Rogue Wave SOFTWARE

Accelerating Great Code

MAXIMIZE PERFORMANCE AND MITIGATE RISKS WITH PHP 7

PHP is proven to be the platform of choice for business-critical applications. With roots as a community-driven web-scripting language and with limited commercial backing, PHP evolved rapidly into a driver of web and mobile growth. PHP grew fast because it embodied two core tenets: Stay simple and be easy to adopt. Usage statistics indicate that PHP accounts for more than 50 percent of all production web applications, topping 240 million sites according to a Netcraft web server survey. PHP has closed most of the gaps since its early incarnations and has joined Java and .NET as a leading choice for web and business application platforms. With the massively redesigned version 7 getting mainstream adoption, PHP offers new language constructs, significant performance improvements, and lower resource utilization, making it a natural choice for fast-paced business-critical applications. PHP has evolved so much that it is now the runtime environment backing many of the world's largest web sites and serves as the foundation of leading open source application platforms such as Drupal, Magento, and WordPress. As the most common CMS, WordPress accounts for almost 20 percent of web applications[1]. PHP, which is commercially supported and enhanced by Rogue Wave Software, is still fundamentally a true community-led open source project with an active community of contributors driving the evolution of the language.

[1] http://venturebeat.com/2013/07/27/19-percent-of-the-web-runs-on-wordpress/

INNOVATION IS ACCELERATING

The pace of innovation in PHP is high. The PHP core team delivers new releases yearly, allowing the platform to adapt quickly and continuously to best practices and industry requirements. As a result, the performance, quality, and maturity is improved with every new version. From the introduction of version 5, PHP transformed from primarily community projects to adoption in enterprises, gradually making its way to mission-critical applications. This drives PHP developers to quickly consume new methodologies and industry best practices. The enhancements in each new PHP version vary from fundamental language constructs, such as the introduction of namespaces in PHP 5.3 to better support larger applications and modern frameworks, to driving coding practices to achieve better code quality, such as the register_ globals behavior in PHP 5.4. Competing constantly with other established enterprise languages such as Java, .NET, and hyped-up newcomers, PHP delivered significant optimizations in performance and resource utilization. Innovations such as the Rogue Wave byte code acceleration extension were added as a default construct in PHP 5.5 to increase overall execution efficiency.

Constant enhancements to the language and the runtime engine have driven developers to write better, more secure, and efficient code. PHP does provide developers with language elements that make it easier to adopt design patterns and modern coding methodologies but it is still flexible enough to allow some bad habits leading to security or quality issues. Companies that can keep up with new PHP versions are better positioned to deliver higher quality, efficiency, and more secure code and thereby allow their developers to spend more time on developing business-critical functionality.

THE NEXT LEVEL IN APPLICATION PERFORMANCE

Performance is a key consideration of any platform suited for business-critical applications. Practically every new PHP version has demonstrated some performance improvements and resource utilization optimizations. According to many benchmarks, PHP 5.4 is nearly twice as fast as PHP 5.1 and 5.2[2]. Introducing the Zend Server OPCache in PHP 5.5 gives most web requests a substantial performance acceleration by caching the byte code to reduce run time code interpretation. A major effort was launched in PHP 5.5 to significantly optimize the memory consumption of the engine. The performance focus reached another milestone with the adoption of the PHPNextGen (PHPNG) project as the new runtime for PHP 7. PHPNG is an optimized rewrite of the PHP runtime engine developed by Rogue Wave Software with the help of its industry partners, such as Intel, in an effort to accelerate the performance of real-world PHP applications. For real-world applications such as content management and ecommerce, PHP 7 shows a significant performance boost. Execution time is often halved when compared to PHP 5.6 and with 30 percent lower memory consumption. Servers running PHP 7 are able to serve up to three times more Magento requests as those running PHP 5.6.

[2] http://www.lornajane.net/posts/2012/php-5-4-benchmarks

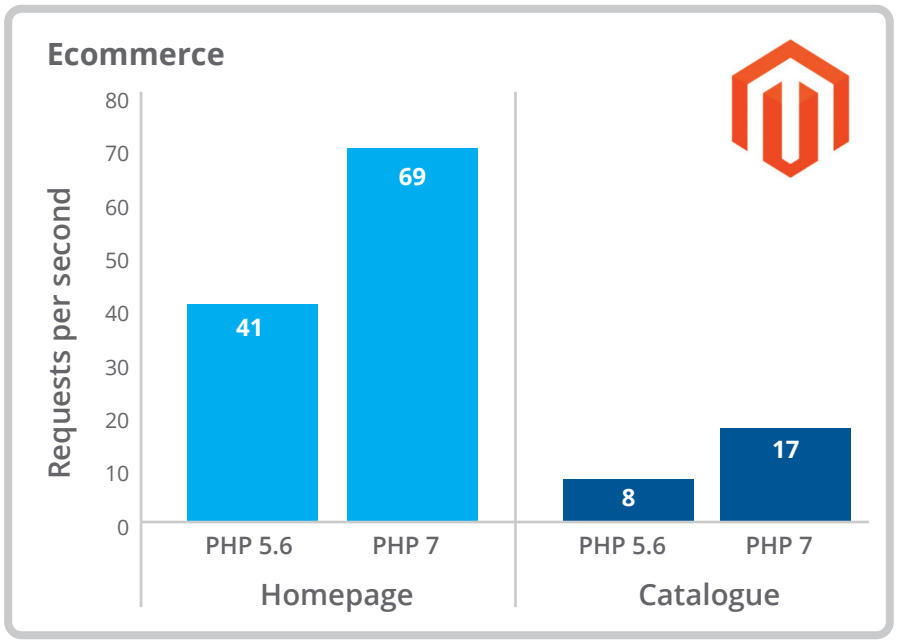

Figure 1: Magento 1.9, PHP 5.6 vs. PHP 7

Drupal 8 runs 72 percent faster and WordPress only executes 25M CPU instructions on a PHP 7 runtime compared to just under 100M doing the same job on older PHP versions[3].

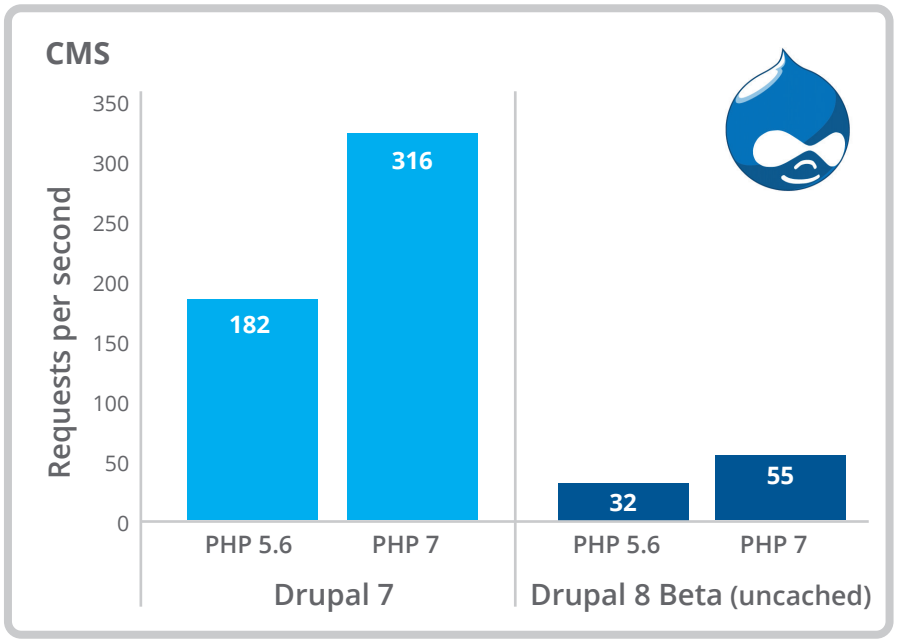

Figure 2: Drupal, PHP 5.6 vs. PHP 7

The benchmarks were also confirmed with larger enterprise applications. When the Tumblr team rolled out PHP 7, they experienced a latency drop of 50 percent, and their CPU load decreased at least 50 percent, often more[4].

[3] https://www.zend.com/en/resources/php7_infographic

[4] https://engineering.tumblr.com/post/152998126990/php-7-at-tumblr

THE RISKY SECURITY CHALLENGE

Application security is a major concern and presents a major risk factor for business. In industries such as ecommerce, financial, and healthcare, data breaches carry regulatory impacts in addition to financial and reputation losses. PHP is no different than the other widely-utilized application runtime platforms. It is a well-maintained open source project with strong contributor support, however multiple vulnerabilities are still identified each year. According to the Common Vulnerabilities and Exposures (CVE) database[5], PHP averages between 20-30 vulnerabilities per year with higher peaks, many of those impacting multiple PHP versions.

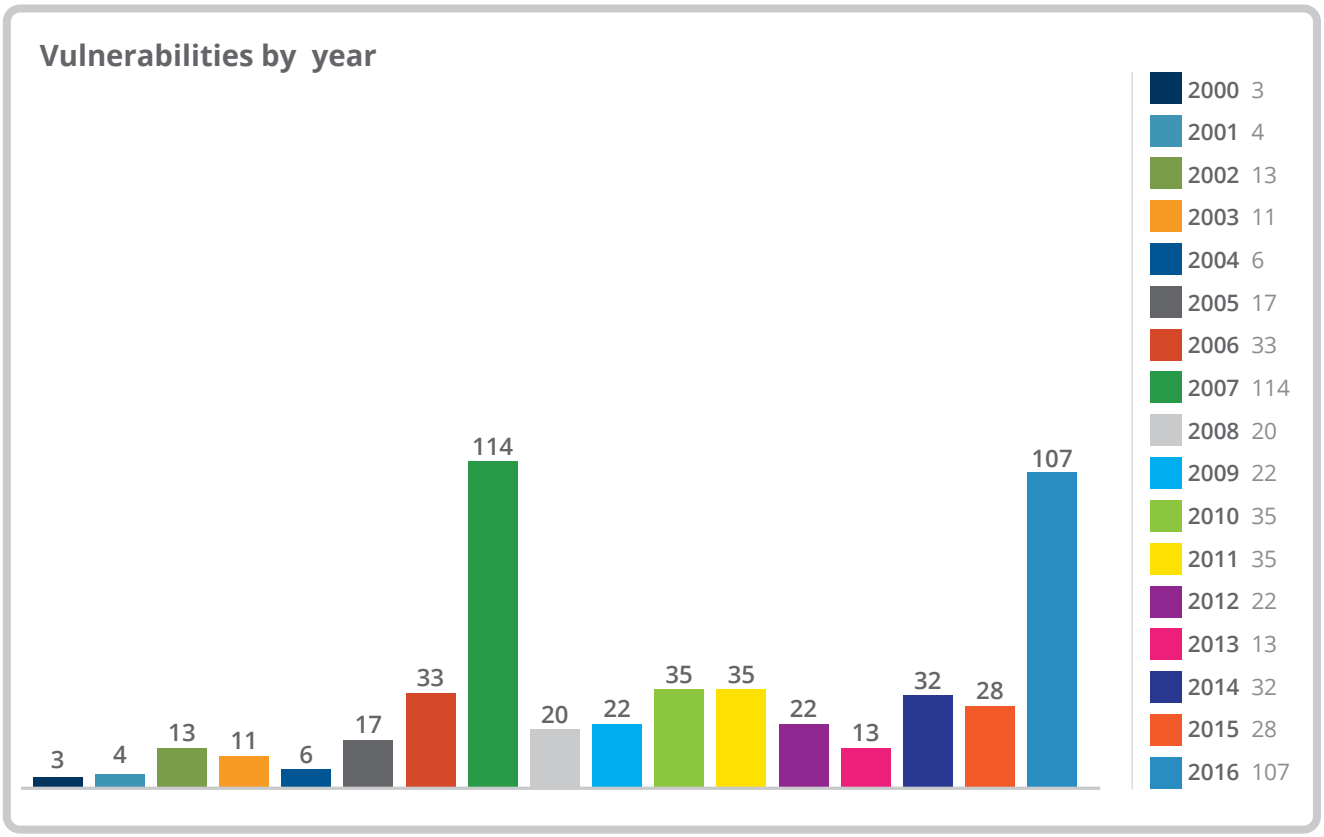

Figure 3: CVE database vulnerabilities by year

The average total cost of a data breach is $3.79 million or $154 cost per lost or stolen record. 47 percent of all breaches are caused by malicious or criminal attacks[6]. Though one cannot completely secure their application, security of a platform is often attributed to the speed in which vulnerabilities are fixed and released to the market. The PHP community responds quickly to identified threats and regularly releases maintenance versions for security vulnerabilities, however it is still very common to encounter production applications running on completely insecure versions of PHP.

There is one caveat. Security fixes are provided for only three years after a version is released. Rogue Wave provides long-term commercial support with guaranteed SLAs for both defects and security issues for no less than five years, providing a significantly longer support period and allowing for additional flexibility in upgrades timing.

[5] http://www.cvedetails.com/product/128/PHP-PHP.html?vendor_id=74

[6] 2015 Cost of Data Breach Study: Global Analysis, Ponemon Institute, 2015

UNSUPPORTED, VULNERABLE ENVIRONMENTS

PHP is the most popular platform for websites, accounting for almost 80 percent of all the websites whose server-side programming language is known[7]. Surprisingly, the vast majority of PHP applications, over 60 percent according to the same data, are running old, unsupported PHP versions. The latest PHP 7 is used by just over 2 percent of known applications, which means that millions of applications are relying on PHP 5.4 or older, exposing the business to a multitude of well-documented security vulnerabilities.

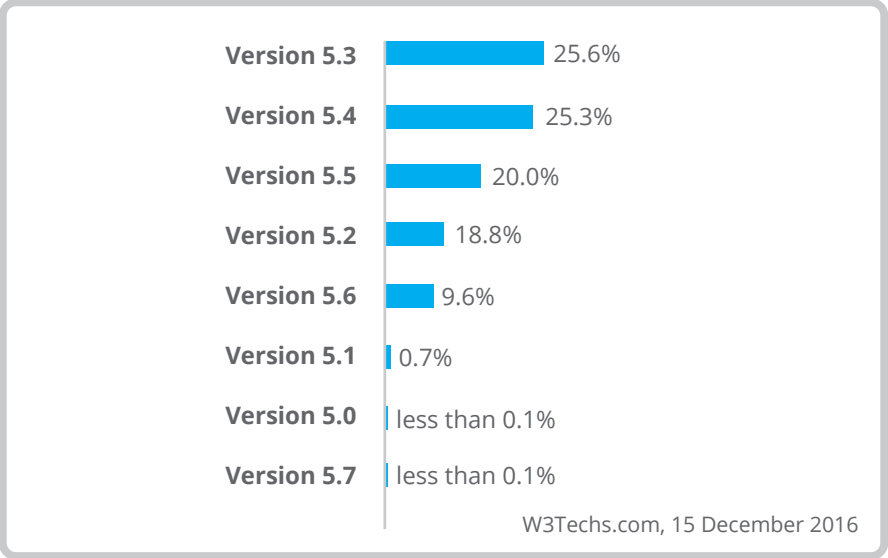

Figure 4: Percentages of websites using various subversions of PHP 5

WHY ARE COMPANIES NOT MODERNIZING?

Running the majority of the code on older, unmaintained, and mostly unsupported runtime environments has been a consistent challenge. Part of it is certainly an awareness issue. PHP is considered very stable, allowing IT organizations to delay proper maintenance and updates to the runtime environment. Most developers and IT managers realize that they need to migrate their applications to improve their performance and security, however they tend to postpone migration projects due to time considerations, lack of expertise, or fear of introducing risks.

Dedicating the time – migration and modernization projects are often a distraction to the business. They do not directly create new business functionality. Most development organizations operate under strict timeframes and do not have the bandwidth to modernize their applications. Migrating between PHP versions requires an effort which includes code changes and testing.

[7] http://w3techs.com/technologies/details/pl-php/5/all

Finding the expertise – lack of experience tends to delay migration projects. Developers and architects focus on their domain of expertise and knowledge of their specific business. They are not typically utilizing tools, nor do they possess the experience required for large migrations and code refactoring projects. Effective modernization projects need specific experience to avoid the typical pitfalls.

Avoiding risks – the constant desire for stability and fear of introducing risks deter organizations from stepping into a code refactoring and modernization process. It's easy to believe that the application works in production delivering its business value, therefore any attempt to modify the code or environment is an opening for problems that could introduce unnecessary risks to the business.

In the 2016 Developer Pulse survey we asked over a thousand developers about their migration plans. About 20 percent have already migrated their code and another 20 percent are in process of migrating. The main reasons for not migrating are incompatibility of the application's custom code or third-party frameworks totalling 60 percent[8]. Addressing these incompatibilities requires time investment, migration expertise, and introduces risks that tend to bring many modernization and optimization attempts to a halt. As the maturity gap widens, migration becomes even more challenging. However, the risks and eventually the costs of not taking any action will eventually catch up.

HOW ARE MIGRATIONS HANDLED?

Migrations are a multistep process where each step depends on the source and target version, scope of the code base, dependencies, and the types of frameworks or extensions used. In general, the projects follow a number of key stages, some being more agile and iterated until quality justifies proceeding to the next phase. Here are the stages:

1. General evaluation of the code base, open source components, PHP extensions, and dependencies, including the main frameworks utilized
2. Build of a new test environment based on the new runtime with its dependencies (database, extensions, web server, queues, etc.)
3. Detailed evaluation of the code to identify potential issues with a combination of automated checkers and manual review:
    - Deprecated functions or language constructs
    - Functionality changes of the same functions between versions
    - Identification of removed and incompatible extensions and APIs
    - Code patterns that need to be modernized
    - Code blocks that were acceptable or ignored previously but could generate fatal errors in PHP 7
4. Definition of a strategy to remediate the identified areas and functions requiring changes

[8] DevPulse 2016 developer survey, Rogue Wave Software, Oct 2016

5. Creation of scripts/automatic checks that prevent accepting new code that is not compatible with PHP 7 in the continuous integration (CI) workflow
6. Refactoring of identified code issues
7. Updating of any required frameworks and dependencies, including an assessment of which frameworks can be upgraded and when incompatible versions require replacement
8. Deployment of code to the test environment and identification of any errors and warnings
9. Address errors and warnings
10. Running of automated test suites to identify functional, performance, and scalability issues
11. Address logical errors resulting from changes in functional behaviors
12. Identification of further performance and functionality optimizations using profiling and code analysis tools
13. Implementation of optimizations to take advantage of new language constructs and features

The application size and its dependencies may add complexity. In general, following a good methodology that covers the significant differences in architecture and language constructs while also providing comprehensive, frequent testing will deliver a successful migration. It also requires investment in researching or developing migration tools, understanding the version differences and common pitfalls, and the development team's time and focus. If your organization does not have the time, expertise or risk tolerance to launch a modernization effort, you can benefit from the expertise of consultants that specialize in migrations.

The Rogue Wave consulting team helps businesses with planning, managing, and delivering their PHP migration and modernization. The migration process is specifically designed to help organizations successfully update their PHP applications to take advantage of enhanced security and performance benefits. Migrating to fully supported PHP 5.6 or PHP 7 from earlier versions often requires re-architecting parts of the application and detailed understanding of the runtime engine. As the commercial entity behind PHP, the methodology we apply saves time by utilizing automated tools developed specifically for identifying code patterns that require refactoring, and manual review of the code by an experienced architect. Automated tools reduce the migration effort by identifying deprecated functions, language constructs, and code patterns that require re-architecting. Understanding potential issues ahead of time accelerates the migration effort and streamlines the transition, saving the developers time and frustration.