Distributed and Persistent Sessions in PHP

Caucho Technology, Inc.

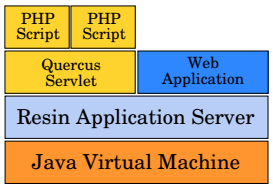
July 2006

Abstract

Quercus, Caucho Technology's 100% Java implementation of PHP, now offers distributed and persistent session management for PHP developers. This technology is built on the solid foundation of Resin, Caucho's proven, high-performance application server. Quercus's session management implementation maintains compatibility with existing applications while at the same time seamlessly adding the ability to distribute and load balance sessions for increased performance and reliability. Truly distributed PHP applications are now possible with Quercus.

Introducing Quercus

Quercus is Caucho's clean-room implementation of PHP5 in Java. With Java-based PHP, developers now have the ability to incorporate the power of Java into their existing PHP applications as well as create a PHP presentation layer for their existing Java applications. Quercus also leverages the stability and security of Java, making PHP more dependable in enterprise applications. Many popular PHP applications are already running on Quercus, including the Drupal content management system and MediaWiki.



The Quercus/Resin software stack

PHP Sessions – Existing Technology

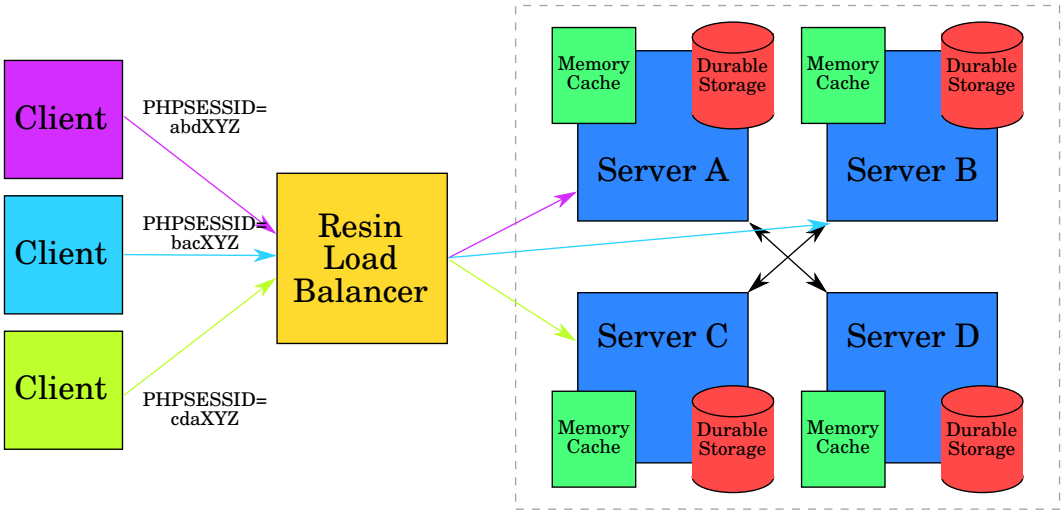
The existing C-based PHP implementation manages sessions by storing and loading session data directly to and from the file system. This infrastructure does not use in-memory session caching for improved performance and does not easily lend itself to distributing sessions over several servers or a cluster. There is an option to use shared memory managed sessions, but this feature does not guarantee safe concurrent session accesses.

Distributed and Persistent PHP Sessions

From technical standpoint, Quercus runs as Java servlet in Caucho's Resin application server. Resin is used by over 5000 customers because it provides the performance and reliability they need for enterprise applications. Quercus is able to offer many of the benefits and features of Resin to PHP developers.

Among these many features is advanced session management. Resin offers a wide range of solutions for session management from simple in-memory and file system based storage to enterprise-ready database and clustered configurations. Using the existing PHP session API, Quercus makes these options available to developers without having to change their code. In all of these configurations, concurrent operations are handled safely.

For the clustered configuration, Resin groups logical sets of servers to maintain a distributed, persistent session store used by Resin and Quercus. Each server has both an in-memory session cache as well as a persistent backing store using the hard disk. In the figure below, there is a cluster with 4 servers. The Resin load balancer sits between the cluster and the clients. Every session that is created is assigned a primary server. When the load balancer receives a request from a client, it first checks the primary server for the session associated with the request. Sessions also have 1-2 randomly chosen backup servers which store the session in the event that the primary fails. Because sessions are assigned to servers in this configuration, they are called *sticky sessions*.



A Quercus/Resin load balanced cluster

The example above shows only 4 servers, but Resin's sticky sessions make it possible to scale to any large cluster configuration. The number of backups for each session remains the same regardless of the size of the cluster, meaning that adding servers to a cluster only increases the capacity of the session management system, not the complexity. Specifically, the network traffic required to access a single session is the same whether the cluster has 4 or 64 servers. In addition, random backups

mean the session management system is able to survive the failure of *any* 2 servers in a cluster before losing a single session.

Another possible approach for session storage is to use a shared database. In fact, implementing this method may be possible using a purely PHP-based solution, though such an approach will not have the benefit of in-memory caching. Resin offers database-backed sessions with in-memory caching for those users who want this capability. However, many applications already have a high database load without also being called upon to manage sessions. Using the Resin clustered session store not only avoids this additional load, it also distributes the load of session management across all the servers. Using the Resin load balancer also gives locality to session accesses.

Feature Comparison

Feature	Quercus	PHP
Easy Java integration	✓	
Memory cached sessions	✓	
File backed sessions	✓	✓
Database backed sessions	✓	✓
In-memory caching	✓	
Distributed sessions	✓	
Safe concurrent sessions	✓	
Number of survivable failures	2	0

Availability

The Quercus session management described in this white paper will be available in Resin 3.0.20. Advanced features such as the database backed sessions, distributed cluster-backed sessions, and load balancing are available in the Professional version of Resin.