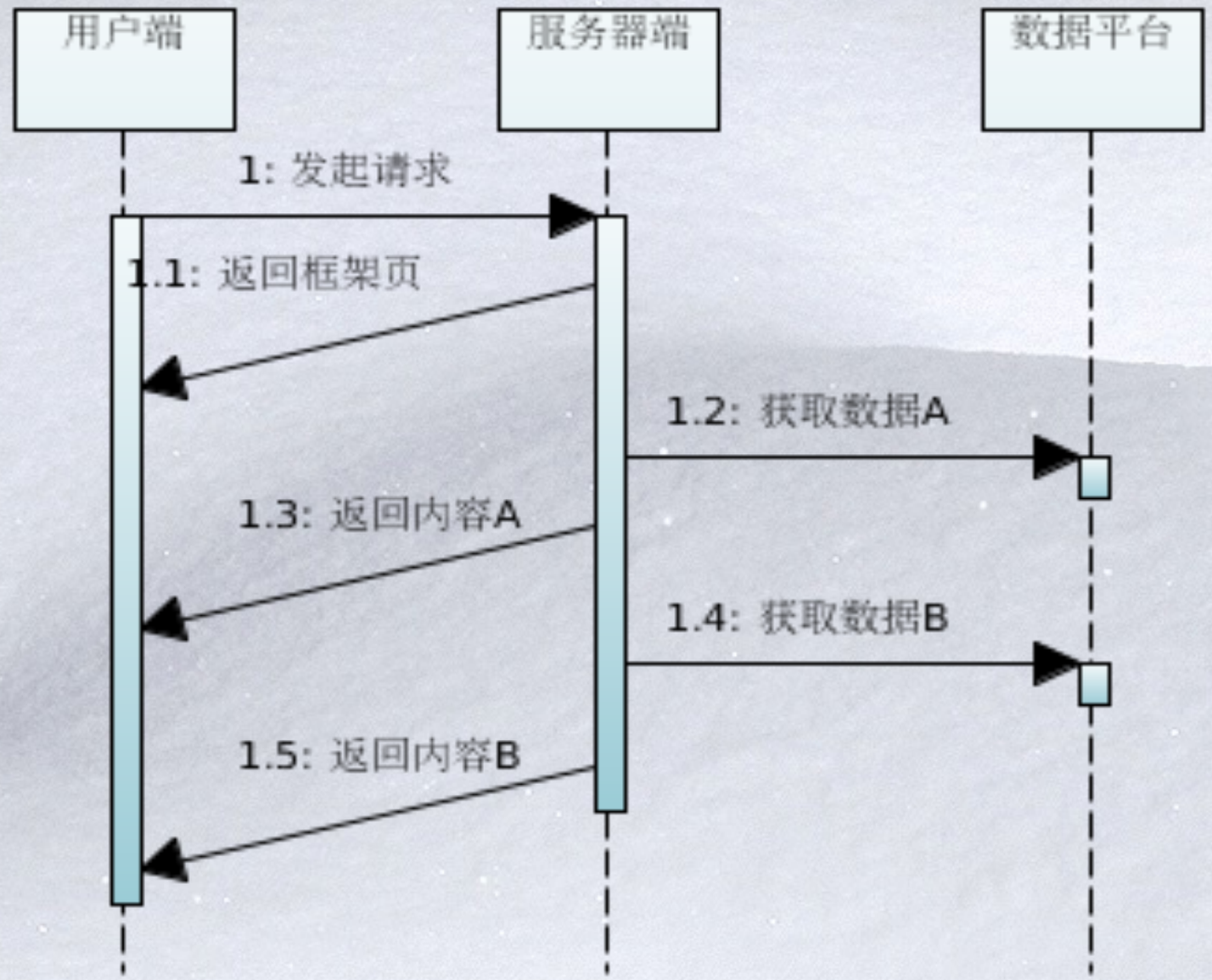
传统页面处理过程



AJAX 模块化处理过程



BigPipe处理过程



输出示例 - 普通模式

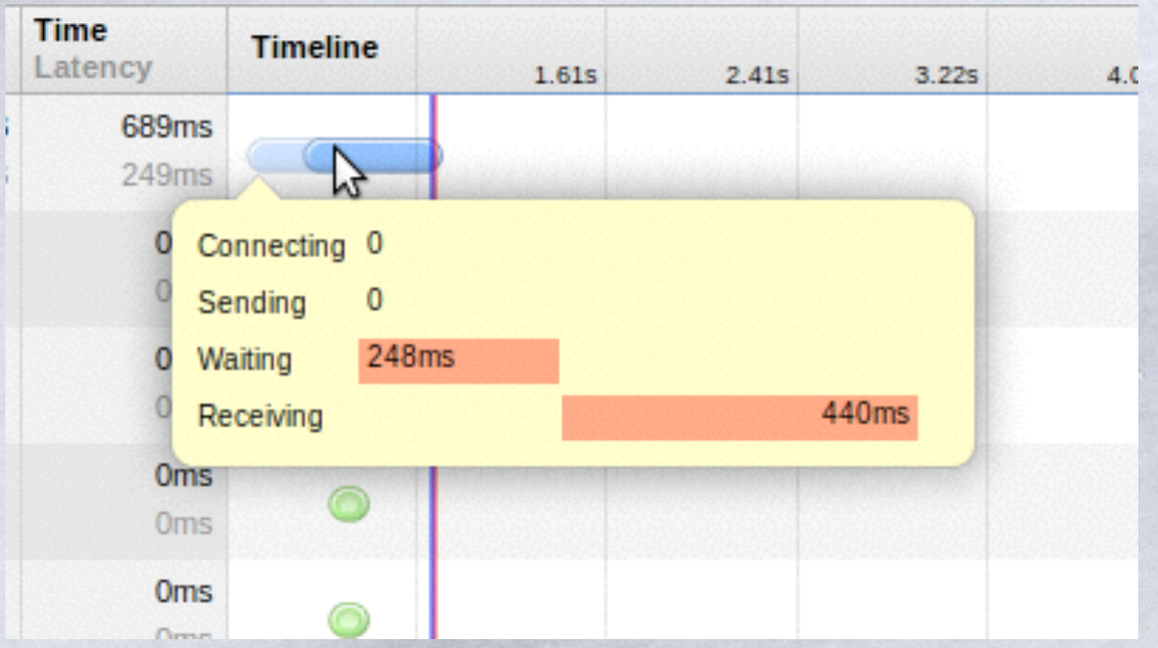
```
<html>
<div id="pl_left">左侧内容</div>
<div id="pl_main">主要内容</div>
</html>
```

输出示例 - BigPipe模式

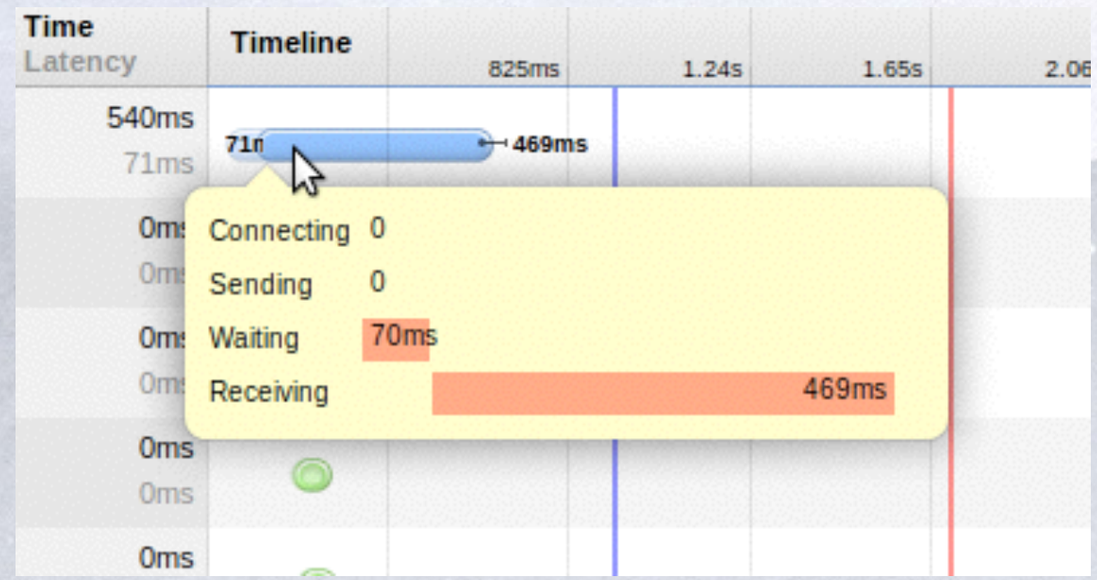
```
<html>
<script src="BigPipe.js"></script>
<div id="pl_left"></div>
<div id="pl_main"></div>
<script>bp.pagelet({'id': 'pl_left' , 'html' : '左侧内容'})</script>
<script>bp.pagelet({'id': 'pl_main' , 'html' : '主要内容'})</script>
</html>
```

比较

传统模式(BP关闭)



流水线模式(BP开启)



248ms => 70 ms

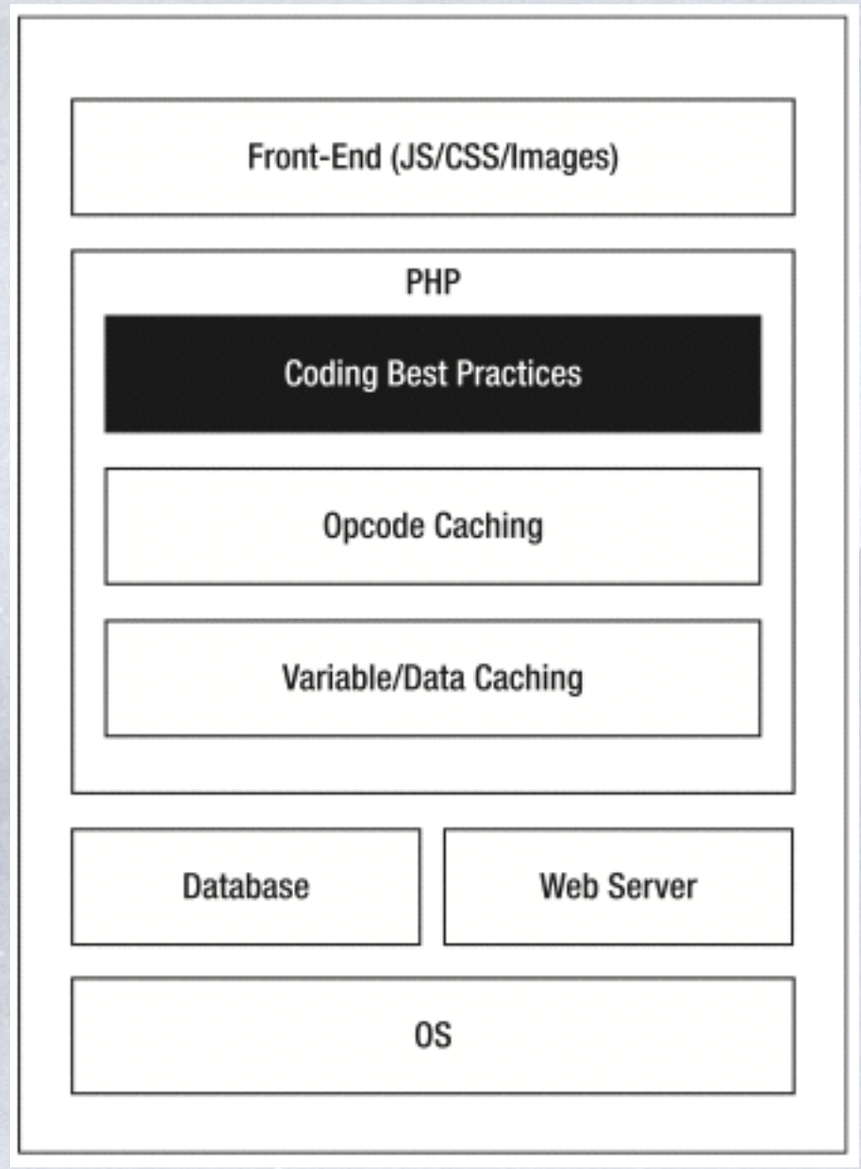
好处

- ✓ 用户更快看到部分内容
- ✓ 减少HTTP请求
- ✓ Pagelet处理可并发

坏处

- 强制页面模块化
- 部分运算移动到浏览器端
- 浏览器兼容性
- 访问者是爬虫或者访问者浏览器禁止使用JS
- SEO（user-agent）

PHP Code Optimization



Profiling PHP Code

- APD (Pure profiler)
  - http://pecl.php.net/apd
- DBG (Profiler & Debugger)
  - http://dd.cron.ru/dbg/

OOP Tweaks

- Always declare your statics! Dynamic methods accessed statically are 50%+ slower.

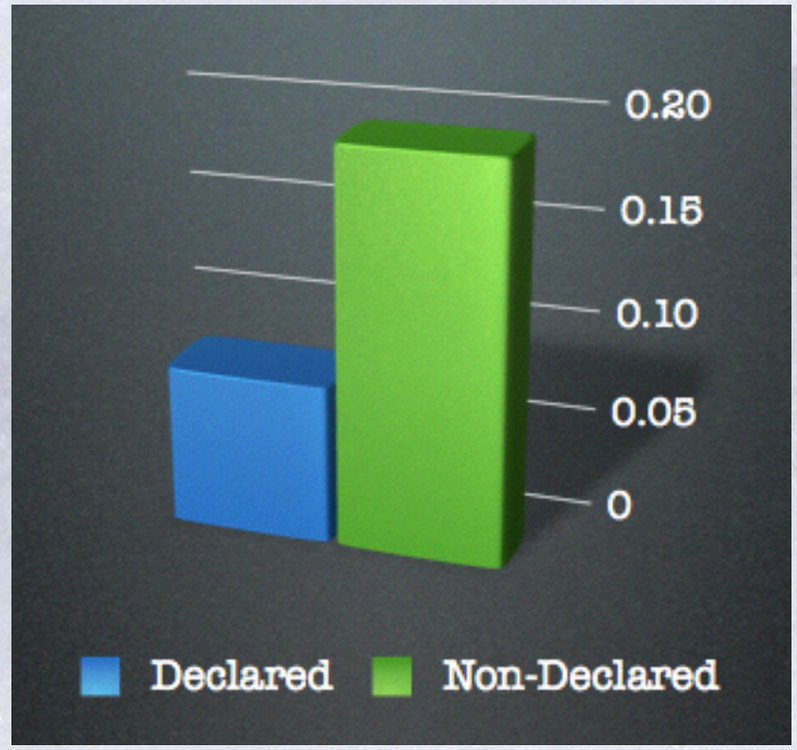
```
protected static function loadExternalFile( $file )
    {
        if ( file_exists( $file ) )
        {
            require( $file );
        } else {
            throw new ezcBaseFileNotFoundException( $file );
        }
    }
```

Quick Static Benchmark

```php
<?php
class bench {
    public function a() { return 1; }
    public static function b() { return 1; }
}
$s = microtime(1);
for ($i = 0; $i < 100000; $i++) bench::a();
$e = microtime(1);
echo "Dynamic Static Method: ".($e - $s)."\n";
$s = microtime(1);
for ($i = 0; $i < 100000; $i++) bench::b();
$e = microtime(1);
echo "Declared Static Method: ".($e - $s)."\n";
```

Conclusion?

- Declaring static methods as was done gives us a 60% speed boost.



Use Class Constants!

```
class ezcInputForm
{
    const VALID = 0;
    const INVALID = 1;
    const DEF_NO_ARRAY =1;
    const DEF_EMPTY =2;
    const DEF_ELEMENT_NO_DEFINITION_ELEMENT = 3;
    const DEF_NOT_REQUIRED_OR_OPTIONAL =5;
    const DEF_WRONG_FLAGS_TYPE =6;
    const DEF_UNSUPPORTED_FILTER =7;
    const DEF_FIELD_NAME_BROKEN =8;
}
```

Advantages

- Parsed at compile time, no execution overhead.
- Faster lookups due to a smaller hash.
- “Namespacing” & shorter hash names.
- Cleaner code speeds up debugging.

require vs. require_once

```php
<?php
require_once("ClassA.php");
require_once("ClassB.php");
require_once("ClassC.php");
require_once("ClassD.php");
echo "Only testing require_once.";
```

VS.

```php
<?php
require("ClassA.php");
require("ClassB.php");
require("ClassC.php");
require("ClassD.php");
echo "Only testing require.";
```

```php
<?php
class A{
}
class B {
}
class C {
}
class D {
}
```

Calculating Loop Length in Advance

Un-optimized for Loop

```
<?php
$items = array(1,2,3,4,5,6,7,8,9,10);
for($i=0; $i<count($items); $i++)
{
    $x = 10031981 * $i;
    }
```

Optimized for Loop

```
<?php
$items = array(1,2,3,4,5,6,7,8,9,10);
$count = count($items);
for($i=0; $i<$count; $i++) {
$x = 10031981 * $i;
}
```

Accessing Array Elements Using foreach vs. for vs. while

- using a foreach loop instead of either a while or for loop

File Access

- Using fread() for small file data access
- file_get_contents for large files

Faster Access to Object Properties

```
<?php
class Person
{
    private $_gender = NULL;
    private $_age = NULL;
    public function getGender() {
        return $this->_gender;
     }
    public function getAge() {
        return $this->_age;
        }
    public function setGender($gender) {
        $this->_gender = $gender;
    }
    public function setAge($age) {
        $this->_age = $age; }
}

$personObj = new Person();
$start = microtime();
for($i=0; $i<100000; $i++) {
    $personObj->getGender();
}
echo microtime()-$start.'ms';
```

```
<?php
class Person
{
    public $_gender = NULL;
    public $_age = NULL;
}

$personObj = new Person();
$start = microtime();
for($i=0; $i<100000; $i++) {
    //Average: 0.0205617ms
    $personObj->_gender;
    }

echo microtime()-$start.'ms';
```

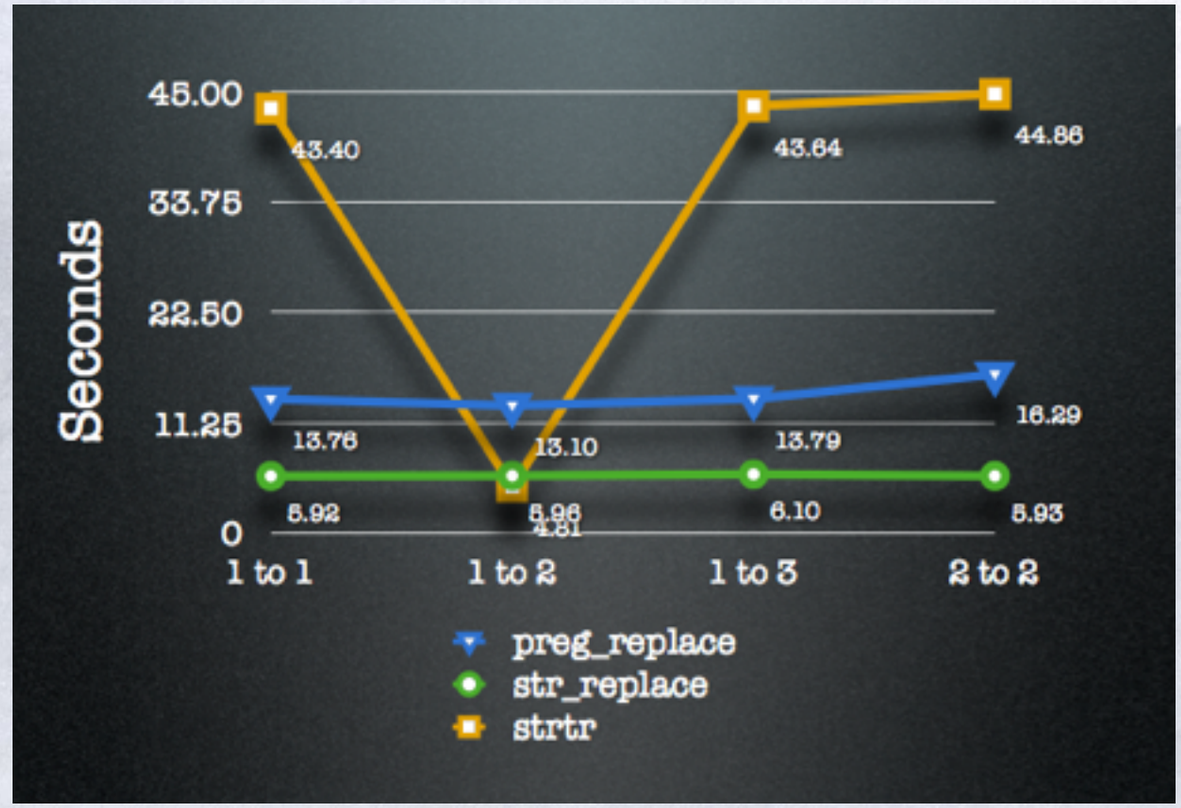
PCRE is slow

```
<?php
preg_replace( "/\n/", "\\n", $text);
str_replace( "/\n/", "\\n", $text);

preg_match(“/^foo_/i", "FoO_")
!strncasecmp("foo_", "FoO_", 4)

preg_match(“/[abce]/", “string")
strpbrk(“string", “abcd")

preg_match("!string!i", "text")
stripos("text", "string")
?>
```



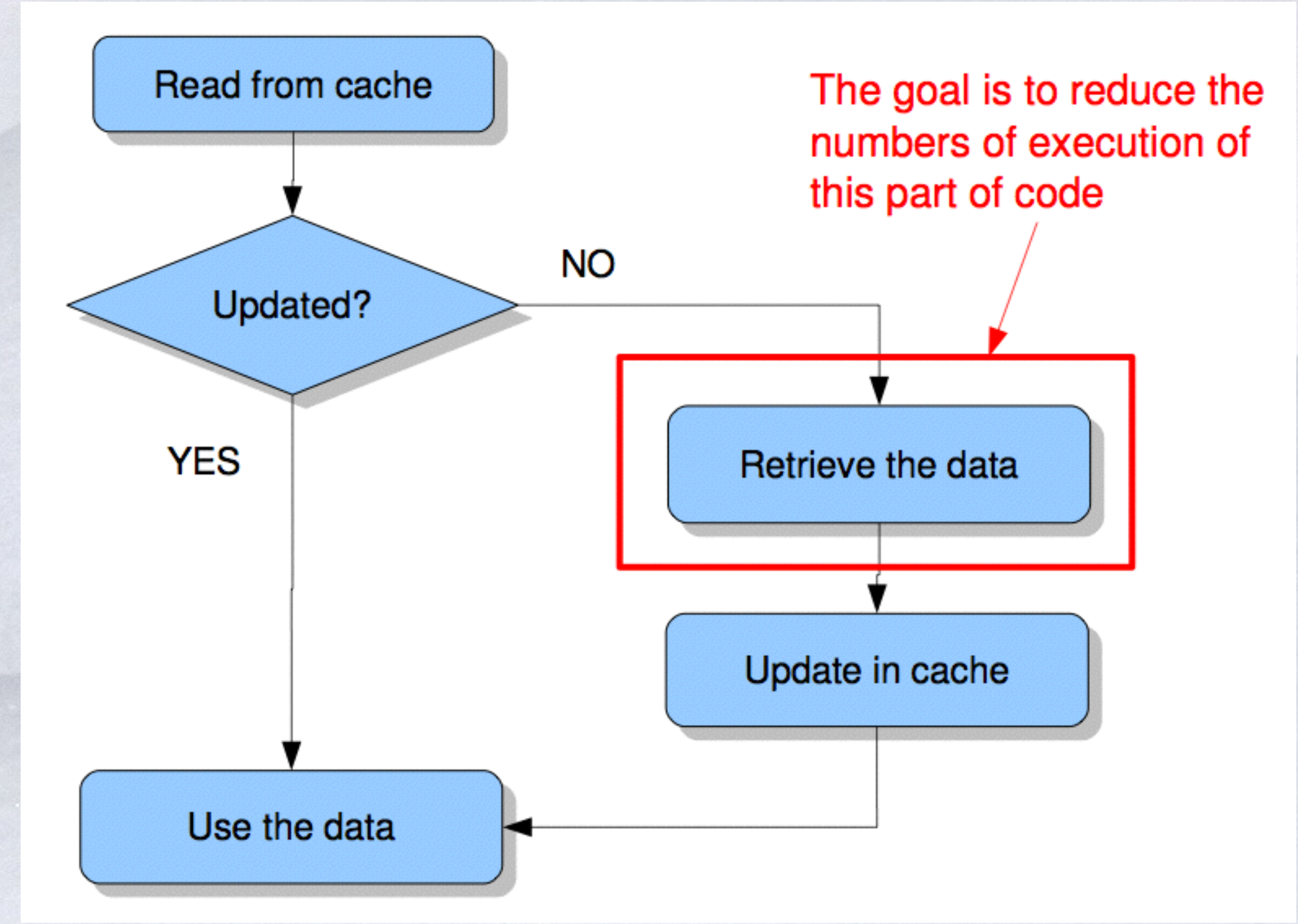
Looking Under the Hood

- VLD
  - Reviewing Opcode Functions
- strace
  - C-level Tracing
- Xdebug
  - Memory consumption of your PHP script
  - Total number of calls made to a function
  - Total time spent within the function
  - Complete stack trace for a function

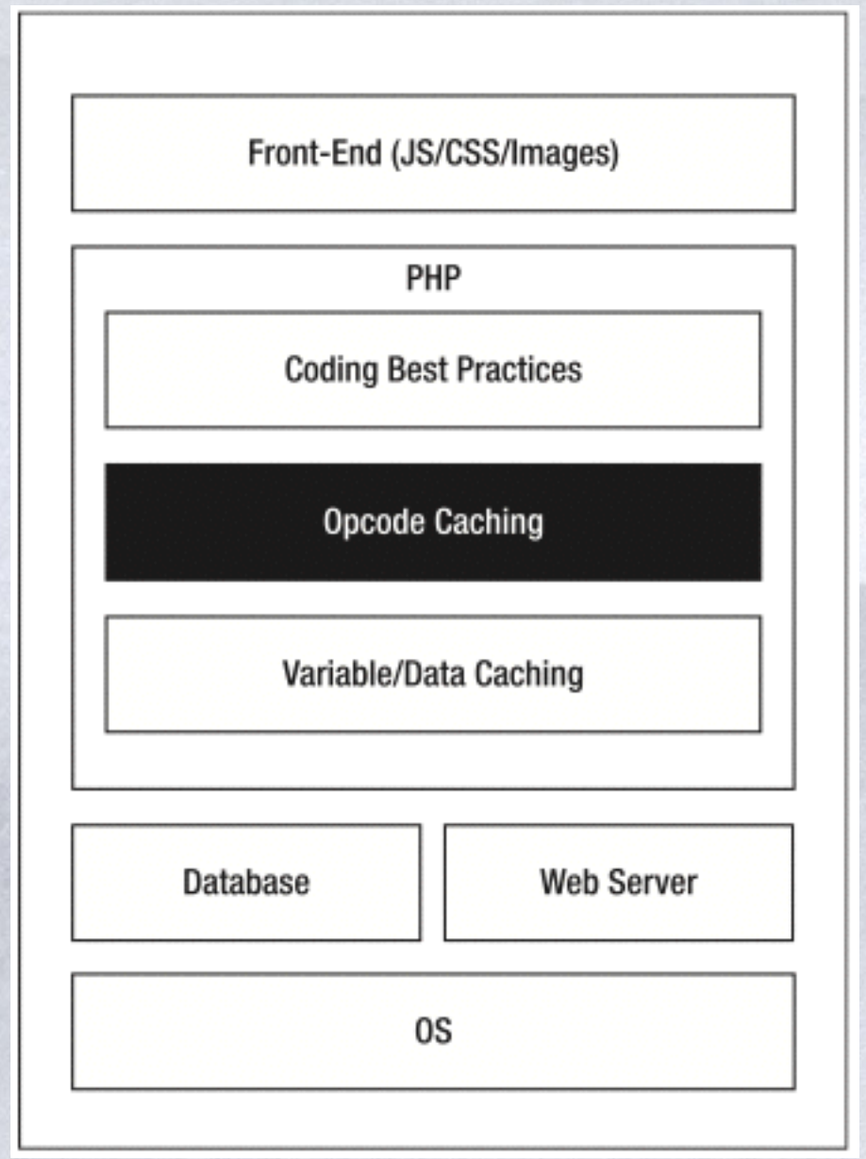
Caching in PHP

- In PHP we can use different kinds of cache systems:
  - File, by hand using the fwrite, fread, readfile, etc
  - Pear::Cache_Lite, using the files as cache storage
  - APC, using the apc extension (apc_add, apc_fetch, etc)
  - Xcache, from the lighttpd web server project
  - Memcached, using the memcached extension (Memcached::add, Memcached::get, etc)
  - Zend Server/CE, using Zend Server API (zend_shm_cache_fetch, zend_shm_cache_store, etc)

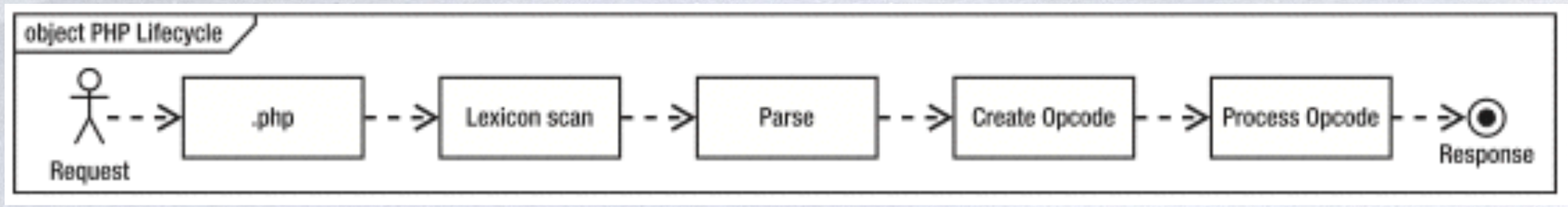
Conditional execution

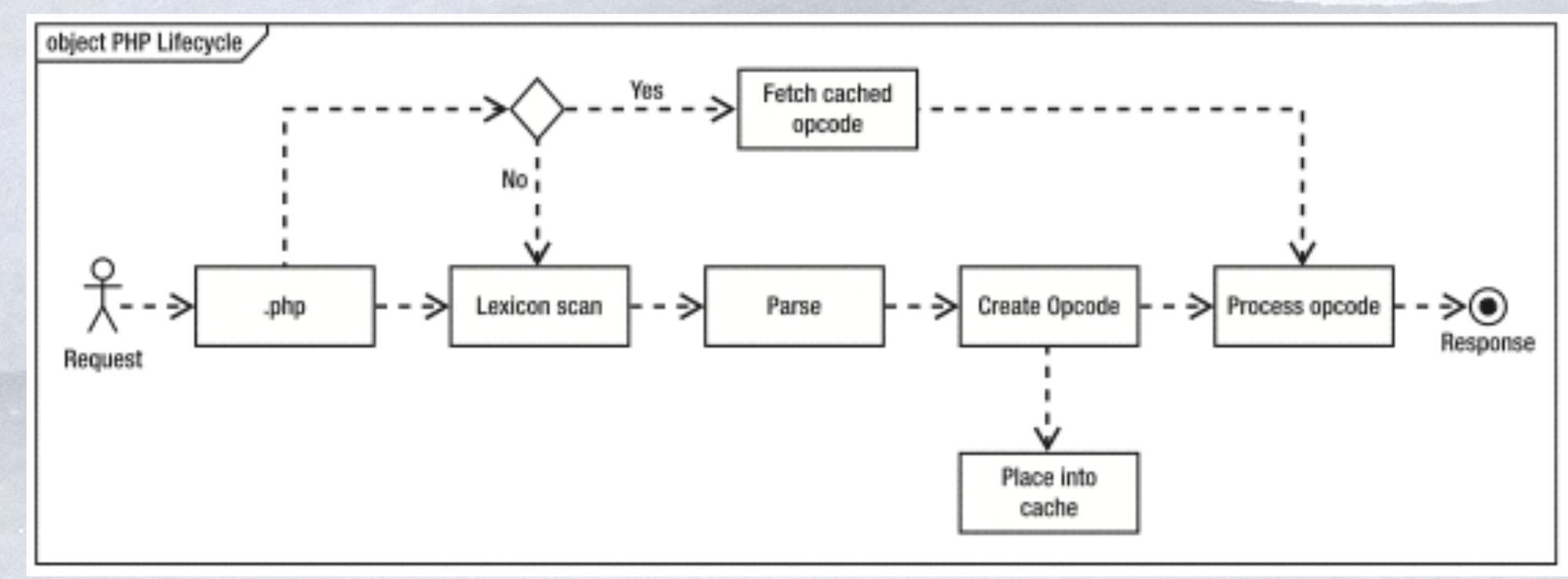


Opcode Caching



The PHP Life Cycle





Alternative PHP Cache(APC)

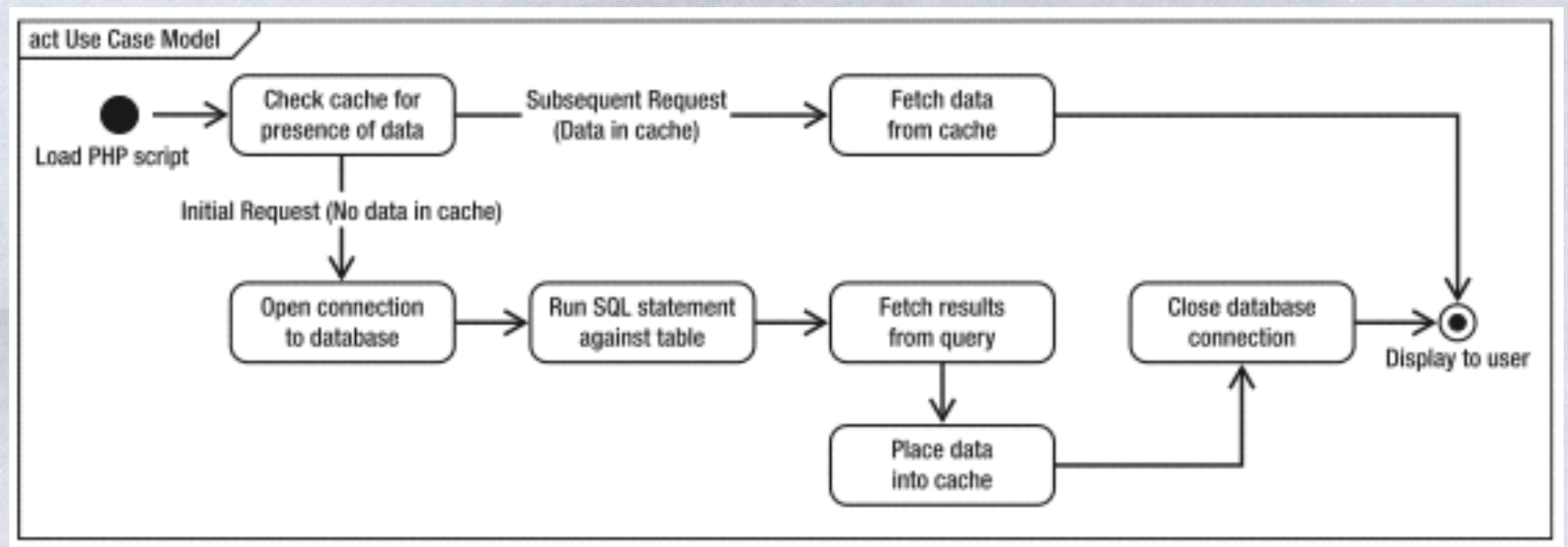
- A free, open source and robust framework for caching and optimizing PHP intermediate code.
- maintained by core PHP developers
- users
  - Yahoo!
  - Facebook
  - Wikipedia and many more

XCache

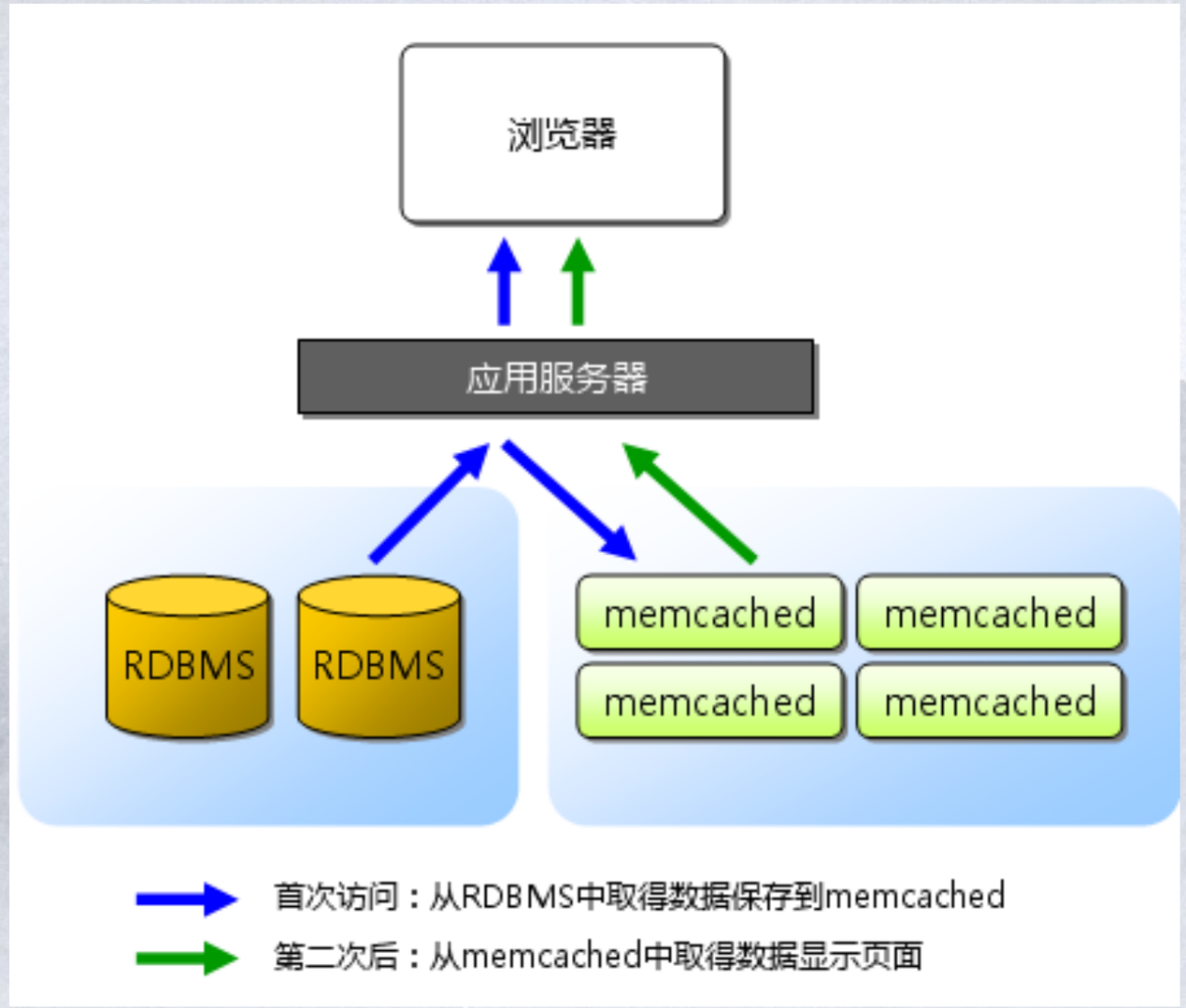
- another Opcode caching tool that is used by PHP. XCache, like APC, uses shared memory to store the Opcode and uses this cached Opcode to satisfy a request for a PHP script.
- Like APC, XCache is also available for both Unix-based systems and Windows.

Variable Caching

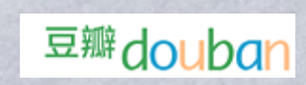
The Value of Implementing Variable Caching

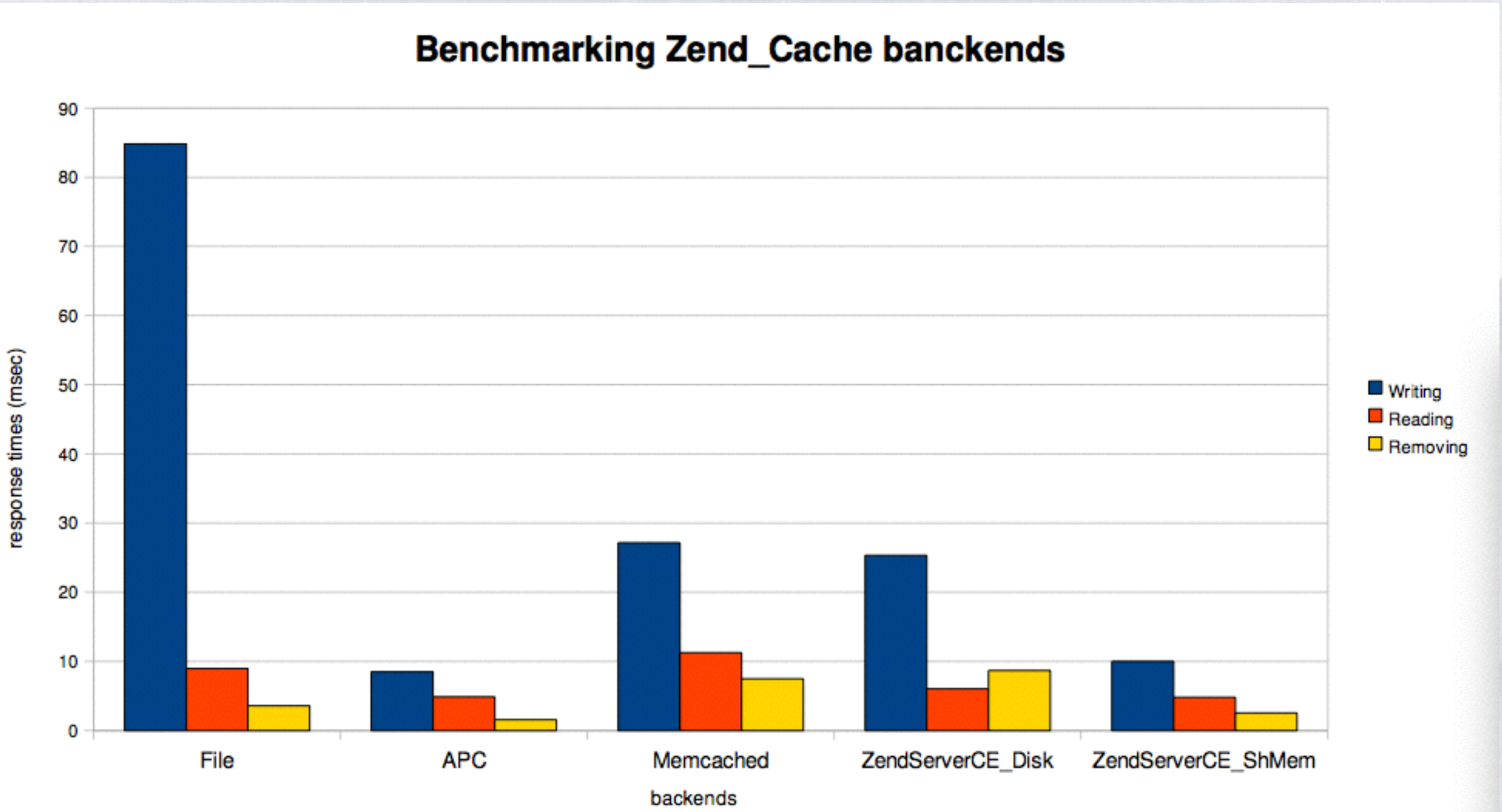


Memcached

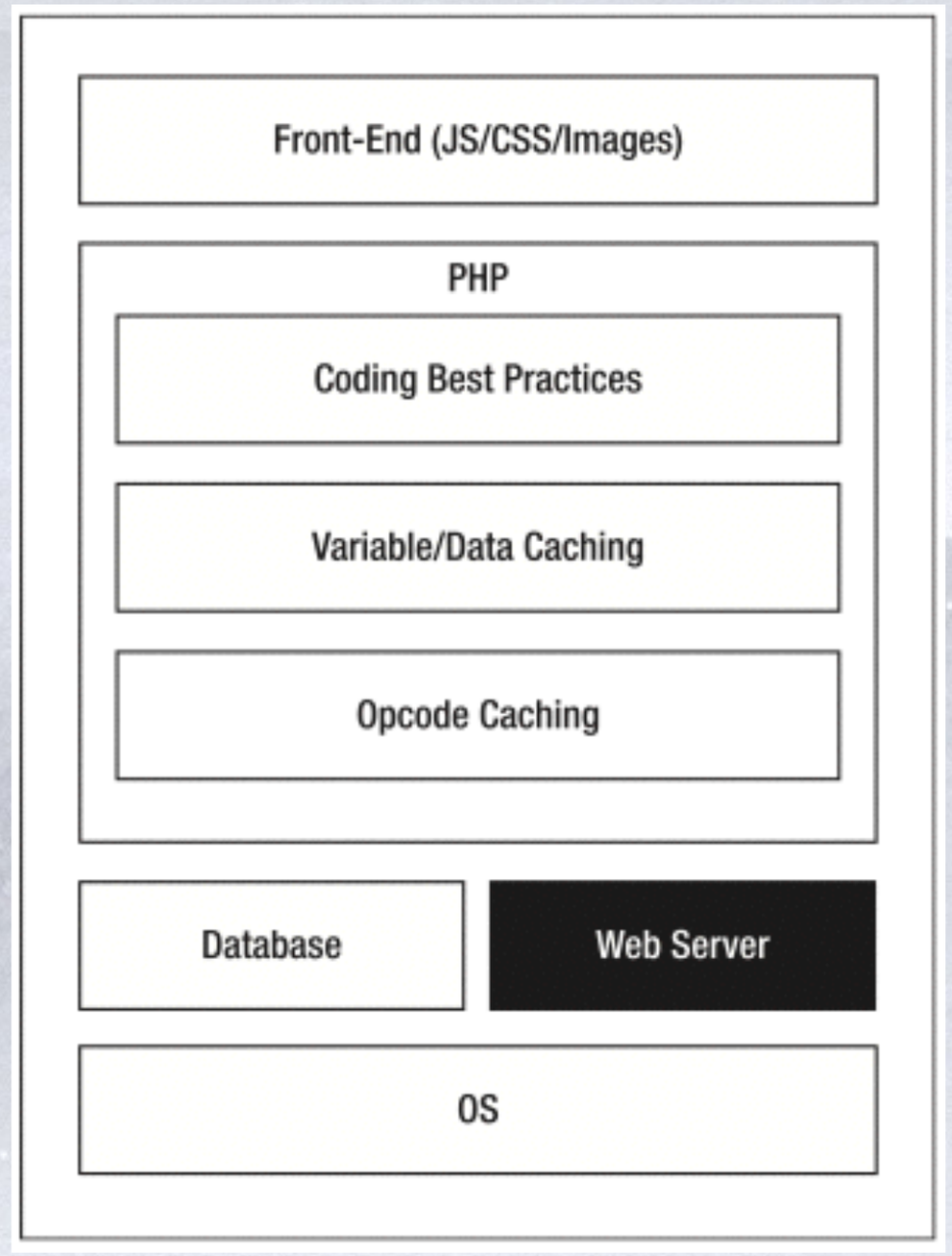

浏览器
应用服务器
RDBMS
RDBMS
memcached
memcached
memcached
memcached
首次访问：从RDBMS中取得数据保存到memcached
第二次后：从memcached中取得数据显示页面

Users of Memcached




Benchmarking Zend_Cache banckends
response times (msec)
90
80
70
60
50
40
30
20
10
0
File
APC
Memcached
ZendServerCE_Disk
ZendServerCE_ShMem
backends
Writing
Reading
Removing

Choosing the Right Web Server



Outline

- PHP Application Performance
- Web Application

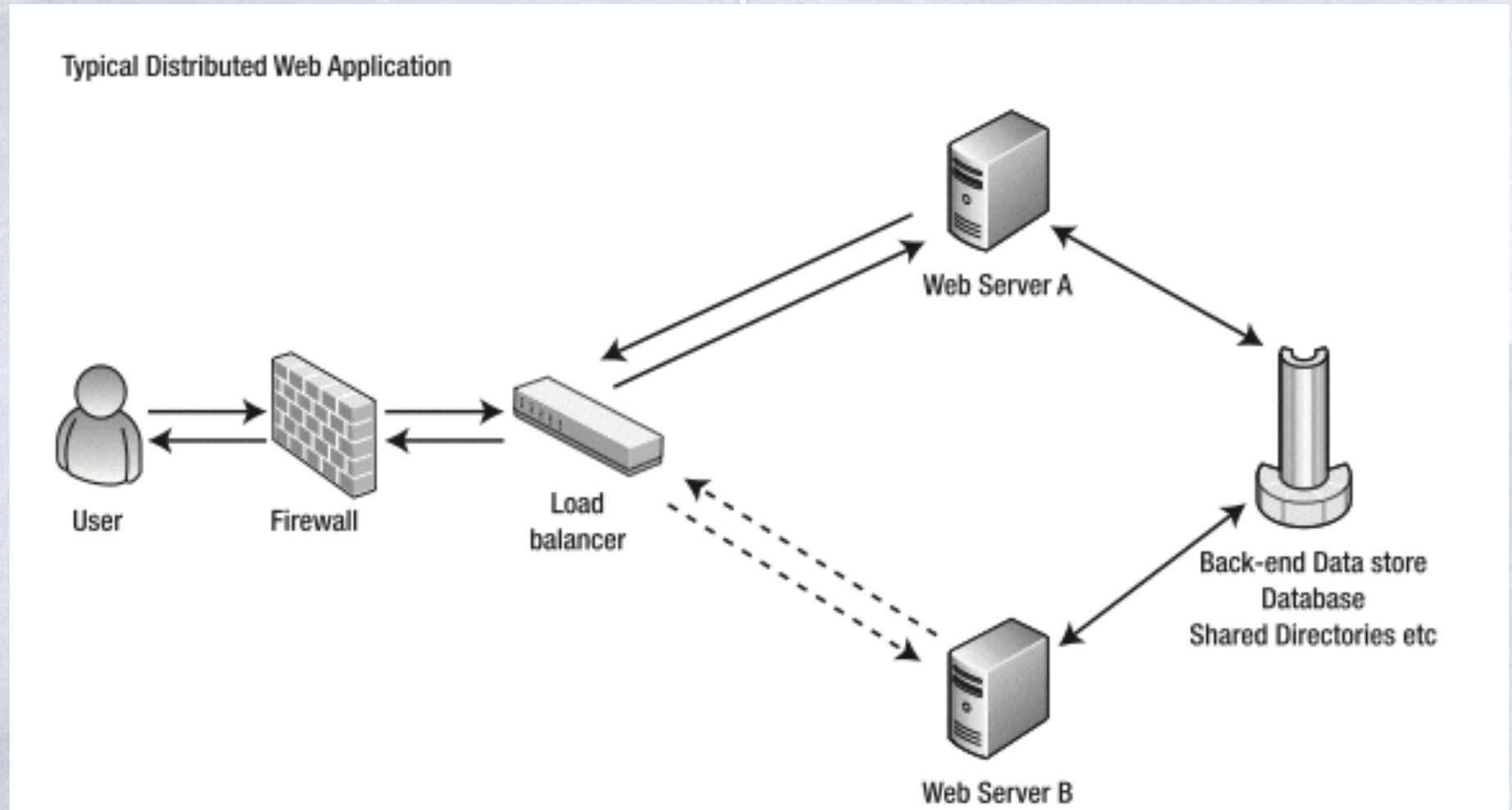
Static Content Serving

- While Apache is great for dynamic requests, static requests can be served WAY FASTER by other web servers.
  - lighttpd
  - Boa
  - Tux
  - thttpd
- For static requests these servers are easily 300-400% faster then Apache 1 or 2.

Scaling Beyond a Single Server

- Using Round-Robin DNS
- Using a Load Balancer
- Using Direct Server Return
- Sharing Assets with a Content Distribution Network

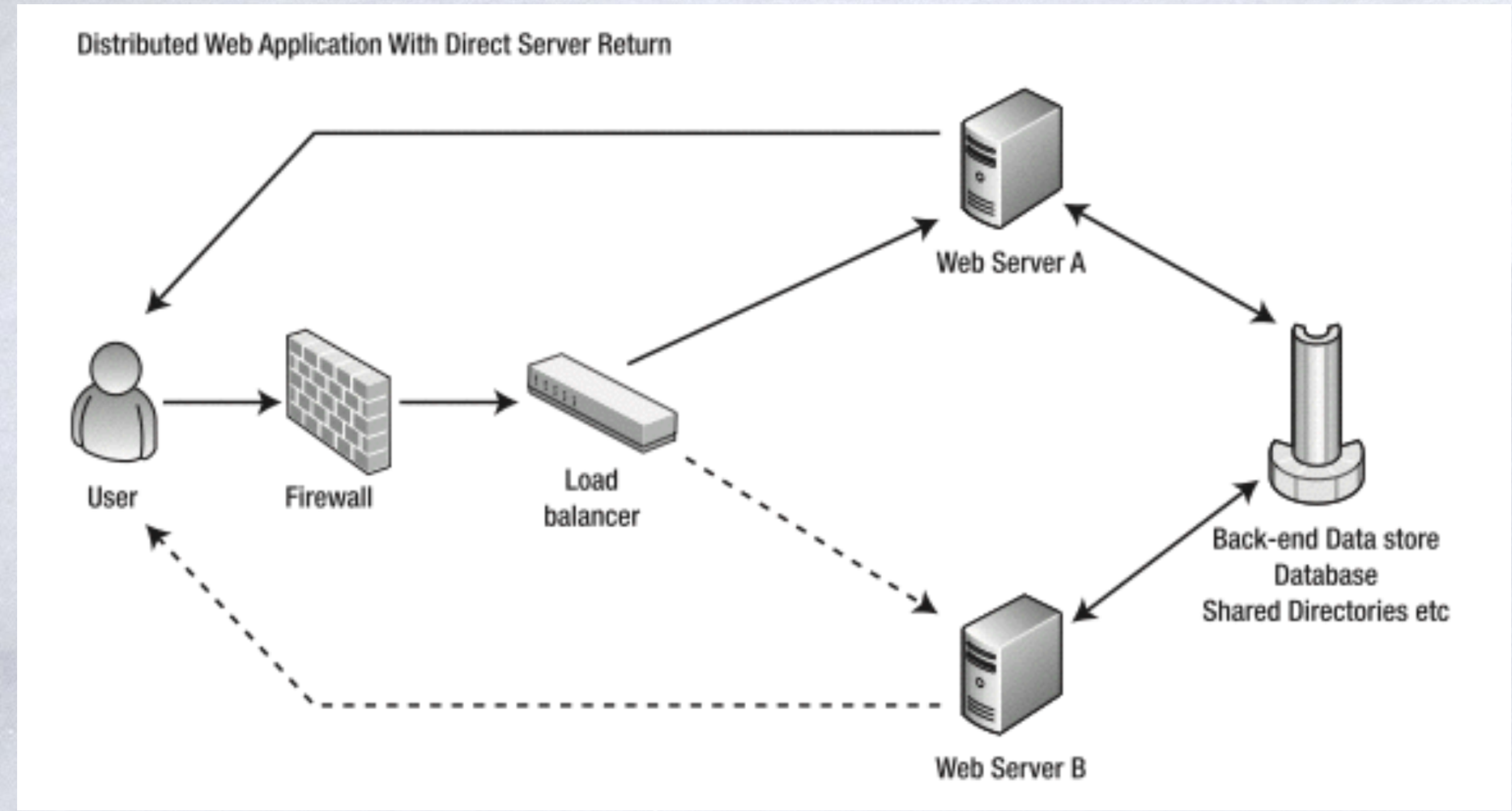
Using a Load Balancer



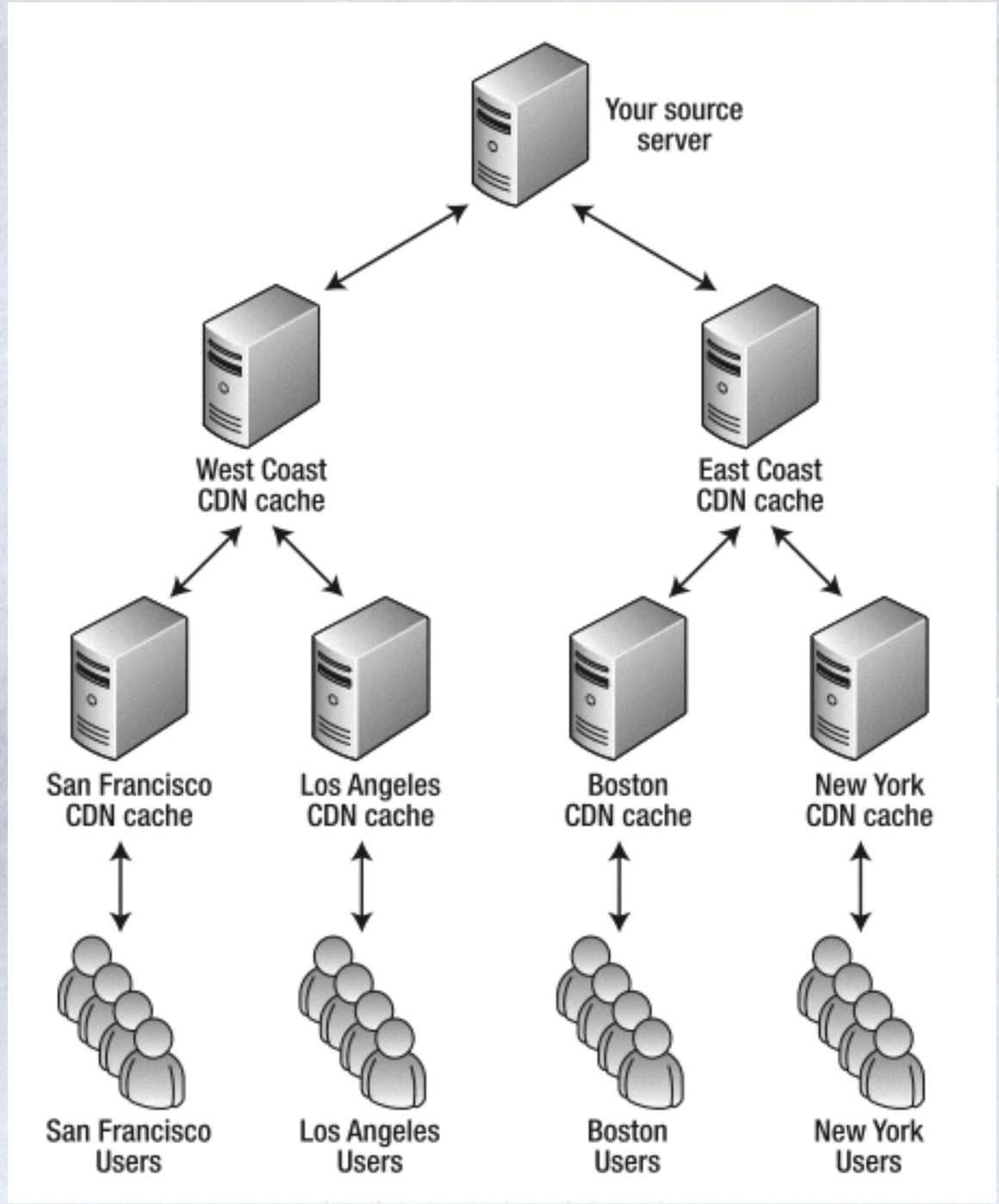
Categories of Load Balancers

- There are many kinds of load balancers available, both hardware- and software-based. Generally load balancers fall into four categories.
  - Totally software-based solutions
  - Software solutions using a separate load balancing server
  - Physical load balancing appliances
  - Load balancing services

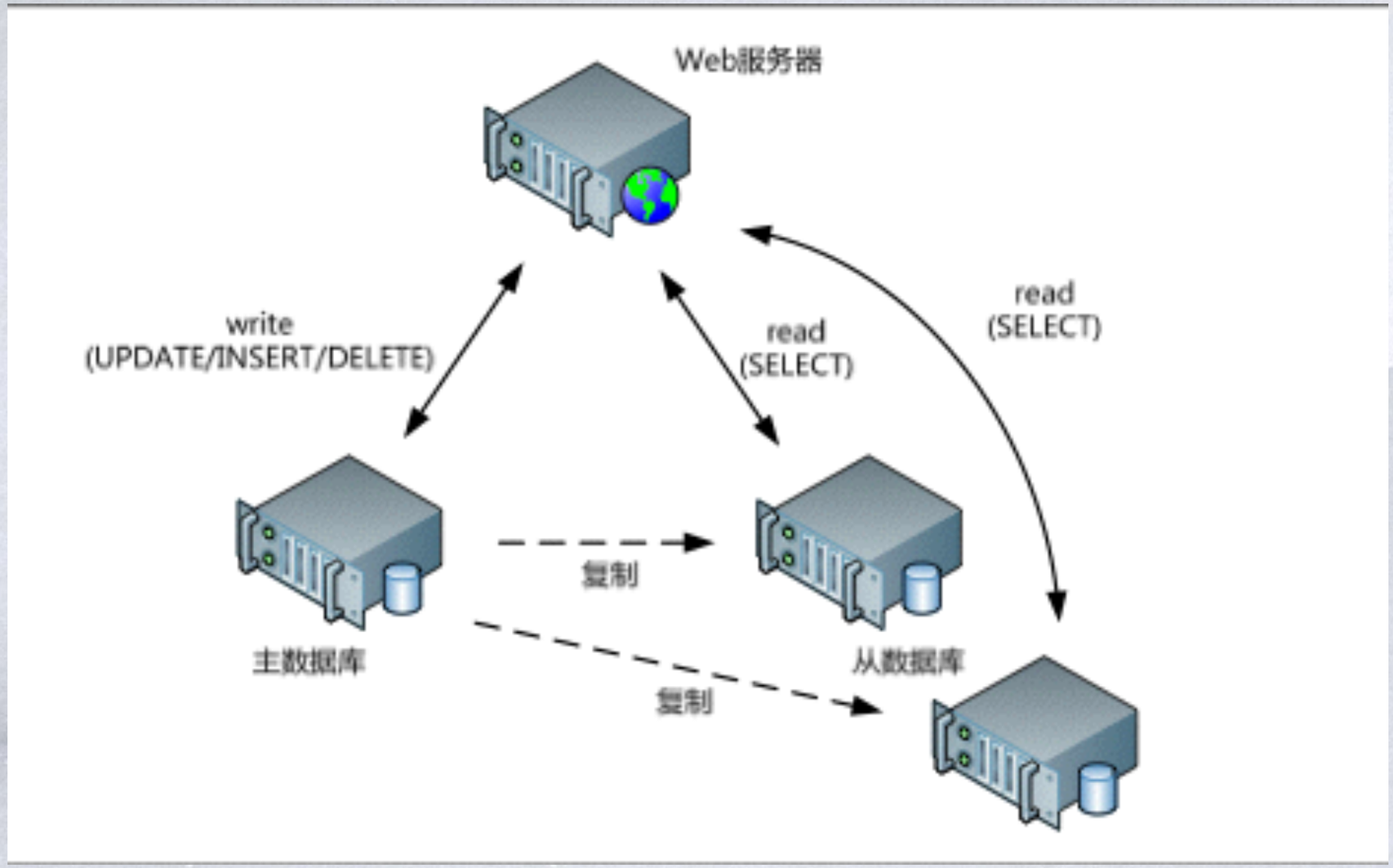
Using Direct Server Return

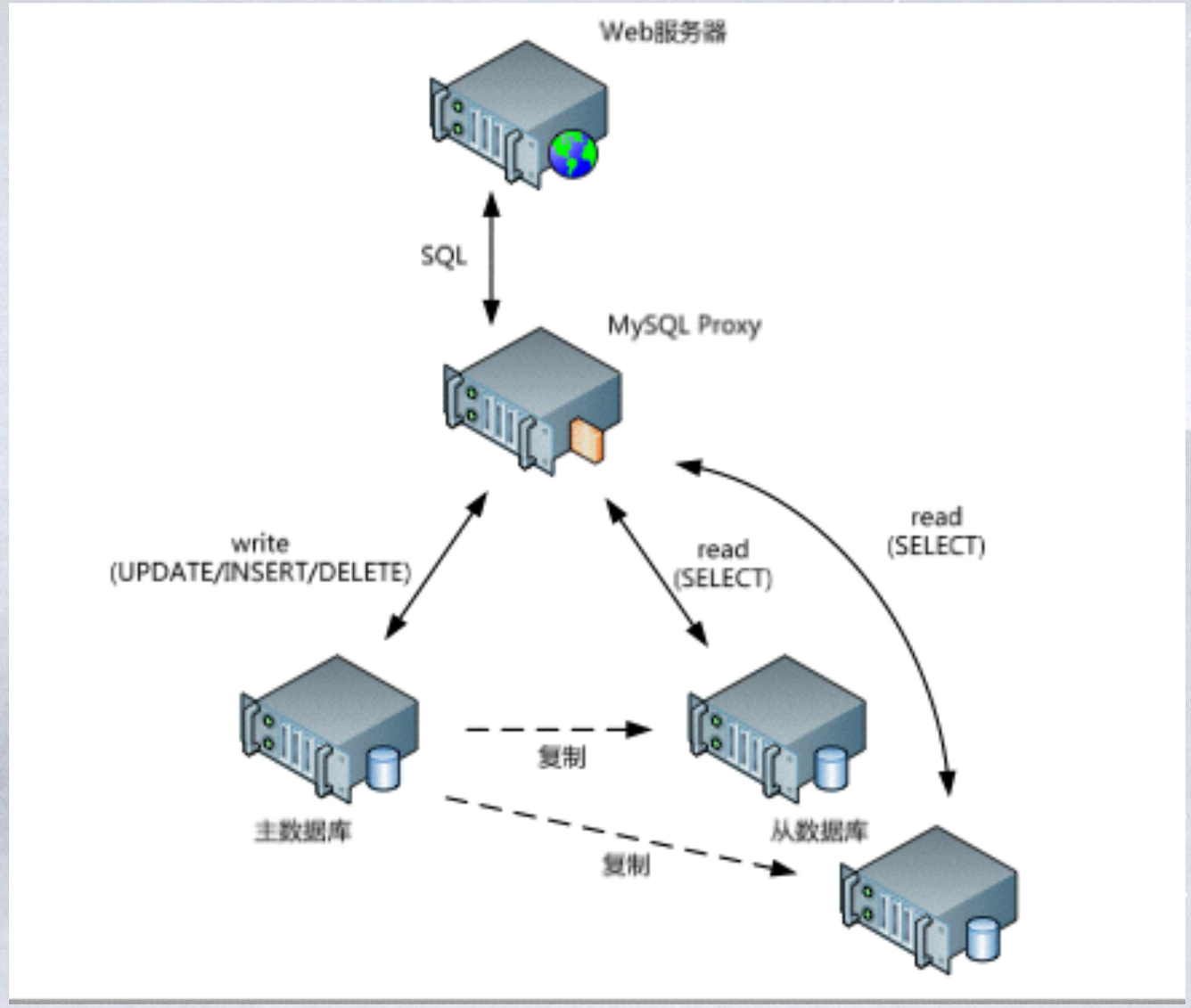


Sharing Assets with a Content Distribution Network



Database Optimization




Web服务器
SQL
MySQL Proxy
write
(UPDATE/INSERT/DELETE)
read
(SELECT)
read
(SELECT)
复制
主数据库
从数据库
复制

References

- Pro PHP Application Performance
- 大规模Web服务开发技术
- http://talks.php.net/show/digg/
- 构建高性能web站点

Thanks!!!