Distributed session handling for stateful PHP web applications in large-scale failover scenarios

Thesis Proposal

Jens Scherbl

September 2, 2016

Abstract

In recent years PHP has become a viable option for large-scale server-side enterprise applications. Some of the platform's fundamental drawbacks related to its default execution model still remain, though, and consequently developers are calling for stateful execution in PHP.

Developers also need to consider load distribution and failover scenarios for enterprise applications, but deploying and running distributed PHP applications in large-scale cluster environments is not yet straightforward. Especially for session handling across large numbers of interconnected application nodes, PHP is still limited to insufficient workarounds and proprietary third-party solutions. For stateful applications, however, there is an opportunity to radically change how session data is handled in PHP – distributed or otherwise.

The thesis gives an extensive overview of the most commonly suggested approaches and provides a conclusive recommendation based on the examined architectures and previously identified requirements. A proof-of-concept is implemented in PHP as a standalone open source library and made available as part of an experimental stateful application framework.

1 Introduction

1.1 Background

PHP – the general purpose scripting language that has dominated server-side web development for many years – has come a long way from its humble beginnings.

Today – over 20 years later – PHP is at the center of a massive and vibrant open source ecosystem, featuring countless libraries, frameworks, content management systems and e-commerce platforms, and drives over 80% [1] of the entire web, in some capacity or another. For large-scale server-side enterprise applications, though, PHP has not always been among the prevalent technologies, and has only become a viable option more recently [2, 3, 4].

In terms of performance, the leading provider of products and services for PHP truthfully admitted that the language would "never be as fast as compiled applications" in 2007 [2], but also suggested that other considerations such as language familiarity, developer efficiency, maintainability and portability would be deciding factors in many cases.

Scott Trent et al. [5] compared technologies for lightweight dynamic content generation in 2008, and found that PHP was in fact 5–10% slower than JSP regarding throughput and performance. The researchers noted, however, that overall application performance often depends on many different aspects under real-world conditions, such as server architecture or the specific task performed. For other purposes, such as working with SOAP-based web services, PHP outperformed Java as a web service engine for larger payloads due to its use of lower-level C implementations for many XML- and SOAP-related tasks [6].

Nikolaj Cholakov [7] still considered the platform to be "not suitable for enterprise scale applications", though, and even claimed that PHP was encouraging poor programming habits at the time.

Newer studies suggest, however, that PHP has improved in terms of software quality in recent years, or was not actually significantly inferior to begin with. For example, Panos Kyriakakis and Alexander Chatzigeorgiou [8] looked into maintenance patterns of popu-

lar open-source projects in 2014 and found that PHP applications undergo "systematic maintenance driven by targeted design decisions". They further noticed that "maintenance is performed with care and in a well-organized manner", and concluded that "PHP does not seem to hinder the adaptive and perfective maintenance activities".

This was also confirmed by Theodoros Amanatidis and Alexander Chatzigeorgiou [9], who analyzed the evolution of PHP applications in 2015 and found that the examined projects were indeed under constant growth and continuous maintenance – as predicted by Lehman's laws of software evolution, which characterize trends in size, changes and quality of evolving software systems. More surprisingly, however, Amanatidis and Chatzigeorgiou did not find any evidence that software quality deteriorates over time for PHP applications:

> Speeding-up development time normally compromises software quality, thereby hindering its sustainability. However, this accumulation of technical debt is not evident for PHP web applications which manage to evolve without increasing their complexity and without demanding increased effort. [9]

The researchers attributed this to PHP's ability to reduce implementation times and enhance developer productivity. It should be noted, however, that this is a presumption mainly derived from older studies of scripting languages in general rather than PHP in particular. Concerning the "latent perception that scripting languages are not suitable for proper software engineering", Amanatidis and Chatzigeorgiou pointed out that "such claims can hardly be found in the scientific literature possibly because they are not backed up by real evidence".

Some of the platform's fundamental drawbacks, often related to its default execution model, still remain, though, and developers are consequently calling for stateful execution in PHP to keep the language competitive.

The thesis goes into more detail on this matter in a dedicated 'Groundwork' chapter, but for a better understanding of what the platform is actually capable of, breaking with its limiting and wasteful per-request lifecycle already enables entirely new kinds of applications for the language, ranging from fully multithreaded application servers[1] to

[1] ↑ `http://appserver.io/`

non-blocking and event-driven concurrency frameworks[2].

As usual, however, with great power there must also come great responsibility [10].

1.2 Motivation

Enterprise applications often serve vast amounts of users and process a rapidly growing mass of data. Hence, PHP developers now also need to consider load distribution and failover scenarios in order to accommodate these increasing demands.

> Due to the critical nature of these applications, scalability and reliability are crucial. Often, entire business processes rely on these applications, and their continued availability can have a direct effect on a company's financial performance. [4]

Unfortunately, deploying and running distributed PHP applications in a large-scale cluster environment is not straightforward – yet.

> Although open source PHP is an excellent starting point for an application, it doesn't come with built-in enterprise features such as clustering, high availability [...] and specialized caching [...] for optimal performance. [4]

Perhaps the most interesting question in this context is how to maintain session state across large numbers of interconnected application nodes – preferably in a way that is transparent to application developers and does not sacrifice consistency or performance.

While other ecosystems have been targeting enterprise customers from the very beginning and already provide reasonable functionality as part of their larger runtime environments, PHP is still limited to insufficient workarounds and proprietary third-party solutions in that regard.

---

[2] ↑ `http://amphp.org/`

1.3 Contribution

Existing approaches for the platform usually rely on shared file systems or external data stores, adding extra complexity to an ever-expanding software stack and generally making the set up and maintenance of the necessary infrastructure more difficult. For stateful applications, however, there is an opportunity to radically change how session data is handled in PHP – distributed or otherwise.

The thesis makes a meaningful contribution in this field by proposing a viable open source implementation based on more appropriate techniques – written directly in PHP and free of dependencies on additional software components.

2 Methods

2.1 Survey

The survey analyzes common use cases involving session state in various types of web applications and further identifies their functional and non-functional requirements, especially in terms of potential failure scenarios in distributed environments.

Different strategies for distributed session handling have previously been evaluated and compared. Thus, most advantages and disadvantages of the existing options are already well understood. Some of the larger software and service providers even offer their own proprietary solutions, combining multiple techniques to avoid the most obvious pitfalls.

The survey gives an extensive overview of the most commonly suggested approaches, provides a systematic review of existing studies and narrows down the field to only the most viable concepts. Most importantly, it reaches a conclusive recommendation based on the examined architectures and the previously identified requirements.

2.2 Concept

The thesis also highlights remaining issues not addressed by the recommended technique and suggests new ideas to further improve the chosen architecture.

A proof-of-concept is implemented in PHP as a standalone open source library and made available as part of an experimental stateful application framework.

To informally test the solution in a large-scale failover scenario, a rudimentary demo application is provided in the form of a Docker[3] container and deployed to a number of interconnected cluster nodes on cloud hosting provider DigitalOcean[4].

References

[1] "Usage of server-side programming languages for websites." `https://w3techs.com/technologies/overview/programming_language/all`, August 2016. 1.1

[2] "An overview on PHP," tech. rep., Zend Technologies Inc., 2007. `https://www.zend.com/topics/overview_on_php.pdf`. 1.1

[3] E. J. Bruno, "The state of PHP in the enterprise," tech. rep., UBM LLC, May 2012. `https://www.zend.com/topics/State-of-PHP-in-the-Enterprise-061212.pdf`. 1.1

[4] E. J. Bruno, "Impact assessment: PHP takes on business-critical apps," tech. rep., UBM LLC, July 2013. `https://www.zend.com/topics/Zend-Impact-Assessment-PHP-July-2013.pdf`. 1.1, 1.2

[5] S. Trent, M. Tatsubori, T. Suzumura, A. Tozawa, and T. Onodera, "Performance comparison of PHP and JSP as server-side scripting languages," in *Proceedings of*

[3] ↑ `https://www.docker.com/`

[4] ↑ `https://www.digitalocean.com/`

the 9th ACM/IFIP/USENIX International Conference on Middleware (V. Issarny and R. Schantz, eds.), (New York, NY, USA), pp. 164–182, Springer-Verlag New York, Inc., December 2008. 1.1

[6] T. Suzumura, S. Trent, M. Tatsubori, A. Tozawa, and T. Onodera, "Performance comparison of web service engines in PHP, Java, and C," in *2008 IEEE International Conference on Web Services*, pp. 385–392, IEEE, September 2008. 1.1

[7] N. Cholakov, "On some drawbacks of the PHP platform," in *Proceedings of the 9th International Conference on Computer Systems and Technologies and Workshop for PhD Students in Computing* (B. Rachev and A. Smrikarov, eds.), (New York, NY, USA), pp. II.7–1–II.7–6, ACM, June 2008. 1.1

[8] P. Kyriakakis and A. Chatzigeorgiou, "Maintenance patterns of large-scale PHP web applications," in *2014 IEEE International Conference on Software Maintenance and Evolution*, pp. 381–390, IEEE, 2014. 1.1

[9] T. Amanatidis and A. Chatzigeorgiou, "Studying the evolution of PHP web applications," *Information and Software Technology*, vol. 72, pp. 48–67, April 2016. 1.1

[10] S. Lee and S. Ditko, "Spider-Man!," in *Amazing Fantasy #15*, New York, NY, USA: Marvel Comics, August 1962. 2