

White Paper:

Session Clustering



By Zend Technologies, Inc.

August 2005

Table of Contents

Introduction	3
Background	3
Session Clustering.....	5
Session-persistent load-balancing	5
Sharing session information through NFS	5
Storing session information in a database	6
Summary	6
Zend's Solution	7
Conclusion	8

Introduction

The following paper details Zend Platform's new session clustering module. Zend's solution is the only solution capable of providing linear scalability to growing clusters, while providing unparalleled performance and data integrity.

This paper will describe the alternative solutions available today, as compared to Zend's solution, and the design philosophy and overview of Zend's session clustering module.

Background

The heart of the Internet and the world-wide-web lies with the HTTP (hypertext transfer protocol). This protocol is stateless which means that the web application cannot correlate between browser access to different sections of a web application or web site. In essence this means that users traversing a certain flow in an application cannot be identified between one stage and the next.

To solve this issue the concept of "cookies" was introduced: Cookies are small data structures used by a web site (server) to deliver data to a web client (user); request that the client store the information; and in subsequent accesses, return the information to the web site. Web sites can thus "remember" information about users to facilitate their preferences for a particular site and allow the use of user passwords. Typical use of cookies for example are "remembering" user preferences or the content of shopping carts in an online store.

However, due to the fact that cookies are stored in their entirety on the user's computer, they are not suitable for storing large quantities of data, that would have to be transmitted to the server on every access; nor are they suitable for saving sensitive data that must not be tampered with by the user. The concept of *HTTP Sessions* builds upon the infrastructure of cookies, and associates an identifying, unique cookie ("session id"), stored on a user's computer - with application data that is stored on the server.

PHP supports the concept of sessions through a built-in extension, which abstracts the underlying architecture. The session support allows you to register an arbitrary number of variables to be preserved across requests.

A complication arises when attempting to use PHP sessions in web applications deployed on multiple servers (clusters).

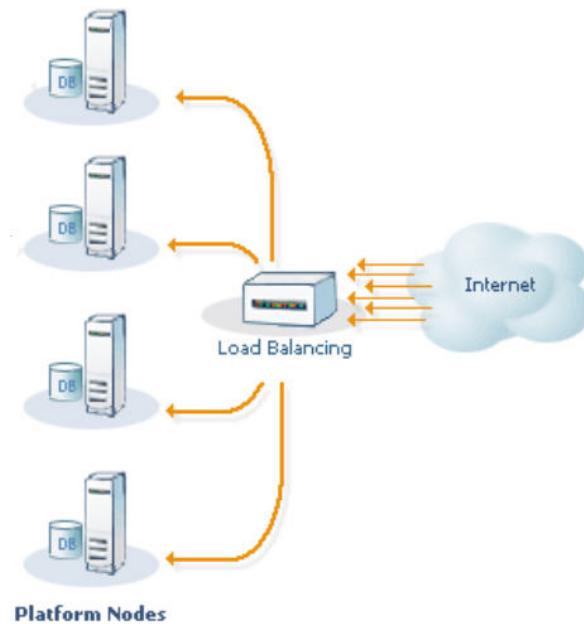


Figure: 1 - SC Environment

A typical configuration of a multi-server cluster involves several servers running identical copies of the PHP application, as well as a load-balancer (a switch that distributes the traffic load between the servers). This allows web applications to scale beyond the confines of a single server, transparently providing heightened performance and response times to the end users.

Due to the statelessness of the HTTP protocol, different HTTP transactions made by the same user may arrive to different servers while the user navigates the application. Consequently, the user's session id may end up being associated with different server-side data on each server, leading to an unpredictable user experience.

Session Clustering

Over time several alternatives have been devised to address the issue described above :

Session-persistent load-balancing

This solution relies on the load-balancing equipment to distribute HTTP connections between the servers so that connections belonging to a particular session are always delivered to the same server. There are several drawbacks to this solution :

- The session affinity is seldom successfully maintained. Without actually inspecting the HTTP protocol it is close to impossible to maintain accuracy in the distribution of connections. It is usually not feasible to inspect the HTTP protocol at the load-balancer because of the performance implications
- When distributing sessions to the same servers, the load is not optimally balanced between the servers in the cluster. This problem is further degraded by the prevalence of Network Address Translation (NAT) in the internet
- Sophisticated load-balancers that can offer some form of session persistence/affinity typically suffer from performance problems, and are prohibitively expensive
- As loads increase the load-balancer becomes a bottle-neck of the cluster's growth. The benefit gained from additional machines in the cluster will be limited by the load-balancer's ability handle additional load.

Sharing session information through NFS

An alternative to relying on the load-balancer for stickiness is sharing the session data (stored in files) between the servers in the cluster. This is accomplished by sharing the session files on an NFS (Network File System) file-server. This approach allows any server in the cluster to successfully serve any request. It also means that using a sophisticated load balancer is no longer necessary, and it is possible to use simpler networking equipment or simple round-robin DNS to distribute the connections between the servers.

There are however some serious drawbacks to this approach:

- Data integrity: the NFS file locking mechanism is inadequate in many cases. The result of this is that session data can easily be corrupted, leading to application faults or unexpected behavior
- Accessing session data through NFS is significantly slower than local session access, it can also put severe demands on the networking infrastructure
- Storing all the session information in one location means that there is a single point of failure in the system: were there to be a failure in the NFS server, the entire PHP application would be unable to function. Highly available NFS appliances with redundant drives and multiple power supplies are available, but these are prohibitively expensive (in most cases, they remain a single point of failure since they typically only have one network interface)
- Scalability issues: placing all the session data in a central location inherently causes a bottleneck in the growth potential of the cluster. The benefit gained by adding additional servers to the cluster will be minimized by the ability of the NFS server to service these additional servers
- Security: NFS is often considered an insecure protocol. Many IT professionals prefer

not to deploy NFS in a production environment where it may threaten the integrity of the application

- Additional hardware to maintain

Storing session information in a database

An additional method of sharing session data between servers is to store the session information in a database. This solves many of the data integrity and security issues presented by working with an NFS server, but still suffers from some of the issues detailed below:

- In addition to being significantly slower than local session access, storing session data in the database puts a lot of additional strain on the database
- A database remains a single point of failure
- Writing session data to the database remains a scalability bottleneck, especially since the scalability of the cluster is impaired not only by the database's ability to serve session data but also by its overall ability being impaired by the load of dealing with sessions

Summary

Standard solutions today do not answer the needs of growing PHP clusters. Specifically they suffer from the following serious drawbacks:

- Bottleneck in scalability: existing solutions have a central component which limits the ability of the cluster to grow, by imposing diminishing returns to the addition of servers into the cluster
- Performance: existing solutions suffer from significant performance penalties that become a critical factor in the deployment of large clusters
- Single point of failure: the reliance of existing solutions on a single central component, means additional risk in the failure of this equipment to the business availability of the application

Zend's Solution

Zend began the design of the session clustering module after repeated requests from customers deploying PHP applications in clusters dealing with heavy load. Being the focal point of best-practices PHP development, Zend became aware of customer complaints ranging from concerns over corruption of session data and erratic application behavior- to requests for our assistance in expanding their clusters. These customers were forced to design very complex and expensive frameworks that distributed session loads through multiple database servers with various levels of redirections. These systems were often bug-prone and slow and threatened to curb the business availability of these applications.

Zend's new session clustering module (a part of the Zend Platform product) is designed to provide a comprehensive solution to the problem of synchronizing session data across a cluster, while avoiding the drawbacks present in alternative solutions - in a cost effective manner.

In Zend's session clustering solution, sessions "reside" on the server where they were first created. These sessions can then be subsequently delivered to other servers in the cluster, by having the alternate server request the session data from the original server. This means that Zend's solution is fully distributed - delivering a high performance, linearly scalable solution that lets customers utilize the most of their existing investment, while ensuring their ability to continue growing over time.

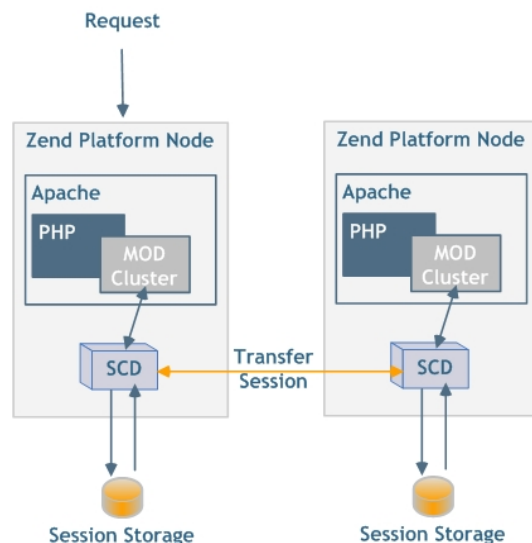


Figure: 2 - SC Architecture

Figure 2 illustrates the components of the session clustering module:

- Session storage : where sessions are stored, this can be either memory, or disk-based storage with a tunable memory cache
- SCD: the Session clustering daemon, that transfers sessions from session storage to the PHP engine and from remote nodes
- mod_cluster: the PHP session handler that communicates with the session clustering Daemon

Zend's session clustering module employs strong locking and data integrity mechanisms to ensure that sessions are never corrupted. Session clustering is also tunable with two different session storage models: disk or memory only, allowing to sacrifice failure recovery for additional performance, or ensuring that session data is available in file format in the eventuality of a crash.

Zend's session clustering module can be seamlessly integrated into any existing PHP application that uses PHP's native session extension - without changing any code. Zend Platform's session clustering solution implements a native PHP session module, and switching between the existing solution and session clustering is simply a matter of changing a PHP.INI directive.

Conclusion

Zend's session clustering module is technically superior to other solutions, currently being the only solution capable of deploying a linearly scalable cluster. Session clustering remains significantly cheaper to deploy and maintain, and offers unsurpassed performance, reliability and data-integrity.

Zend's session clustering solution is the only solution that is deeply integrated within PHP, written and designed by the principle architects of the PHP programming language.

Listed below are the advantages of session clustering over alternative solutions:

- Linearly scalable: there are no inherent bottlenecks in the architecture, and the addition of servers can proportionally increase the scalability and performance of the cluster. Session clustering also puts no additional load on your networking equipment, servers, or database
- Fast: the session clustering module has been benchmarked to be 4 to 6 times faster than alternative solutions in a typical configuration
- No single point of failure: even in the eventuality of complete failure of a server in the cluster, the loss in session data is only that of its share of the sessions
- Safe : Session clustering's inherent data integrity and locking mechanism allow for robust use of sessions, and ensures business availability at high loads
- Fault tolerant: session clustering's ability to store sessions on disk allows it to withstand operating system or web server failures
- Session clustering enables you to leverage your existing investment in equipment: session clustering will work with any load balancing technique ranging from round-robin DNS to sophisticated load balancers. In the event of previous use of load-balancing equipment, session clustering's performance will only improve
- Seamless integration: Zend's session clustering solution allows seamless integration with your existing application code and infrastructure. The same session-based applications will go on working without any code changes - transparently taking advantage of Session Clustering's benefits