Java is a trademark of Sun Microsystems, Inc.

Java

JavaOne℠

How to Run PHP faster using Java™

Emil Ong
Caucho Technology
http://quercus.caucho.com

caucho

Introduction to Quercus

- 100% Java Implementation of PHP
  - All modules/libraries rewritten in Java
- Interpreted PHP
- PHP compiled to Java
- Runs custom and popular open source PHP apps

Project Goal: Full implementation of PHP in Java

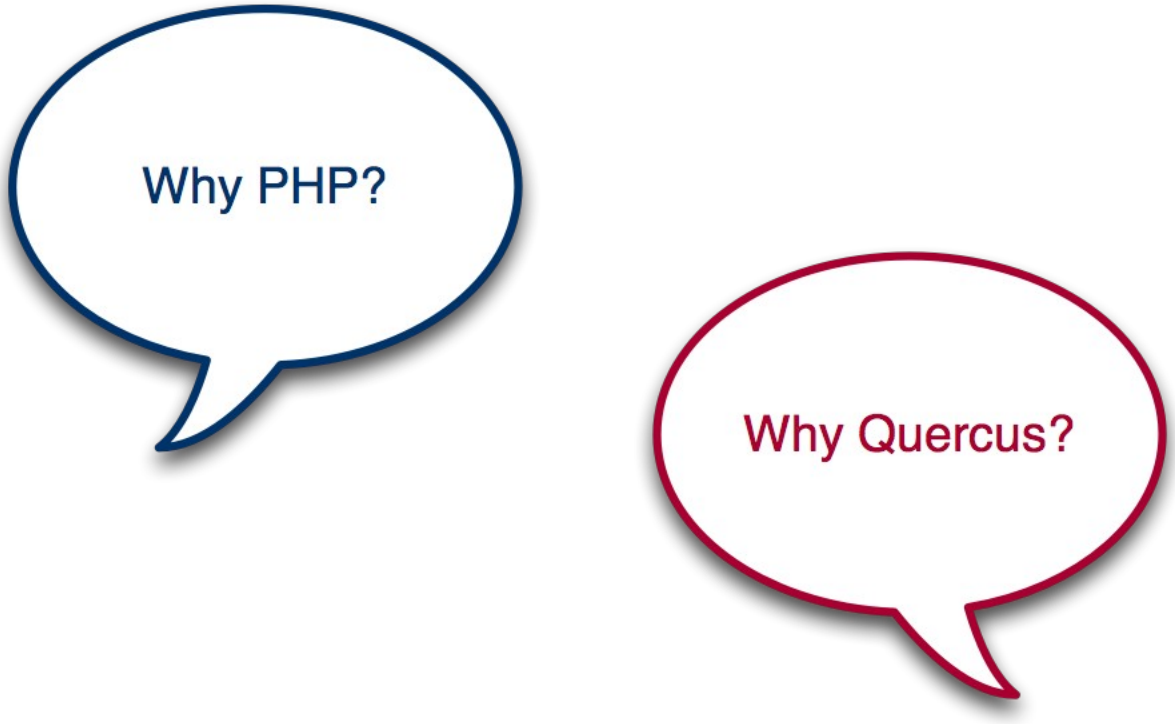
Current status of Quercus

- Introduced in 2006
- PHP 5 and 6 language compatible
- More than 40 of PHP's most popular modules implemented
- Compatible with latest versions of popular PHP applications (MediaWiki, WordPress, et al.)
- Deployed to several production sites (e.g. webmonkey.com, liveprocess.com)

What's new?

- Major performance improvements
  - Now 2x-8x faster than PHP **with** acceleration
  - Reduced memory footprint
- New performance profiling tools
  - Time-based sampling
  - Request-based sampling
- New Java/PHP integration
  - Spring MVC
- A surprise... at the end of the talk

Justifying Quercus



PHP – Just another scripting language on the JVM?

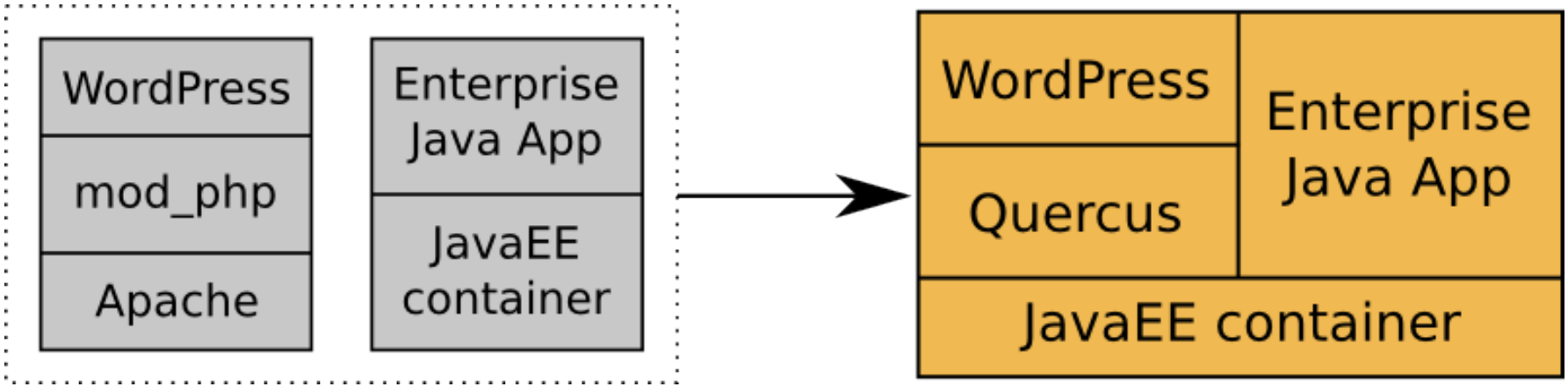
- PHP powers nearly a third of the web
- Large existing developer base
- Top language for off-the-shelf, open source applications
- **Quercus gives Java developers access to these valuable resources**
- **Quercus can grow the Java community by including PHP developers**

When and Why to use Quercus

- You need an application ready right **now** and there's a popular PHP implementation available, but you're a Java shop
  - Blogs (WordPress)
  - Wikis (MediaWiki)
  - CMS (Drupal)
  - Forums (phpBB3)
  - Education (Moodle)

When and Why to use Quercus

- You want to unify existing but separate PHP and Java deployments into one



When and Why to use Quercus

- You need PHP for the frontend and Java libraries on the backend
- You also want them to be able to talk to each other easily
- Examples
  - Spring MVC with PHP view
  - JMS w/eCommerce
  - Mule w/Customer support

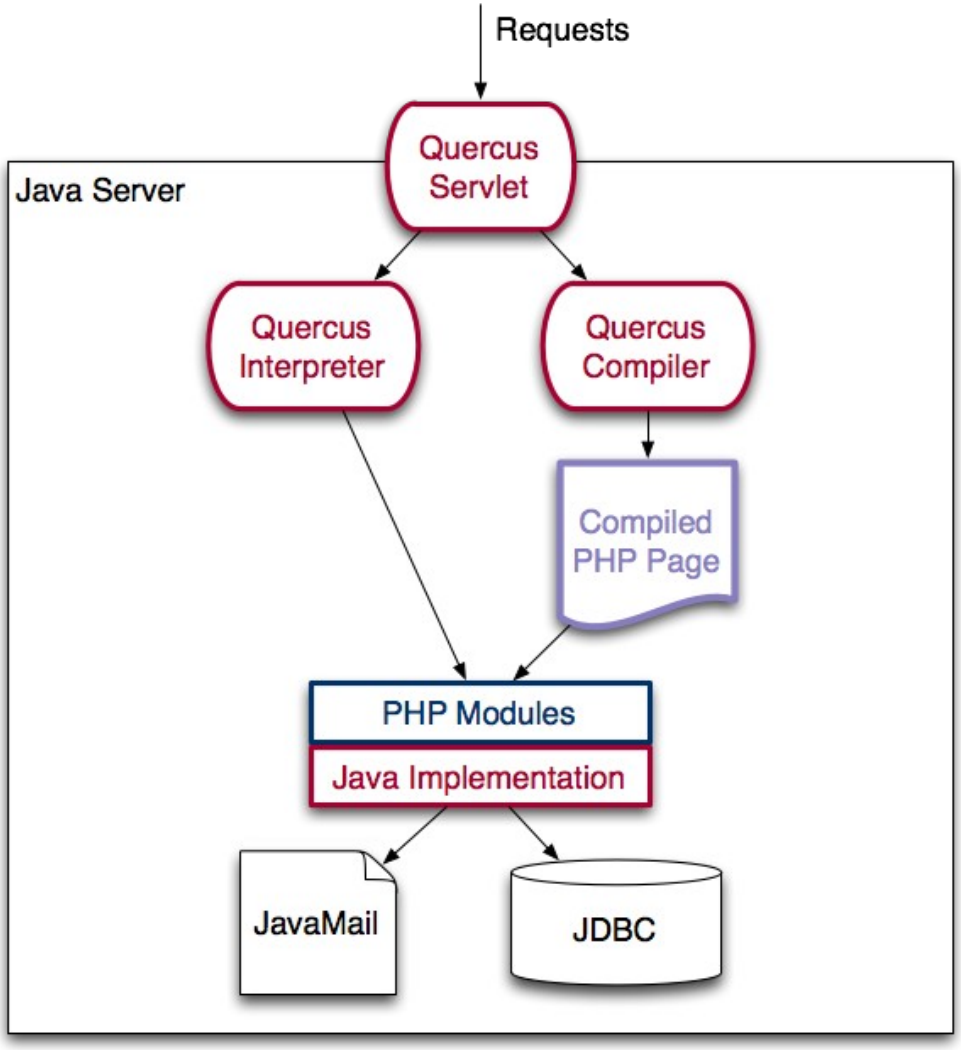
When and Why to use Quercus

- You need to move a PHP-only site to Quercus for improved monitoring, performance, and reliability
- Apache/mod_php has a reputation for being
  - Hard to scale
  - Slow
  - Memory/process heavy
  - Hard to monitor
  - Brutal on databases

How does Quercus Work?

- Architecture
- Using Quercus
- Achieving Performance
- Scaling
- Security

Quercus Architecture



Quercus Engine

- Quercus servlet delegates requests to the Quercus engine to execute
- The engine can execute PHP in either interpreted mode or compiled mode
- The engine is flexible enough to be embedded in numerous architectures outside of servlets
  - MVC views
  - javax.script
  - Messaging

Examples of how Quercus libraries work

- Quercus implements numerous (>40) PHP modules
- Examples
  - File modules delegate to Java file I/O
  - Database modules delegate to JDBC pools
  - Image modules delegate to javax.imageio

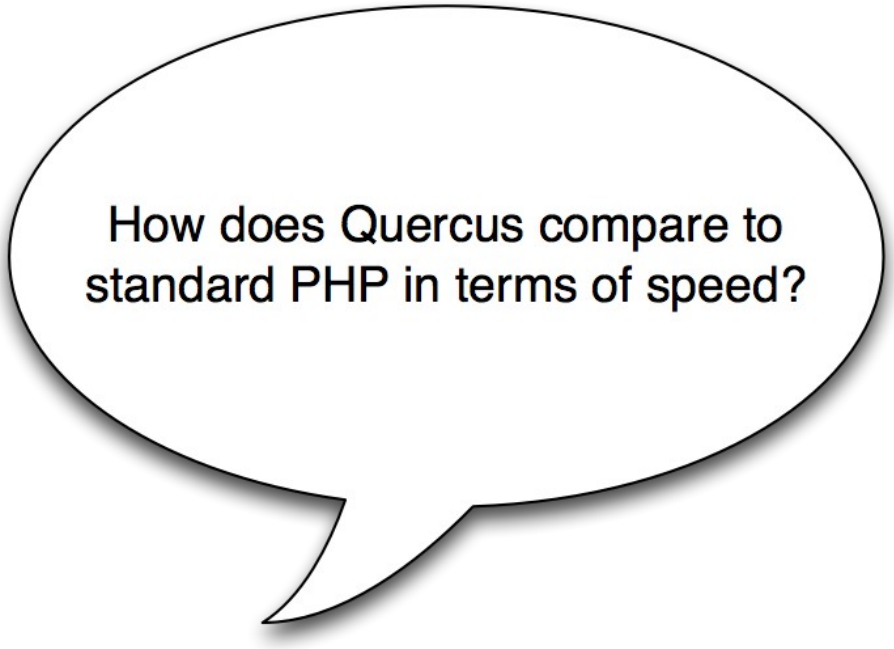
Running Applications with Quercus
Installing WordPress

1. Download WordPress and unpack PHP files to a webapp directory
2. Install MySQL™ Connector/J in WEB-INF/lib
3. Create WordPress database
4. Install using online WordPress install process

Running Applications with Quercus
Installing WordPress

- Can bundle webapp directory into a .war
- Resin includes Quercus, so no need to add extra .jars or configure database manually
- For application server independent installation, include two Quercus .jars, configure JDBC from JNDI, and start the Quercus servlet

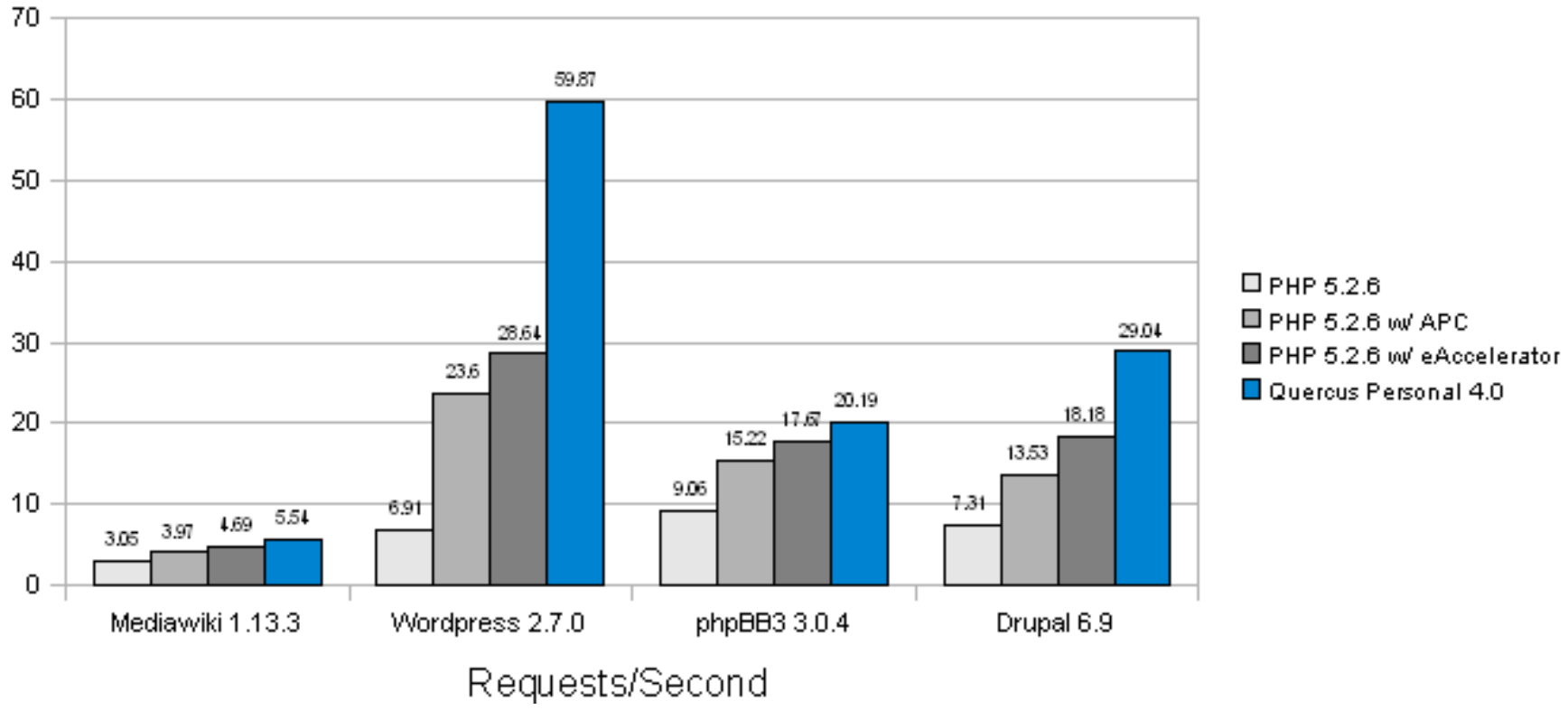
Quercus Performance



Quercus Performance
Synthetic Benchmarks

- Standard “bench.php” available from php.net
  - Function call times
  - Sorting
  - Math
  - Loops
- **Quercus is 796% faster overall than PHP 5.2.6 w/APC**

Quercus Performance
Applications



Why is Quercus faster?

- Java
  - Quercus compiles PHP to Java, which is then compiled to bytecode, and again using JIT
  - Threads instead of processes, reducing request overhead
  - Shared data between requests (regexps, strings, function definitions, et al.)

Why is Quercus fast, even for a Java implementation?

- Static code analysis
- Direct function calls
- Lazy array and string copying
- Precomputed hashes and keys for strings
- Extensive caching
  - File paths
  - Page caching

Why are the synthetic benchmarks faster than the applications?

- Database access
  - The applications shown spend up to 45% of their time in the database
  - Can be fixed with simple object caching
    - Quercus implements the PHP APC API for object caching
  - Profiling can show exactly where caching can help
- Most PHP applications don't do object caching for portability purposes

Other Benefits of Compilation

- Compiling to Java offers not only speed, but static analysis
- This erroneous code would not be found until runtime in a standard interpreted or opcode cached environment:

```
function foo($id) {
  if ($id == 1) {
    break;
  }
  return null;
}
```

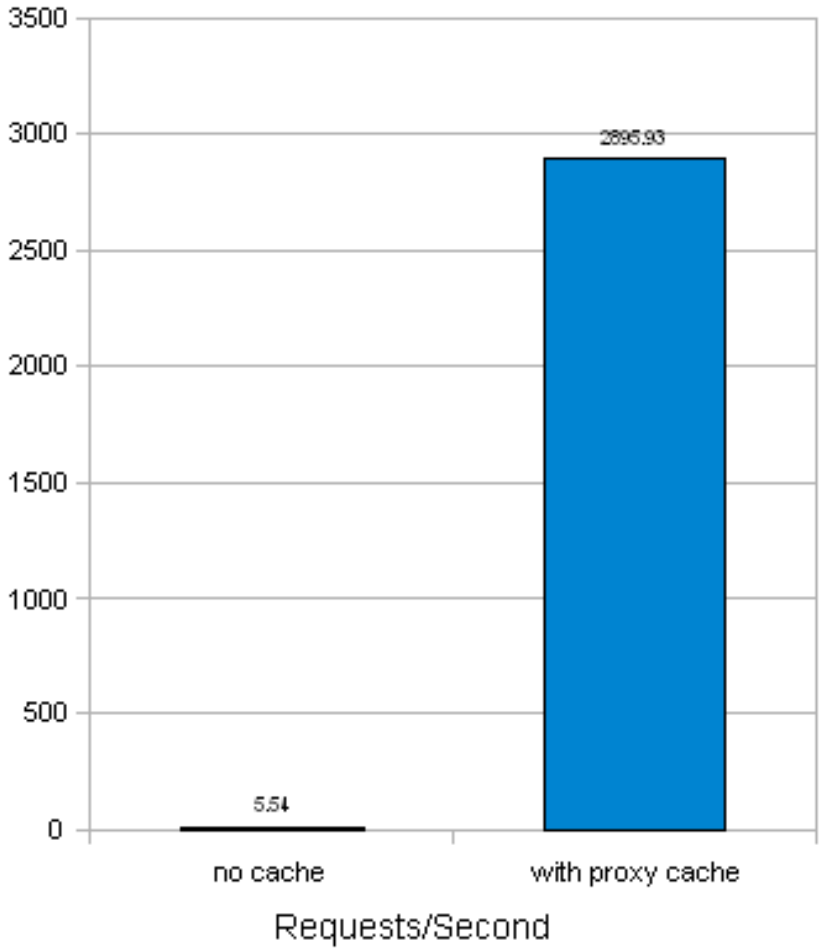
Scaling PHP with Quercus
Databases

- Connection pooling
  - Quercus implementations of PHP database functions use JDBC DataSources
  - App servers like Resin, open source libraries, and vendor JDBC drivers can all provide pooling transparently
  - Apache/mod_php creates a new connection every time by default, slamming the database

Scaling PHP with Quercus
Transparent clustering

- Distributed sessions
  - Quercus can use Servlet Sessions or Resin sessions directly for PHP sessions
- Fail-safe load balancing
  - Java application servers like Resin can provide fail-over, even within a single request
- Cluster-wide proxy caching
  - Reverse proxy caches in servers like Resin can reduce processing time dramatically

Example of Proxy Cache Performance
MediaWiki on Quercus/Resin



Improving PHP security with Java

- Java and JavaEE provide instant security benefits
  - Buffer overflows
  - Servlet filters for input/output verification (e.g. XSS)
  - Can use Java form-based or other authentication
- Hiding the view: PHP files can be placed in WEB-INF/
- Resin can use OpenSSL for speed

Integrating PHP and Java
Calling Java from PHP

- Using Java objects and methods from PHP simply compiles to the equivalent Java code
- "Java" PHP class
  - $foo = new Java("example.Foo", "foo");
- JNDI
  - $foo = jndi_lookup("java:comp/env/foo/foo-a");
- Java CanDI (aka JSR-299, JCDI, WebBeans)
  - $foo = java_bean("foo-a")
- Spring
  - $foo = spring_bean("foo-a")
- Method invocation - $foo->myMethod($a, $b);

Integrating PHP and Java
Calling Java from PHP

- Extend Java classes in PHP
- Catch Java exceptions within PHP
- Use PHP “foreach” to iterate over Java collections and arrays
- Extended “import” syntax allows more natural use of Java classes in PHP

Quercus versus PHP-Java Bridge

- PHP-Java bridge uses XML-based web services protocol to communicate between PHP and Java servers

	PHP-Java Bridge	Quercus	Result

`$a = new java("java.lang.StringBuilder");`
 	
`for ($i = 0; $i < 1000; $i++) {`
`  $a->append("foo");`
`}`
	16.08 requests/sec	357.54 requests/sec	**22 times faster**

`$a = new java("java.lang.StringBuilder");`
 	
`for ($i = 0; $i < 1000; $i++) {`
`  $a->append("foo");`
`  $b = substr($a, $i – 10, 10);`
`}`
	2.94 requests/sec	128.00 requests/sec	**44 times faster**

Quercus in the Real World

- Case Studies
  - LiveProcess
  - CondeNet
- Production use cases

Case Study
LiveProcess

- Developed first standardized software solution designed to help healthcare-related organizations prepare for and respond to emergencies
- Had existing PHP application
  - 8 person years of code
  - Rewrite in Java deemed infeasible
- CTO familiar with Java and PHP
- Java-PHP bridge determined to be unusable for production

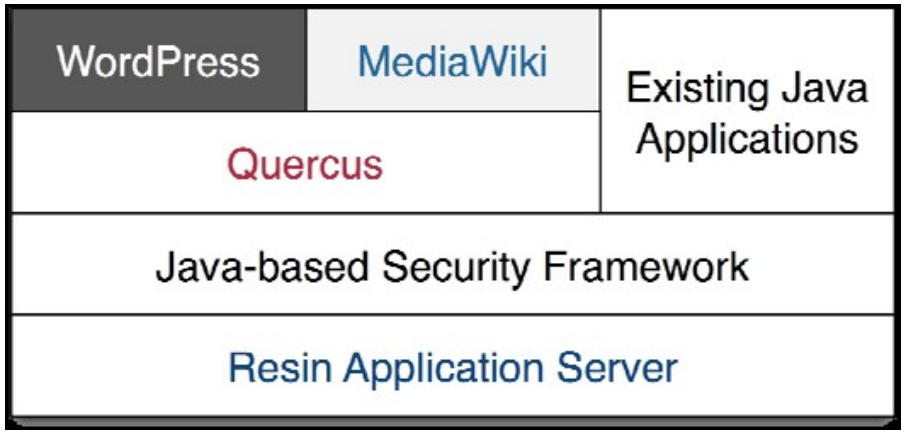
Case Study
LiveProcess

- Early adopter of Quercus, but existing code ran almost immediately
- Used background tasks with PHP – threads
- Java authentication eliminated messy PHP idiom
  - isLoggedIn() at top of every page

Case study
CondeNet

- CondeNet delivers online content for CondeNast magazines
  - Wired.com
  - Webmonkey.com
  - Vogue.com
  - Style.com

Case study
CondeNet



- WordPress blogs
  - http://style.com/stylefile
- MediaWiki
  - http://webmonkey.com/reference
- Java/PHP integration
  - Java-based security module used for both Java and PHP applications
- 5x performance increase using Quercus

Production use cases

- PHP with Java security
- Both open source and custom PHP applications
- Java used to supplement PHP in functionality
- PHP used for speed and simplicity of development

Advanced Quercus

- Lessons learned
  - Java tuning for PHP and other dynamic languages
- Code samples
  - Java/PHP integration

Lessons Learned
JVM tuning for PHP

- Dynamic languages like PHP can stress the JVM memory system
  - Each request creates a large number of objects which will be used only during the request
  - Pattern: Requests fill Eden and trigger frequent Eden GCs
  - Cannot use object pools because of GC marking algorithms

Lessons Learned
JVM tuning for PHP

- Multi-core systems present a unique problem related to allocation
  - With multiple threads all allocating huge numbers of objects from the same memory pool, CPU-level caching becomes difficult
  - Increasing Eden/Heap size doesn't help
  - Observation: Most requests are handled by a single thread

Lessons Learned
JVM tuning for PHP

- Two solutions for multi-core allocation
  - Reduce memory footprint
    - Basic memory analysis
    - Copy-on-write
  - Use obscure JVM feature that allows an allocation pool per thread (TLAB)
    - Each thread/request allocates from its own independent memory pool, reducing contention
    - Improves performance by up to 11% with multiple cores

Lessons Learned
JVM tuning for PHP

- Server mode
  - Just add -server to the JVM args
  - Forces JIT compilation
  - 10% performance improvement immediately
  - Not always enabled by default, especially on virtual machines (n.b. for cloud environments)

Profiling PHP applications

- PHP applications that are compiled to Java have access to existing profilers like JProfiler and YourKit
  - Names of PHP methods/scripts are mangled, but understandable
- Resin includes two profilers for Quercus
  - Time-sampled Java profiler with Quercus name demangling
  - Request-sampled instrumented PHP profiler

Time-sampled Profiler

- Java-based profiler
- Periodically does thread dumps
- Sorts stacks to determine hot spots
- Period and stack depth are configurable
  - e.g. 10ms and 16 frames
- Provides Quercus name demangler
- May be most useful when profiling Java and PHP simultaneously
  - e.g. PHP/Java integration

Request-sampled Profiler

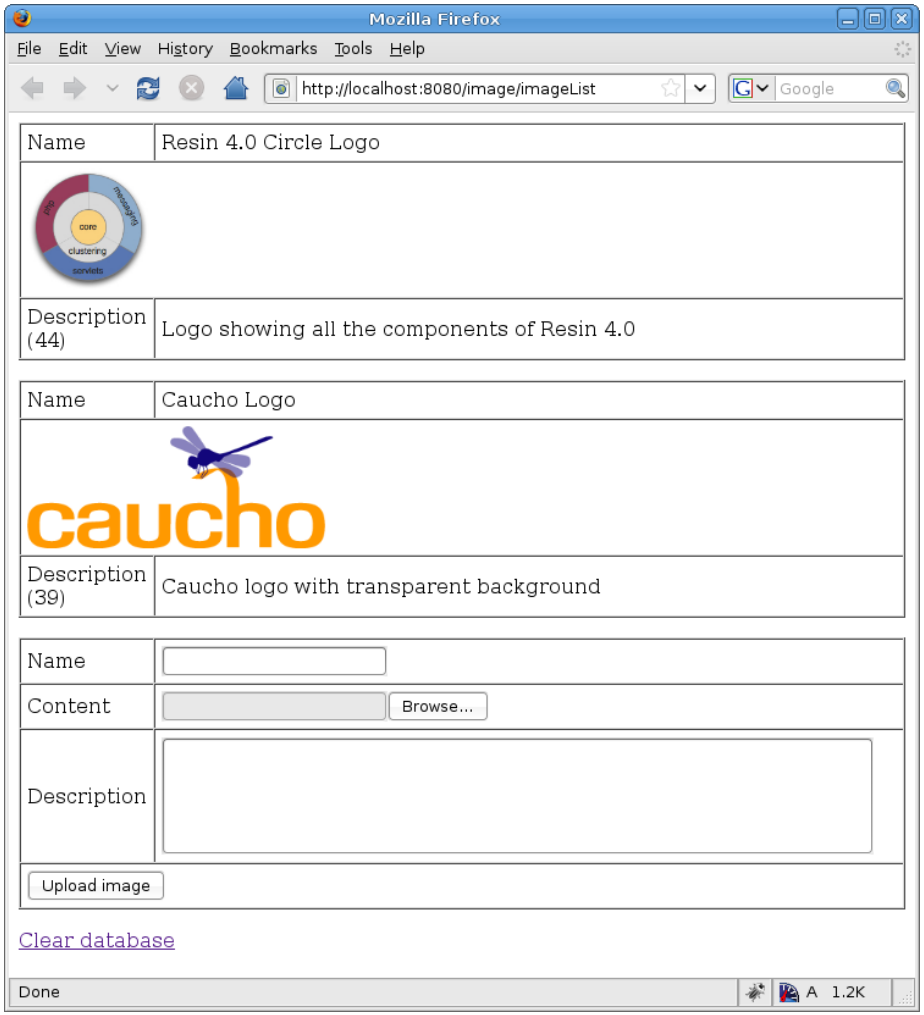
- Each PHP function is compiled twice
  - One instrumented version that times execution
  - One uninstrumented version
- Quercus executes the instrumented version based on probability
  - e.g. 1:1000 requests are profiled
- Fine-grained instrumentation gives complete results of execution

Java/PHP integration example
PHP as a Spring MVC View

- Spring MVC provides an extensible “view” mechanism that previously allowed JSP, FreeMarker, XSTL, et al.
- Quercus Spring View allows simple integration
- Model exposed as global variables in PHP

Java/PHP integration example
PHP as a Spring MVC View

- ImageDB sample
  - Included with Spring
  - Upload image
  - Displays previously uploaded images with metadata



Java/PHP integration example
PHP as a Spring MVC View

- Example JSP view code (ImageDB sample)

```
<%@ page session="false" %>
<%@ page import="java.util.List,java.util.Iterator,
                 org.springframework.samples.imagedb.ImageDescriptor"%>
<html>
<body>

<%
List images = (List) request.getAttribute("images");
for (Iterator it = images.iterator(); it.hasNext();) {
ImageDescriptor image = (ImageDescriptor) it.next();
%>
<table border="1" cellspacing="0" cellpadding="5">
  <tr><td width="10%">Name</td><td><%= image.getName() %> </td></tr>
  <tr><td colspan="2">
      <img src="imageContent?name=<%= image.getName() %>" height="100"></td></tr>
  <tr><td>Description (<%= image.getDescriptionLength() %>)</td>
      <td><%= image.getShortDescription() %> </td></tr>
</table>
<p>
<%
}
%>
```

Java/PHP integration example
PHP as a Spring MVC View

- Same code, rewritten in PHP

```
<html>
<body>

<?php
foreach ($images as $image) {
?>
<table border="1" cellspacing="0" cellpadding="5">
  <tr><td width="10%">Name</td><td><?= $image->getName() ?> </td></tr>
  <tr><td colspan="2">
      <img src="imageContent?name=<?= $image->getName() ?>" height="100"></td></tr>
  <tr><td>Description (<?= $image->getDescriptionLength() ?>)</td>
      <td><?= $image->getShortDescription() ?> </td></tr>
</table>
<p>
<?php
}
?>
```

Late Breaking News

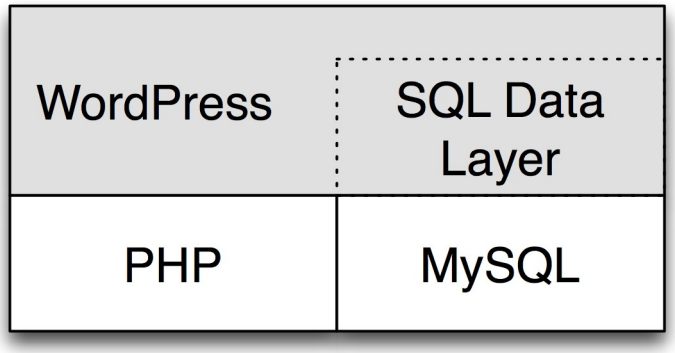
PHP on the Google App Engine with Quercus

- Google App Engine
  - No native PHP support
  - Servlet environment
  - JDO and JPA, but no SQL
- PHP *code* works great with Quercus, but most PHP apps need SQL...
  - Quercus' Java/PHP integration to the rescue!

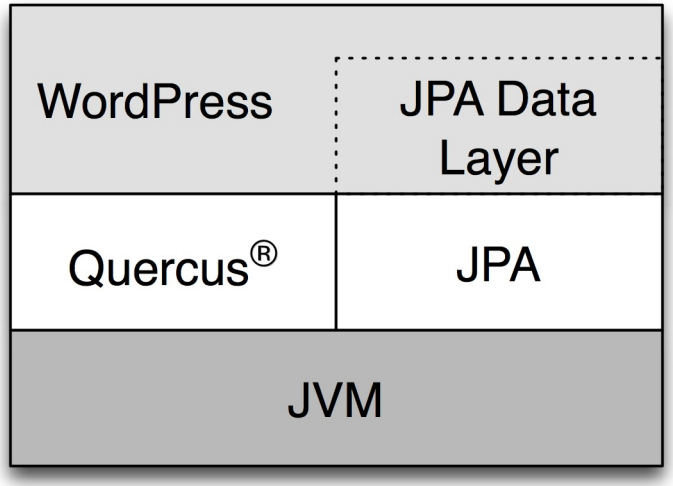
Real Open Source PHP Apps on GAE

- Example: WordPress
  - Convert tables to JPA entities
  - Remove SQL from PHP, add Java integration that queries entities

A



B



Storing Data from PHP on GAE

```
package com.caucho.gae.wordpress271;

import javax.persistence.*;

@Entity
public class Comment
{
  @Id
  @GeneratedValue(strategy=
    GenerationType.IDENTITY)
  private Long _commentID;
  private Long _commentPostID;
  ...
```

```
function get_comments($id)
{
  $query =
    $this->pm->createQuery('
      SELECT FROM Comment WHERE
      _commentPostID = :id AND
      _commentApproved = "1"
      ORDER BY _commentDate');
  $query->setParameter("id", $id);

  $results =
    $query->getResultList();
...
```

Notes and Future Plans for GAE

- PHP apps that use web services may work without modification
- GAE demos, WordPress, and CodeIgniter already shown to work, though with modification
- MySQL compatibility layer is the next step for complete portability

Thank You

Emil Ong
emil@caucho.com
http://quercus.caucho.com

Sun microsystems