



The PHP Company



Session Storage in Zend Server Cluster Manager

Shahar Evron

Technical Product Manager, Zend Technologies

zend[®] server
cluster manager

Welcome!

- All Phones are muted - type your questions into the Webex Q&A box
- A recording of this session will be available on-line
- Today's Agenda:
 - ▶ Introducing Zend Server and ZSCM
 - ▶ An overview of session in PHP
 - ▶ Zend Session Clustering
 - Session High Availability
 - Scaling up and Scaling down with Session Clustering

Zend Server Cluster Manager

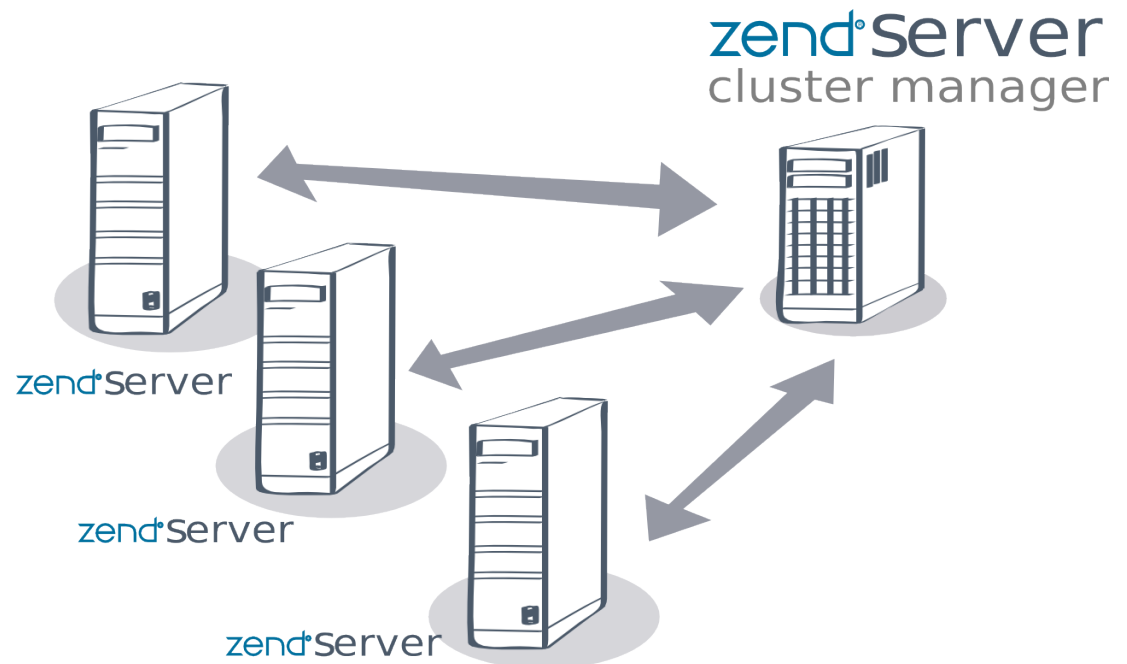
Zend's leading PHP Web Application Server just got Clustered

What is Zend Server?

- Zend's Web Application Server for PHP
- Provides a stable, certified PHP environment as well as multiple advanced features
 - ▶ Some are available for free as part of Zend Server Community Edition (but not the clustering features)
- Consistent environment for Linux and Windows
- Performance: Opcode Caching, Data Caching, Page Caching
- Reliability: Monitoring, Security hot fixes, Code Tracing
- Infrastructure: Job Queuing, Session Clustering

What is Zend Server Cluster Manager?

- A central management point for a cluster of Zend Servers
- Central Configuration Management
- Central Monitoring
- Session Clustering



PHP and Sessions

Representing state over a stateless protocol

User Sessions in HTTP

- HTTP is a stateless protocol - there is no “user session”
 - ▶ Compare HTTP to FTP or SSH
 - ▶ The HTTP protocol does not define any way to correlate between a series of requests and a single user session
 - ▶ Requests from a single user may be sent through one or more TCP connections, and even through multiple connections in parallel
- The good news: this makes HTTP highly scalable and versatile
- The bad news: representing user state is left to the implementation

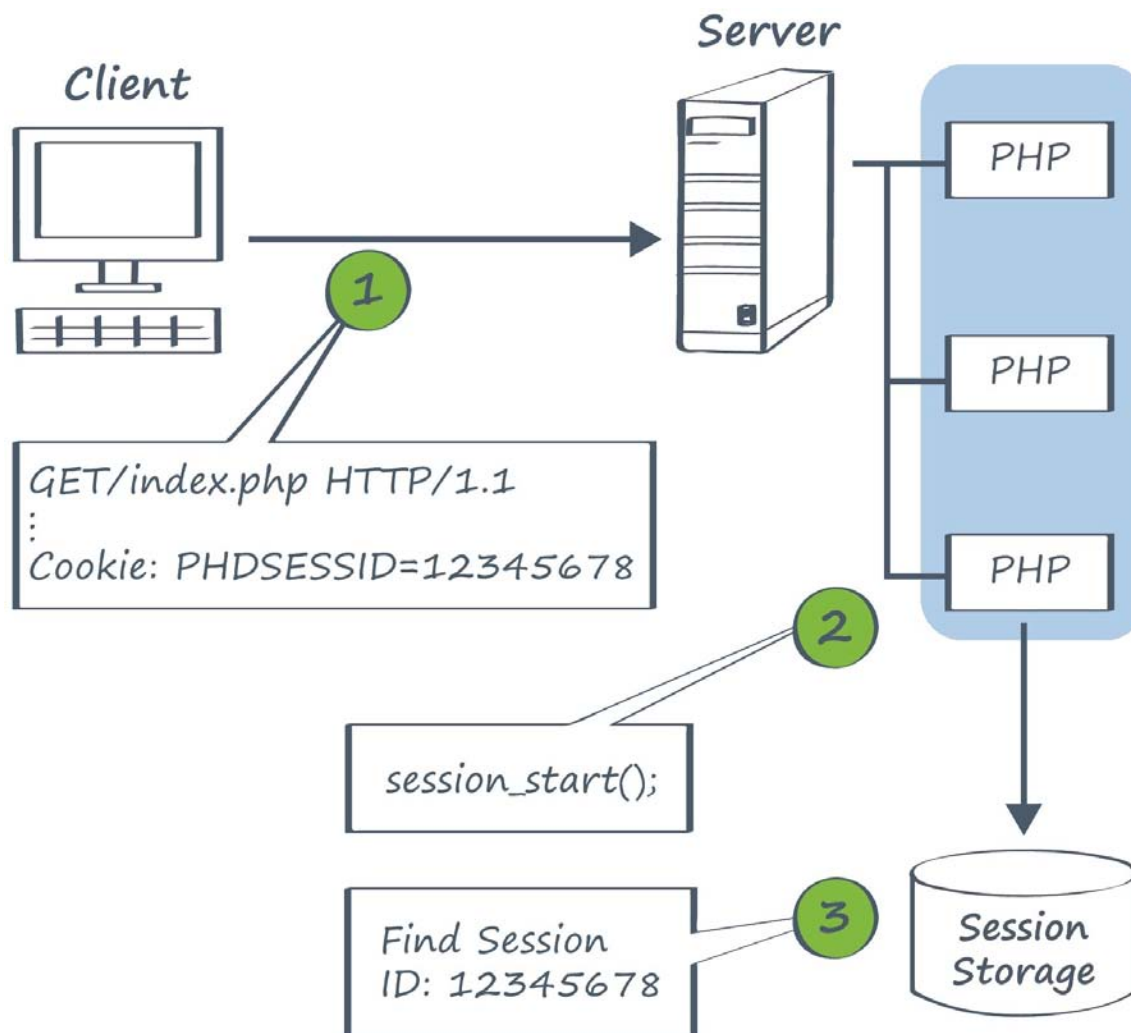
User Sessions in HTTP

- A basic use case: identifying logged in users
 - ▶ If I get two consecutive HTTP requests, how can I know that they are from the same user?
 - ▶ How can I know if that user has logged in or not, before I authorize some actions?
- I need some way to identify that a set of requests are related to each other, and come from the same user
- I need some way to store and access information related to this user which will persist between requests

PHP Sessions to the Rescue!

- PHP provides a built-in mechanism for handling user sessions
- A Session ID identifies a series requests as belonging to a specific user session
- The Session ID is sent by the browser to the server through a Cookie or a GET/POST parameter
- A Session Save Handler is used by PHP to store semi-persistent session data
- There are multiple Session Save Handlers out there, and switching between them usually requires no code changes

Session Handling in PHP



Sessions vs. Cookies

- HTTP Cookies can also be used to store user-specific semi-persistent data and through this represent state
- Unlike PHP Sessions, Cookies are stored on the client side
 - ▶ Data stored in a cookie should not be trusted by the application
- Cookie data is sent back and forth between the browser and the server on each request
 - ▶ Storing larger amounts of data in a cookie is impractical
- Cookies are a very good way to pass a session ID

Different Session Handling Mechanisms

- By default PHP stores session data in local files
 - ▶ Other save handlers can be implemented as extensions or in user space PHP code
- In most cases it is very hard to scale beyond one server using local files storage
 - ▶ A sticky load balancer can help, but has downsides
 - ▶ NFS is a bad idea
- There are several “Cluster worthy” session save handlers out there

Zend Session Clustering

Designed and built for serious session handling

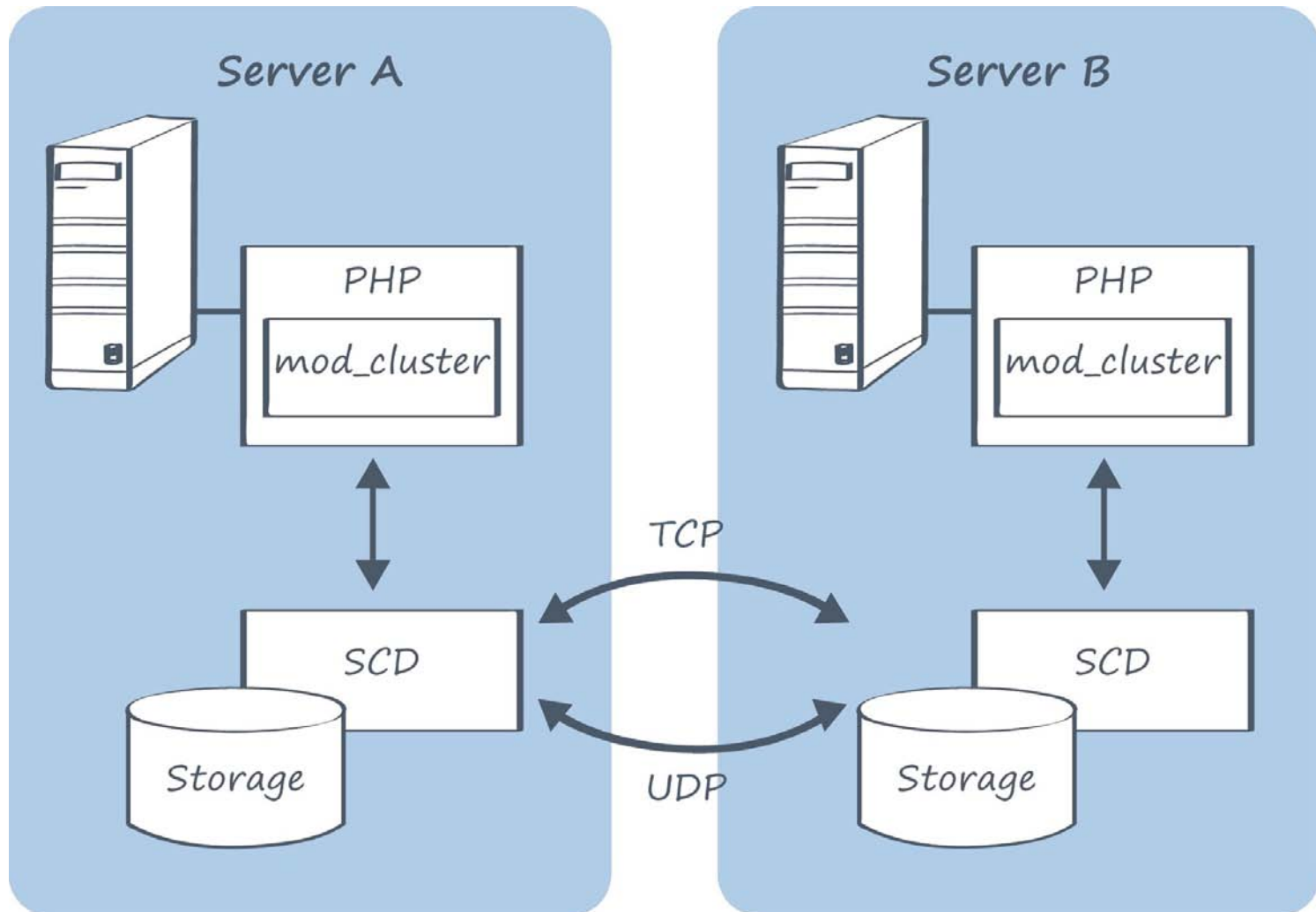
What Session Clustering Does?

- Session Clustering is a PHP Session storage mechanism
 - ▶ Session Clustering is scalable
 - ▶ Session Clustering is fault-tolerant
 - ▶ Session Clustering is relatively fast
 - ▶ Session Clustering is Cloud ready
- Session Clustering is a part of Zend Server Cluster Manager
- Session Clustering can be used transparently by almost any PHP application

How Session Clustering Works?

- Session Clustering is composed of two main parts:
 - ▶ The Session Clustering Extension (*mod_cluster* save handler)
 - ▶ The Session Clustering Daemon (“SCD”)
- Most of the work is done by the daemon
 - ▶ The extension is mostly responsible for getting PHP to talk to the daemon
- Sessions are stored either in memory or on disk
- Daemons in a cluster talk to each other to share sessions
- Each session is stored on two servers: a master and a backup server

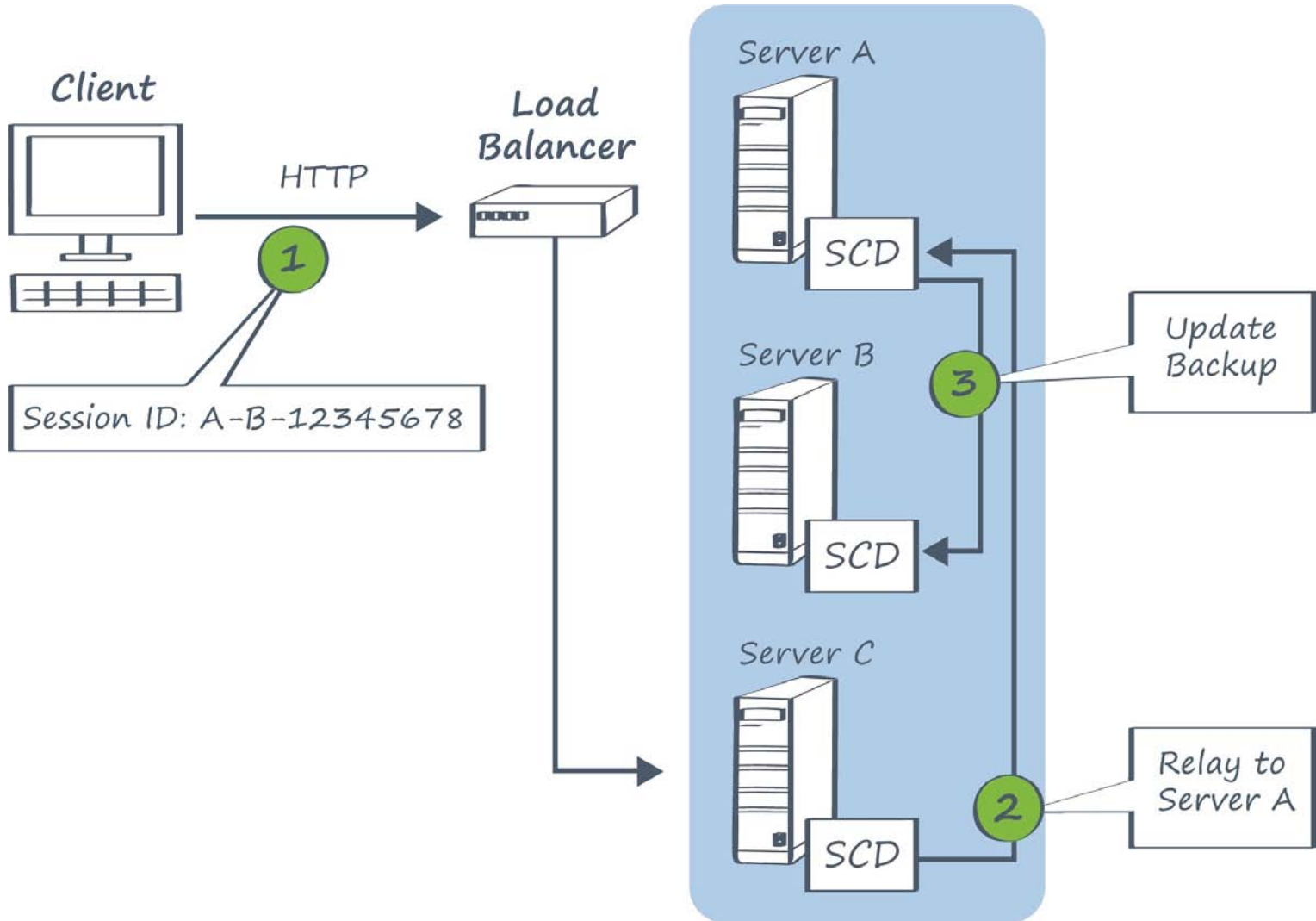
Architectural Overview



Storing and Fetching Session Data

- When a new session is started by a user, it will be created on the server which received the request
- This server will pick a backup server, and will copy the session to it
- The user's Session ID has a special format: it identifies the session's Master and Backup server
- If the request ends up at any server which is not the master, the session is requested from the master

Normal Operation



High Availability

- If the Master Server goes down...
 - ▶ The server receiving the request will get the session from the backup server instead
 - ▶ The backup server will designate itself as the new master
 - ▶ The backup server will pick a new backup server
- If the Backup Server goes down...
 - ▶ The master server will pick a new backup server
- The user's Session ID is changed to reflect these changes

Scalability & Cloud Readiness

- **Session Clustering provides a Graceful Shutdown mechanism**
 - ▶ When a machine is shut down, it will transfer all it's sessions to a different server in the cluster
 - ▶ All cluster members will know to use the replacement server while the original owner is down
 - ▶ This process rarely takes more than 30 seconds
- **Graceful Shutdown allows shutting down machines permanently or for maintenance without losing sessions**
- **Allows scaling down in addition to scaling up - Cloud ready!**

Tuning Tips

- Default settings will work for most people, but...
- If you intend to use Session Clustering in Amazon EC2:
 - ▶ `zend_sc.ha.use_broadcast=0`
- To switch to disk storage:
 - ▶ `zend_sc.storage.use_permanent_storage=1`
- Session Lifetime:
 - ▶ `zend_sc.session_lifetime=3600`

Recap

Recap

- PHP offers a native mean to handle user sessions
- PHP allows you to use different session storage mechanisms, and each has pros and cons
- With Zend Server Cluster Manager, you get Session Clustering - a scalable, fault-tolerant session handler
- If you need to scale up or down and use sessions to store important data, Session Clustering may be right for you!

Where do I go from here?

- Read the Session Clustering Whitepaper:
 - ▶ <http://zend.com/products/server-cluster-manager/resources>
- Download and try Zend Server and ZSCM - evaluation is free
- Questions? Email me: shahar.e@zend.com
 - ▶ Twitter: @shevron, IRC: shevron in #zendserver on FreeNode
- Meet me at ZendCon in Santa Clara, CA, Nov 1-4, 2010

Thank you!