Helion

Szybsza Sieć z językami PHP, MySQL i JavaScript

Zaawansowane aplikacje z wykorzystaniem najnowszych technologii

Packt›

Andrew Caya

Tytuł oryginału: Mastering the Faster Web with PHP, MySQL and JavaScript:
Develop state of the art Web applications using the latest Web technologies

Tłumaczenie: Krzysztof Bąbol

ISBN: 978-83-283-5521-7

Copyright © Packt Publishing 2018. First published in the English language under the title 'Mastering The Faster Web with PHP, MySQL, and JavaScript – (9781788392211)'.

Polish edition copyright © 2019 by HELION SA.
All rights reserved.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from the Publisher.

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiejkolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autor oraz Helion SA dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz Helion SA nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel. 32 231 22 19, 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *http://helion.pl* (księgarnia internetowa, katalog książek)

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:
ftp://ftp.helion.pl/przyklady/szysie.zip

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://helion.pl/user/opinie/szysie
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

• Kup książkę
• Poleć książkę
• Oceń książkę

• Księgarnia internetowa
• Lubię to! » Nasza społeczność

Spis treści

Kup książkę Poleć książkę

Kup książkę Poleć książkę

Kup książkę Poleć książkę

Kup książkę Poleć książkę

ROZDZIAŁ 1

Wprowadzenie do koncepcji Szybszej Sieci

„Szybsza Sieć" (ang. *Faster Web*) to określenie używane już od kilku lat do opisywania wielu różnych aspektów wydajności sieci WWW. W książce przyjrzymy się temu zagadnieniu bliżej. Dlaczego jest ważne? Czy jest tym samym, co wydajność? Jak je mierzyć? Od jakiego momentu powinno brać się je pod uwagę podczas pracy nad nowym projektem? Jakie technologie leżą u jego podstaw i jak wykorzystać ich możliwości, by uczynić swój projekt internetowy częścią środowiska Szybszej Sieci?

Pierwszy rozdział tej książki rozpoczniemy od zdefiniowania pojęcia „Szybszej Sieci" i próby lepszego zrozumienia jego formalnych aspektów.

W treści książki przedstawię wiele przykładów kodu, które pozwolą Ci lepiej zrozumieć koncepcje leżące u podstaw Szybszej Sieci. Poświęcimy trochę czasu, aby przyjrzeć się jej źródłom, ocenić jej bieżący rozwój i spojrzeć w przyszłość, ku następnemu ważnemu etapowi.

Na razie rozpoczniemy od instalacji wewnątrz kontenera Docker narzędzi do mierzenia wydajności i profilowania, abyś nauczył się ich używać. Zajmiemy się mierzeniem wydajności oraz ustalaniem, czy strona lub aplikacja WWW jest już częścią środowiska Szybszej Sieci.

W rozdziale tym poruszymy następujące tematy:

- zrozumienie idei Szybszej Sieci i jej znaczenia;
- odróżnianie pojęcia „Szybszej Sieci" od wydajności;
- poznanie sposobów mierzenia szybkości sieci;
- instalacja, konfiguracja i użycie narzędzi do testów porównawczych i profilowania.

Kup książkę Poleć książkę

Istota Szybszej Sieci

W 2009 roku firma Google zapowiedziała swoje zamiary przyspieszenia sieci[1] i podjęła odpowiednią inicjatywę, w ramach której zaprosiła społeczność sieci WWW do zastanowienia się, jak można przyspieszyć internet. Główną przyczyną inicjatywy firmy Google miało być to, że „ludzie chcą szybszych, lepiej reagujących aplikacji". Oświadczenie to zawierało również wykaz zidentyfikowanych przez Google wyzwań, którymi planowano się zająć w pierwszej kolejności. Główne z nich to:

- aktualizacja przestarzałych protokołów;
- poprawa wydajności języka JavaScript;
- wynalezienie nowych narzędzi pomiarowych, diagnostycznych i optymalizacyjnych;
- zapewnienie lepszego dostępu do szerokopasmowego internetu na całym świecie.

Pojęcie Szybszej Sieci a wydajność

Szybszą Sieć można zdefiniować jako szereg cech jakościowych rozwijanych we wszystkich dziedzinach technologii internetowych w celu przyspieszenia transakcji między klientem a serwerem.

Na ile ważna jest szybkość? Dla firmy Google jest bardzo istotna, ponieważ w 2010 roku jej specjaliści odkryli, że każde spowolnienie ma bezpośredni wypływ na ruch na stronie firmy i zyski z reklam. Udało im się określić korelację statystyczną między ruchem a przychodem z reklam, a także liczbę wyników i czas potrzebny na ich uzyskanie. Badania pokazały, że w sytuacji, gdy strona zawierająca więcej wyników ładowała się przez 0,9 sekundy, nastąpił spadek ruchu oraz przychodów z reklam rzędu 20% w stosunku do strony z mniejszą liczbą wyników, ale wyświetlanej w ciągu 0,4 sekundy. Również firma Yahoo potwierdziła, że od 5 do 9% jej użytkowników opuszcza stronę WWW, która ładuje się przez ponad 400 milisekund. Usługa Microsoft Bing notowała 4-procentowy spadek przychodu wtedy, gdy wyniki wyszukiwania dostarczała jedynie o 2 sekundy później. Widać więc wyraźnie, że szybkość nie tylko zapewnia zaangażowanie użytkowników, ale ma też duży wpływ na przychód i ogólne wyniki firmy.

Na pierwszy rzut oka wydaje się, że pojęcie „Szybszej Sieci" jest synonimem wydajności sieci WWW. Ale czy jest tak naprawdę?

Wydajność definiuje się jako sposób działania mechanizmu. Według André B. Bondi [2], „wydajność systemu komputerowego często określa się jako jego zdolność do wykonywania określonych serii działań w szybkim tempie i z krótkim czasem odpowiedzi". Z kolei w książce J. D. Meier i innych, poświęconej testom wydajnościowym [3], stwierdzono, że „testy wydajnościowe to rodzaj testów służących określeniu responsywności, przepustowości, niezawodności i/lub skalowalności systemu przy danym obciążeniu".

Widać więc bardzo wyraźnie, że wydajność jest podstawą koncepcji Szybszej Sieci. Czy jednak zawsze jest jej jedyną właściwością? Jeśli aplikacja ma wykonać dokładną analizę dysku twardego i kończy ją w ciągu niespełna 5 sekund, z pewnością pomyślimy, że coś poszło nie tak.

Kup książkę Poleć książkę

Według Denysa Mishunova [4] wydajność ma również związek z percepcją. Stéphanie Walter [5] w jednej z prezentacji na temat postrzegania wydajności stwierdziła, że „pomiar czasu zależy od chwili i może się różnić w zależności od złożoności wykonywanego zadania, stanu psychicznego użytkownika (stresu) oraz jego oczekiwań określonych przez to, jaki program wykonujący dane zadanie uważa za wzorcowy". Tak więc dobre działanie aplikacji oznacza również, że musi ona sprostać oczekiwaniom użytkownika co do tego, jak powinien działać program komputerowy.

Mimo że wysiłki w ramach koncepcji Szybszej Sieci skoncentrowano najpierw na przyspieszeniu rozmaitych technologii internetowych, w wyniku różnych studiów mających na celu dokładne zmierzenie, jak wydajność stron WWW wpłynęła na przyzwyczajenia i ogólne zachowanie internautów, badacze odeszli od koncepcji czasu obiektywnego, odmierzanego przez zegar, na rzecz czasu subiektywnego, odczuwanego.

W książce tej opiszę koncepcję Szybszej Sieci w odniesieniu do głównych technologii sieci WWW, czyli tych, na których działa od 70 do 80% serwerów WWW na świecie, a mianowicie Apache, PHP, MySQL i JavaScript oraz wszystkich najważniejszych przeglądarek. Te główne technologie sieciowe omówię nie tylko z punktu widzenia programisty, ale także administratora systemów, opisując w ostatnich rozdziałach protokół HTTP/2 i buforowanie zwrotne (ang. *reverse proxy caching*). Pomimo że większa część tej książki dotyczy tylko wydajności sieci WWW, ostatni rozdział obejmuje inne aspekty Szybszej Sieci, dotyczące dobrego, spełniającego oczekiwania internautów projektu **interfejsu użytkownika** (ang. *user interface* — UI).

Pomiar Szybszej Sieci

Zrozumiałeś już, dlaczego wydajność sieci WWW jest bardzo ważnym składnikiem całej koncepcji Szybszej Sieci, która jednak odnosi się nie tylko do wydajności i szybkości, ale zaspokojenia wszystkich oczekiwań użytkowników. Warto teraz postawić sobie pytanie, jak obiektywnie zmierzyć parametry Szybszej Sieci i jakie narzędzia najlepiej się do tego nadają.

Przed pomiarem

Przy omawianiu pomiaru szybkości należy zawsze pamiętać, że zależy ona ostatecznie od sprzętu i że problem niekoniecznie musi leżeć w niezbyt wydajnym oprogramowaniu, jeśli zostało uruchomione na mało wydajnej infrastrukturze sprzętowej. Za większą część łącznego opóźnienia infrastruktury sprzętowej odpowiadają oczywiście zawsze operacje **wejścia i wyjścia** (ang. *input and output* — I/O). Głównymi „wąskimi gardłami", jeśli chodzi o szybkość, są sieć i system plików.

Dostęp do danych na dysku jest na przykład do stu razy wolniejszy niż do pamięci **RAM** (ang. *random access memory*), a bardzo obciążona sieć może spowodować praktyczną niedostępność usług WWW.

Ograniczenia wielkości pamięci RAM również zmuszają do pewnych kompromisów w zakresie szybkości, skalowalności i dokładności. Aby uzyskać najwyższą możliwą wydajność można przecież buforować większą część danych aplikacji i ładować wszystko do pamięci operacyjnej. Czy jednak w każdych okolicznościach będzie to rozwiązanie optymalne? Czy da się utrzymać

Kup książkę Poleć książkę

szybkość w przypadku dużego obciążenia? Czy w przypadku dużej zmienności danych będą one właściwie odświeżane? Na te pytania niemal na pewno nie da się odpowiedzieć pozytywnie. Optymalna szybkość to zrównoważenie czystej prędkości, rozsądnego zużycia pamięci i akceptowalnej trwałości danych.

Mierzenie wydajności w celu określenia optymalnej szybkości programu komputerowego to sztuka znalezienia idealnej równowagi w ramach konkretnych reguł biznesowych oraz dostępnych zasobów dzięki implementacji właściwych kompromisowych rozwiązań i późniejszemu ich dopracowaniu.

Pierwszym krokiem w zakresie oceny wydajności powinna być zatem analiza dostępnych zasobów i określenie dolnej i górnej granicy wydajności sprzętu. Ponieważ pracujemy nad wydajnością sieci WWW, to w celu wykonania tego pierwszego kroku musimy przeprowadzić testy samego serwera.

Kolejny etap to profilowanie aplikacji WWW w celu analizy wydajności każdego z jej elementów składowych oraz określenie, jakie fragmenty kodu aplikacji są dalekie od doskonałości i powinny zostać zoptymalizowane.

Testy porównawcze i profilowanie

Testowanie serwera WWW polega na ocenie jego wydajności przy pewnym obciążeniu. Profilowanie oprogramowania to analiza zużycia pamięci i czasu wykonywania w celu optymalizacji wewnętrznej struktury programu.

W tej części rozdziału zainstalujemy i przetestujemy kilka narzędzi do testowania wydajności serwera i profilowania kodu źródłowego, które zostaną opisane w kolejnych rozdziałach.

Praktyczne wymagania wstępne

Aby uruchomić zawarte w tej książce kody źródłowe, należy zainstalować program **Docker** (*https://docs.docker.com/engine/installation/*).

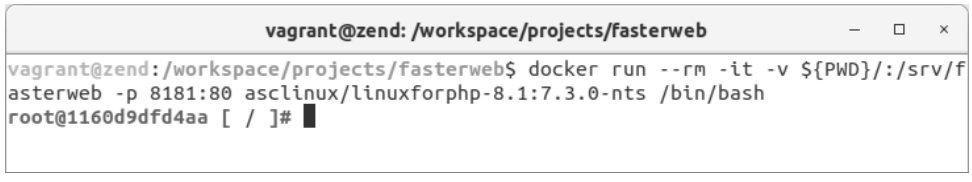
Docker to platforma kontenerów z oprogramowaniem, pozwalająca łatwo łączyć się ze znajdującymi się w komputerze narzędziami w izolowanym, zaawansowanym środowisku, podobnym do trybu chroot. W odróżnieniu od maszyn wirtualnych, kontenery nie zawierają całego systemu operacyjnego, ale tylko pliki binarne wymagane do uruchomienia danego oprogramowania. Program Docker można zainstalować w systemie Windows, Linux lub na komputerze Mac. Należy jednak zauważyć, że niektóre jego funkcje, między innymi w pełni funkcjonalne usługi sieciowe, w systemie macOS są wciąż niedostępne (*https://docs.docker.com/docker-for-mac/networking/#known-limitations-use-cases-and-workarounds*).

Zasadniczą wersją obrazu platformy Docker używaną w tej książce jest Linux for PHP 8.1.3 (*https://linuxforphp.net/*), z niezapewniającą bezpieczeństwa wątków wersją środowiska PHP 7.3.0 i bazą danych MariaDB (MySQL) 10.2.8 (*asclinux/linuxforphp-8.1:7.3.0-nts*). Najpierw zainstaluj na komputerze program Docker i pobierz z serwera FTP wydawnictwa Helion przykłady kodu z książki. Następnie, w celu uruchomienia właściwego kontenera, w Terminalu obsługującym składnię powłoki bash wydaj następujące polecenia:

Kup książkę Poleć książkę

```
# cd fasterweb
# docker run --rm -it \
-v ${PWD}/:/srv/fasterweb \
-p 8181:80 \
asclinux/linuxforphp-8.1:7.3.0-nts \
/bin/bash
```

Po wykonaniu tych poleceń pokaże się interfejs linii poleceń kontenera Linux for PHP:



Uwaga dla użytkowników systemu Windows: W powyższym wywołaniu narzędzia `docker` zamień (w opcji dotyczącej współdzielonych woluminów) fragment `${PWD}` na pełną ścieżkę do katalogu roboczego (np. */c/Users/fasterweb*), bo inaczej nie uda się uruchomić kontenera. Upewnij się też, że w ustawieniach narzędzia Docker zostało włączone współdzielenie woluminów. Ponadto jeśli program Docker został uruchomiony w systemie Windows 7 lub 8, kontener nie będzie dostępny pod adresem */localhost:8181*, a tylko *http://192.168.99.100:8181*.

Wszystkie przykłady podane w tej książce są zawarte w folderach o nazwach zgodnych z numerem rozdziału. Aby uruchomić kod z danego rozdziału, powinieneś najpierw zmienić katalog roboczy. W przypadku tego rozdziału wprowadź więc w wierszu poleceń kontenera poniższe polecenia:

```
# mv /srv/www /srv/www.OLD
# ln -s /srv/fasterweb/r01 /srv/www
```

Po przejściu do kolejnego powinieneś wprowadzić następujące komendy:

```
# rm /srv/www
# ln -s /srv/fasterweb/r02 /srv/www
```

W przypadku kolejnych rozdziałów postępuj podobnie.

Jeśli zaś podczas optymalizacji kodu chciałbyś używać technologii wielowątkowych, skorzystaj z bezpiecznej wątkowo wersji kontenera Linux for PHP (*asclinux/linuxforphp-8.1:7.3.0-zts*).

Jeśli chcesz, możesz uruchomić kontener w trybie odłączonym (opcja `-d`). Pozwala to wykonywać komendy `docker exec` w stosunku do tego samego kontenera w wielu powłokach poleceń oraz utrzymywać jego działanie niezależnie od tego, czy masz uruchomiony Terminal.

Ponadto w celu stworzenia nowych obrazów wszelkie zmiany wprowadzone do kontenera należy zatwierdzić poleceniem `docker commit`. Obrazy można później uruchomić poleceniem `docker run`. Jeśli nie jesteś obeznany z wierszem poleceń programu Docker i komendą `run`, zapoznaj się z dokumentacją (w języku angielskim) dostępną na stronie *https://docs.docker.com/engine/*

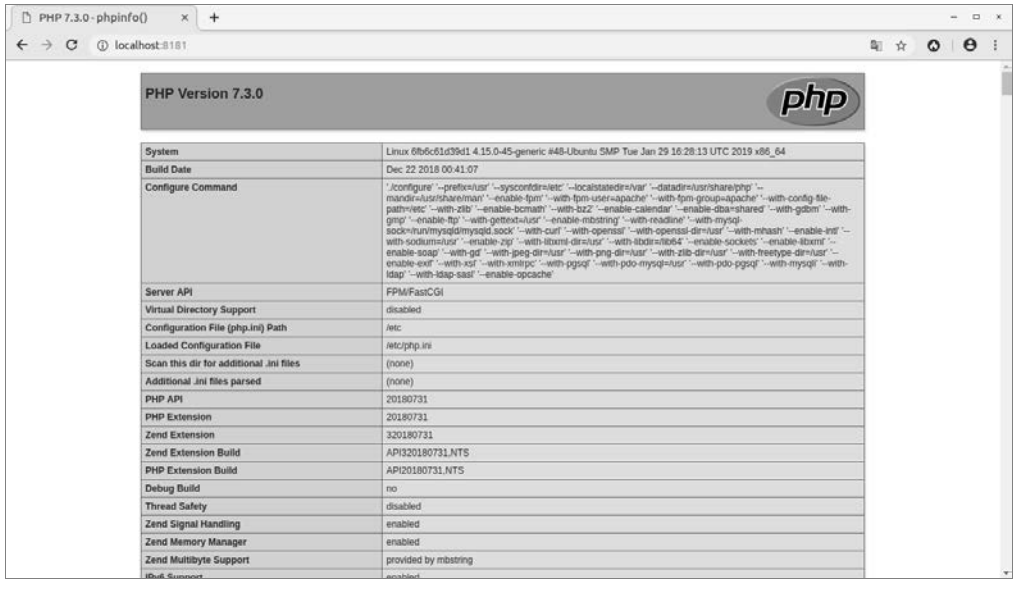
Kup książkę Poleć książkę

reference/run/. Możesz także skorzystać z którejś spośród opublikowanych przez wydawnictwo Helion książek na ten temat (np. *Docker. Praktyczne zastosowania* Karla Matthiasa i Seana P. Kane'a).

W celu uruchomienia wszystkich usług, które będą potrzebne podczas lektury tej książki, stworzymy skrypt testowy, który pozwoli upewnić się, że wszystko działa tak jak powinno. Wprowadź poniższe polecenia:

```
# cd /srv/www
# /etc/init.d/mysql start
# /etc/init.d/php-fpm start
# /etc/init.d/httpd start
# touch /srv/www/index.php
# echo -e "<?php phpinfo();" > /srv/www/index.php
```

Następnie przejdź w swojej ulubionej przeglądarce na stronę *http://localhost:8181/*, a zobaczysz poniższy wynik:



System	Linux 6fb6c61d39d1 4.15.0-45-generic #48-Ubuntu SMP Tue Jan 29 16:28:13 UTC 2019 x86_64
Build Date	Dec 22 2018 00:41:07
Configure Command	'./configure' '--prefix=/usr' '--sysconfdir=/etc' '--localstatedir=/var' '--datadir=/usr/share/php' '--mandir=/usr/share/man' '--enable-fpm' '--with-fpm-user=apache' '--with-fpm-group=apache' '--with-config-file-path=/etc' '--with-zlib' '--enable-bcmath' '--with-bz2' '--enable-calendar' '--enable-dba=shared' '--with-gdbm' '--with-gmp' '--enable-ftp' '--with-gettext=/usr' '--enable-mbstring' '--with-readline' '--with-mysql-sock=/run/mysqld/mysqld.sock' '--with-curl' '--with-openssl' '--with-openssl-dir=/usr' '--with-mhash' '--enable-intl' '--with-sodium=/usr' '--enable-zip' '--with-libxml-dir=/usr' '--with-libdir=/lib64' '--enable-sockets' '--enable-libxml' '--enable-soap' '--with-gd' '--with-jpeg-dir=/usr' '--with-png-dir=/usr' '--with-zlib-dir=/usr' '--with-freetype-dir=/usr' '--enable-exif' '--with-xsl' '--with-xmlrpc' '--with-pgsql' '--with-pdo-mysql=/usr' '--with-pdo-pgsql' '--with-mysqli' '--with-ldap' '--with-ldap-sasl' '--enable-opcache'
Server API	FPM/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc
Loaded Configuration File	/etc/php.ini
Scan this dir for additional .ini files	(none)
Additional .ini files parsed	(none)
PHP API	20180731
PHP Extension	20180731
Zend Extension	320180731
Zend Extension Build	API320180731,NTS
PHP Extension Build	API20180731,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	enabled
Zend Memory Manager	enabled
Zend Multibyte Support	provided by mbstring

Jeśli nie widzisz tej strony, spróbuj rozwiązać problem z instalacją programu Docker.

Pamiętaj ponadto, że jeśli nie zatwierdzisz zmian poleceniem `docker commit`, bo wolisz używać oryginalnego obrazu dystrybucji Linux for PHP, to będziesz musiał wydawać powyższe polecenia przed każdym rozpoczęciem pracy z kodem zawartym w tej książce.

Możemy już przystąpić do testów porównawczych serwera.

Poznajemy Apache Bench (AB)

Istnieje wiele narzędzi do mierzenia wydajności serwera WWW. Do najbardziej znanych należą Apache Bench (AB), Siege, JMeter i Tsung. JMeter (*https://jmeter.apache.org/*) i Tsung

Kup książkę Poleć książkę

(*http://tsung.erlang-projects.org/*) to bardzo interesujące narzędzia do pomiaru obciążenia, warte zbadania przez administratorów przy okazji bardziej zaawansowanych testów. Programistom wystarczy jednak skupić się na programach AB and Siege.

Program AB jest dostarczany wraz z narzędziami programistycznymi serwera WWW i w obrazach dystrybucji Linux for PHP zawierających pliki binarne PHP jest domyślnie zainstalowany. W większości innych dystrybucji systemu Linux program AB znajduje się w oddzielnym pakiecie instalacyjnym narzędzi programistycznych serwera Apache. Warto wiedzieć, że Apache Bench nie wspiera wielowątkowości, co może powodować problemy podczas testów o dużym stopniu współbieżności.

Podczas pomiaru wydajności trzeba unikać często pojawiających się problemów. Przede wszystkim:

- nie należy uruchamiać równocześnie na testowanym komputerze aplikacji pochłaniających dużo zasobów;
- nie należy testować wydajności zdalnych serwerów, ponieważ połączenia sieciowe mogą być głównym źródłem mierzonych opóźnień, zwłaszcza w przypadku testów współbieżnych;
- nie należy testować stron WWW buforowanych za pomocą akceleratorów ruchu sieciowego ani pośredników (ang. *proxies*), bo wyniki będą wypaczone i nie pokażą prawdziwej wydajności serwera;
- nie powinno się sądzić, że pomiar wydajności i testy obciążeniowe będą dokładnie odzwierciedlać interakcję użytkowników z serwerem, bo ich wyniki są tylko orientacyjne;
- trzeba zdawać sobie sprawę, że wyniki testów dotyczą tylko danej architektury sprzętowej i na każdym komputerze będą inne.

Podczas testów będziemy korzystać z przełączników `-k`, `-l`, `-c` i `-n` programu Apache Bench. Oto opis tych parametrów:

- `-k` włącza funkcję KeepAlive (podtrzymywania aktywności), pozwalającą na wykonywanie wielu żądań HTTP w jednej sesji;
- `-l` wyłącza raportowanie błędów w sytuacji, gdy kolejne odpowiedzi mają różny rozmiar;
- `-c` uaktywnia współbieżność, pozwalając wykonywać wiele żądań jednocześnie;
- `-n` określa liczbę żądań wykonywanych w ciągu bieżącej sesji testowej.

W celu uzyskania dalszych informacji na temat opcji narzędzia AB zapoznaj się z odpowiednim wpisem w dokumentacji serwera Apache (*https://httpd.apache.org/docs/2.4/programs/ab.html*).

Zanim uruchomisz testy porównawcze, otwórz nowe okno Terminalu i poleceniem `docker exec` otwórz w kontenerze nową powłokę `bash`. Dzięki temu będziesz mógł sprawdzić zużycie zasobów za pomocą narzędzia `top`. Na początek odczytaj nazwę kontenera. Pojawi się na ona na liście wyników polecenia:

```
# docker ps
```


Kup książkę Poleć książkę

Będziesz mógł wówczas podłączyć się do kontenera i zacząć obserwować zużycie zasobów komendą:

```
# docker exec -it [tutaj nazwa kontenera] /bin/bash
```

W wierszu poleceń kontenera, do którego uzyskałeś dostęp, uruchom polecenie `top`:

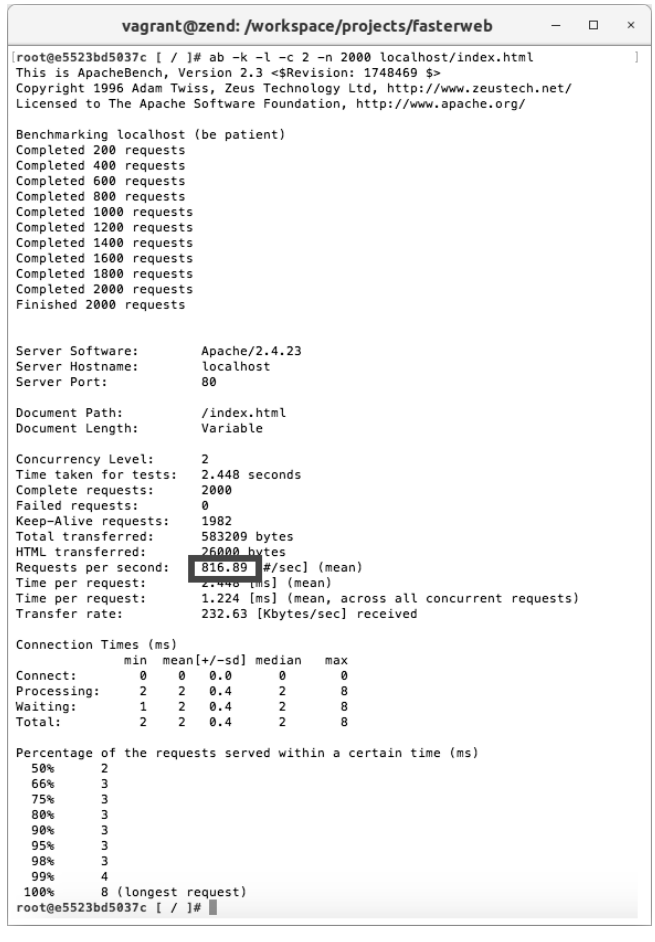
```
# top
```

Teraz w pierwszym oknie Terminalu uruchom test porównawczy:

```
# ab -k -l -c 2 -n 2000 localhost/index.html
```

Po jego wykonaniu uzyskasz raport zawierający informację o średniej liczbie żądań na sekundę, które serwer zdołał obsłużyć (`Requests per second`), przeciętnym czasie odpowiedzi na żądanie (`Time per request`) i rozkładzie czasu odpowiedzi (`Percentage of requests served within a certain time (ms)`).

Twój raport powinien być podobny do tego:

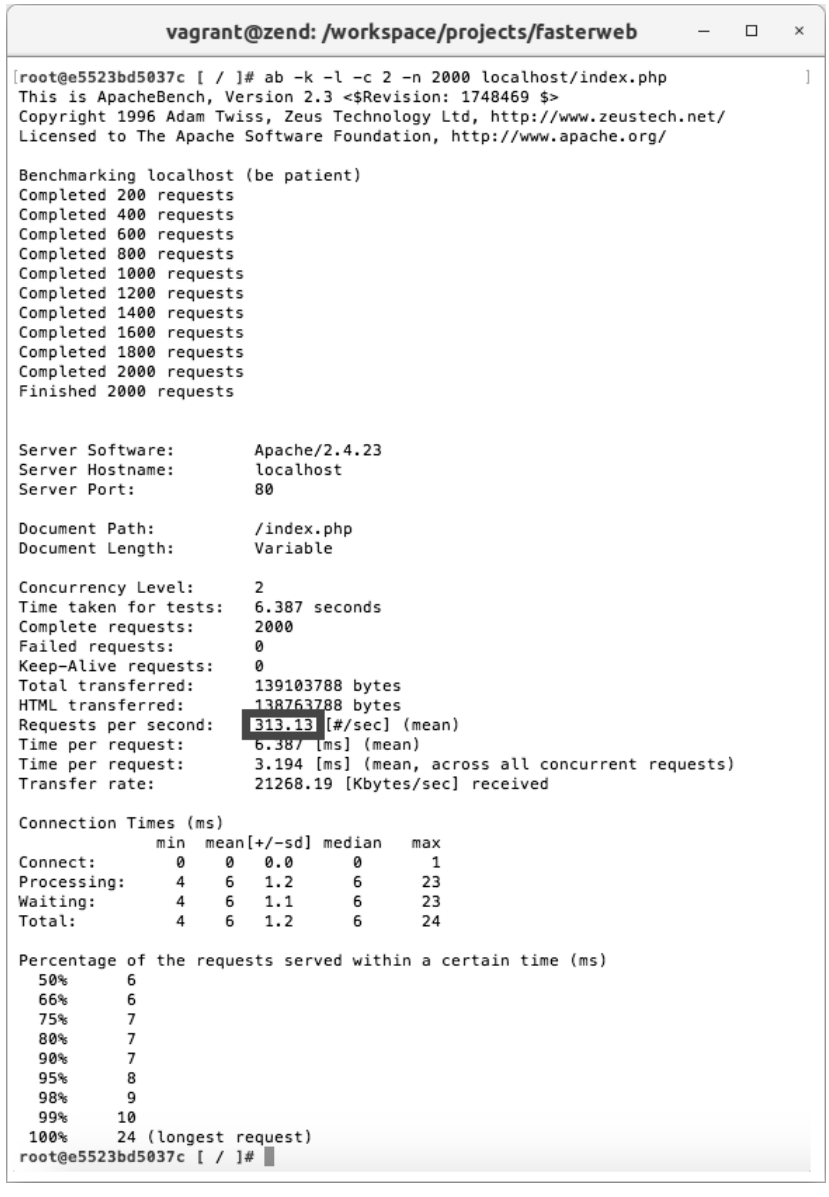


Kup książkę Poleć książkę

Uruchom teraz nowy test porównawczy, wykonując zapytanie o plik `index.php`:

```
# ab -k -l -c 2 -n 2000 localhost/index.php
```

Można zauważyć, że spadła średnia liczba żądań na sekundę, a wzrósł przeciętny czas odpowiedzi i wartość odchylenia standardowego. Na moim komputerze średnia liczba żądań spadła z 800 do około 300, przeciętny czas odpowiedzi wzrósł z 2 do 6 milisekund, a czas dla 100% obsłużonych żądań z 8 do 24 milisekund:



Kup książkę Poleć książkę

Te wyniki dają ogólne pojęcie o granicach wydajności naszego sprzętu i pozwalają określić różne wartości progowe, z którymi będziemy mieli do czynienia próbując zwiększyć wydajność skryptów PHP generujących treści dynamiczne.

Wniknijmy teraz nieco głębiej w wydajność serwera WWW dzięki najlepszemu narzędziu do testowania osiągów i obciążenia — Siege.

Poznajemy Siege

Siege jest narzędziem do testowania obciążenia i wydajności, które pozwala na dalszą analizę osiągów serwera WWW. Zacznijmy od zainstalowania go wewnątrz kontenera Docker.

Z poziomu wiersza poleceń kontenera pobierz i zdekompresuj wersję 4.0.2 programu Siege:

```
# wget -O siege-4.0.2.tar.gz http://download.joedog.org/siege/siege–4.0.2.tar.gz
# tar -xzvf siege-4.0.2.tar.gz
```

Następnie wejdź do katalogu z kodem źródłowym Siege w celu skompilowania i instalacji oprogramowania:

```
# cd siege-4.0.2
# ./configure
# make
# make install
```

Podczas testów z użyciem Siege będziemy korzystać z opcji `-b`, `-c` i `-r`. Oto opis tych parametrów:

- `-b` włącza tryb testu wydajnościowego, co oznacza, że pomiędzy jego powtórzeniami nie ma opóźnień;
- `-c` włącza współbieżność w celu wykonywania jednocześnie wielu żądań;
- `-r` określa liczbę żądań wykonywanych przez każdego z równocześnie symulowanych użytkowników.

Aby uzyskać więcej informacji na temat opcji programu Siege, możesz z wiersza poleceń kontenera wywołać jego podręcznik systemowy:

```
# man siege
```

Rozpocznij test porównawczy Siege:

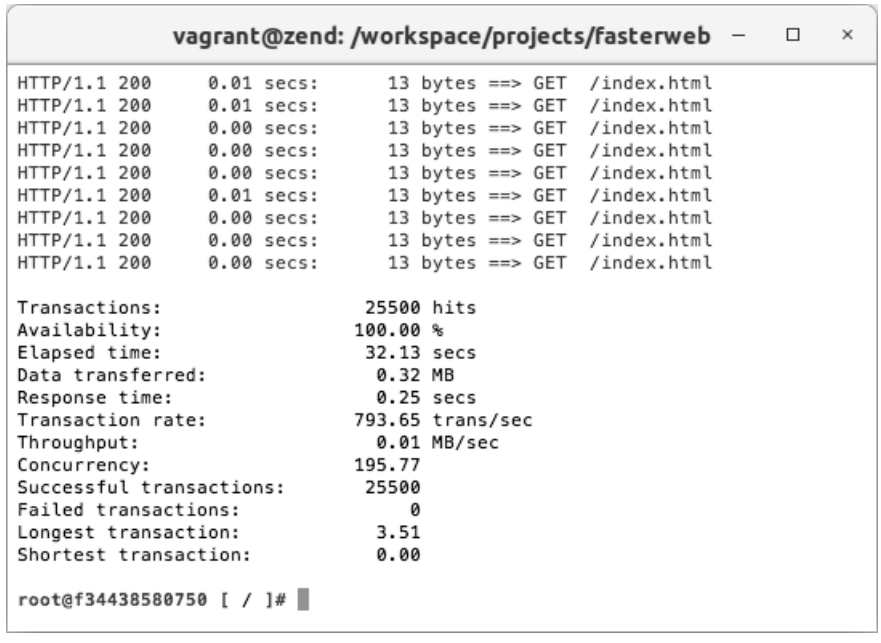
```
# siege -b -c 3000 -r 100 localhost/index.html
```

Uzyskasz po nim raport podobny do przedstawionego na rysunku na następnej stronie.

Jak widać, wyniki odpowiadają rezultatom zwróconym przez narzędzie AB. Nasz test wykazuje, że wskaźnik transakcji wynosi prawie 800 zapytań na sekundę.

Do programu Siege dołączono poręczne narzędzie o nazwie Bombard, pozwalające automatyzować testy i weryfikować skalowalność serwera. Bombard uruchamia Siege z ciągle rosnącą liczbą równocześnie symulowanych użytkowników. Przyjmuje kilka argumentów opcjonalnych.

Kup książkę Poleć książkę



Są to: nazwa pliku z adresami URL używanymi podczas przeprowadzania testów, początkowa liczba równoczesnych klientów, liczba klientów do dodania przy każdym wywołaniu programu Siege, liczba wywołań Siege oraz odstęp w sekundach pomiędzy zapytaniami.

Możemy zatem potwierdzić wyniki poprzednich testów, wydając w kontenerze polecenia:

```
# cd /srv/www
# touch urlfile.txt
# for i in {1..4}; do echo "http://localhost/index.html" >> urlfile.txt ;
done
# bombardment urlfile.txt 10 100 4 0
```

Na koniec powinieneś uzyskać raport podobny do przedstawionego na rysunku na następnej stronie. Wyniki wskazują, że przy ponad 210 użytkownikach transakcje są znacznie wydłużone:

Powtórz próbę, wykonując tym razem zapytanie o plik PHP:

```
# echo "http://localhost/index.php" > urlfile.txt
# for i in {1..3}; do echo "http://localhost/index.php" >> urlfile.txt ;
done
# bombardment urlfile.txt 10 100 4 0
```

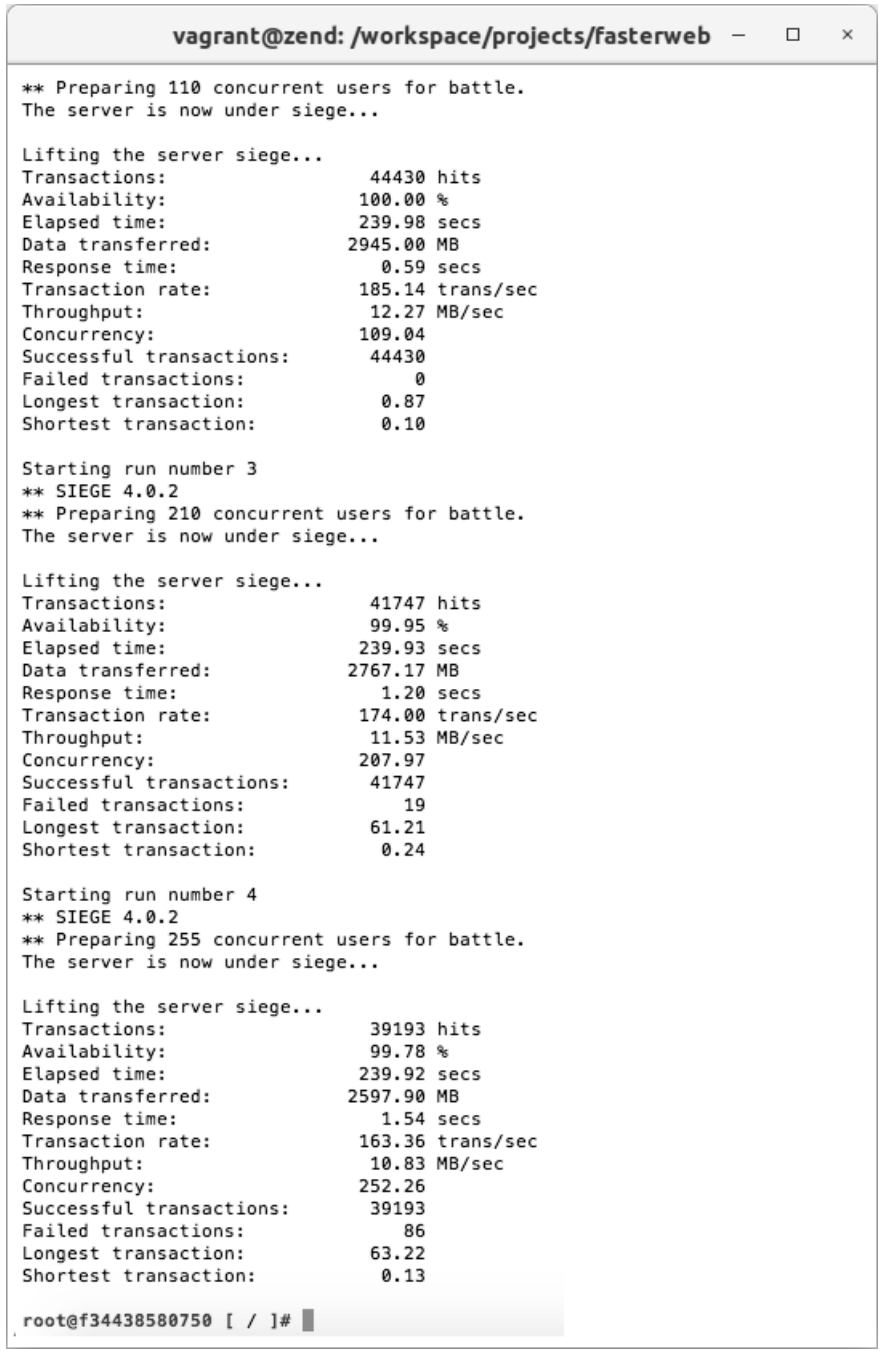
Wyniki tego testu powinny być zbliżone do zaprezentowanych na rysunku na stronie 29. Wydajność serwowania treści dynamicznych jest analogiczna, jak w przypadku zawartości statycznej, ale wskaźnik transakcji jest dużo niższy.

Kup książkę Poleć książkę

vagrant@zend: /workspace/projects/fasterweb

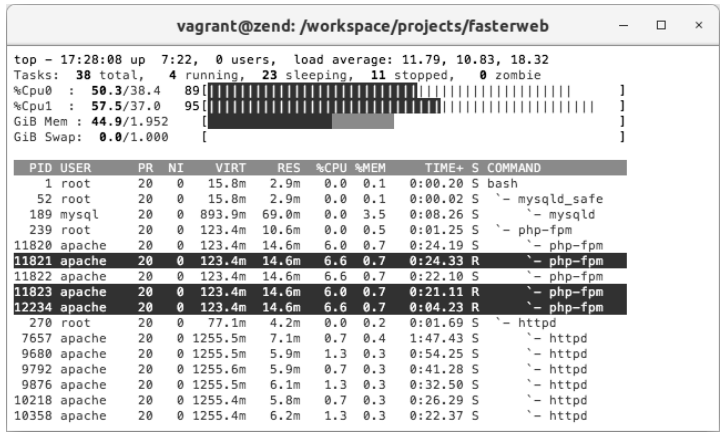
```
** SIEGE 4.0.2
** Preparing 110 concurrent users for battle.
The server is now under siege...

Lifting the server siege...
Transactions:                  362727 hits
Availability:                  100.00 %
Elapsed time:                  300.02 secs
Data transferred:                4.50 MB
Response time:                   0.09 secs
Transaction rate:             1209.01 trans/sec
Throughput:                      0.01 MB/sec
Concurrency:                   104.14
Successful transactions:       362727
Failed transactions:                0
Longest transaction:             0.69
Shortest transaction:            0.00

Starting run number 3
** SIEGE 4.0.2
** Preparing 210 concurrent users for battle.
The server is now under siege...

Lifting the server siege...
Transactions:                  360510 hits
Availability:                  100.00 %
Elapsed time:                  300.03 secs
Data transferred:                4.47 MB
Response time:                   0.17 secs
Transaction rate:             1201.58 trans/sec
Throughput:                      0.01 MB/sec
Concurrency:                   200.54
Successful transactions:       360513
Failed transactions:                0
Longest transaction:             7.36
Shortest transaction:            0.00

Starting run number 4
** SIEGE 4.0.2
** Preparing 255 concurrent users for battle.
The server is now under siege...

Lifting the server siege...
Transactions:                  358487 hits
Availability:                  100.00 %
Elapsed time:                  299.91 secs
Data transferred:                4.44 MB
Response time:                   0.21 secs
Transaction rate:             1195.32 trans/sec
Throughput:                      0.01 MB/sec
Concurrency:                   245.22
Successful transactions:       358487
Failed transactions:                0
Longest transaction:             7.41
Shortest transaction:            0.00
```


Kup książkę Poleć książkę



Komenda `top` uruchomiona w drugim oknie Terminalu pokazuje teraz na moim komputerze użycie obu dostępnych procesorów na poziomie 50% oraz wykorzystanie prawie 50% pamięci RAM (patrz rysunek na następnej stronie).

Kup książkę Poleć książkę



Dowiedzieliśmy się, że jeśli nie ma zbyt wielu równoczesnych zapytań, sprzęt ten zapewnia dobrą wydajność w ograniczonej skali: 800 transakcji na sekundę przy plikach statycznych oraz około 200 transakcji na sekundę na stronach z zawartością generowaną dynamicznie.

Skoro poznaliśmy lepiej bazowe osiągi serwera WWW, oparte wyłącznie na jego zasobach sprzętowych, możemy przystąpić do prawdziwego pomiaru szybkości i wydajności generowania przez niego treści dynamicznych w drodze profilowania. Przejdziemy teraz do instalacji i konfiguracji narzędzi pozwalających profilować i optymalizować kod PHP.

Instalacja i konfiguracja użytecznych narzędzi

Zainstalujemy i skonfigurujemy narzędzia do mierzenia wydajności bazy MySQL oraz profilowania kodu JavaScript. Zacznijmy jednak od instalacji i konfiguracji programu xdebug, debugera i profilera kodu PHP.

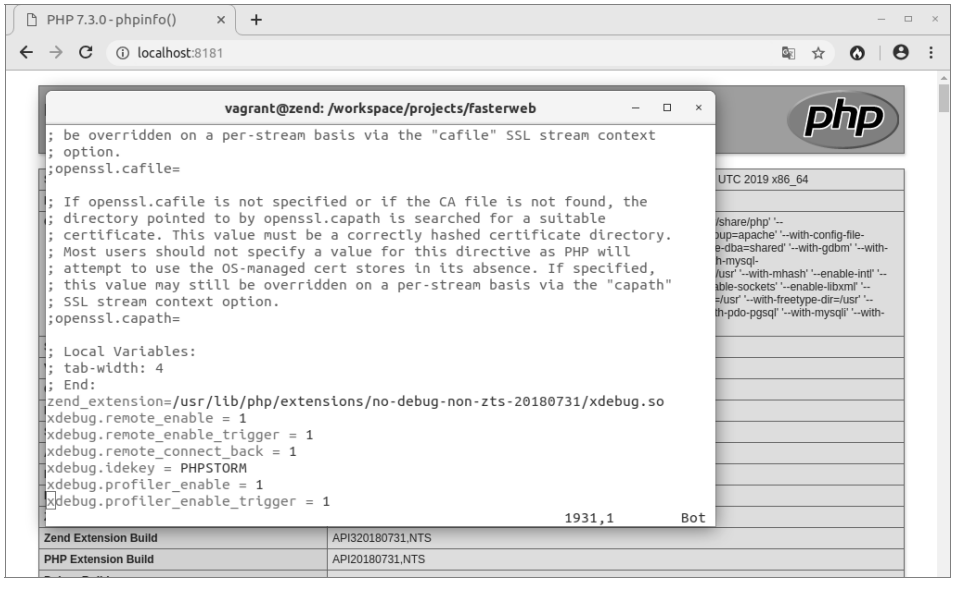
Profilowanie kodu PHP — instalacja i konfiguracja programu xdebug

Jak wspomnieliśmy, zaczniemy od instalacji i konfiguracji xdebug, narzędzia do debugowania i profilowania kodu PHP. To rozszerzenie można pobrać, zdekompresować, skonfigurować, skompilować i zainstalować w bardzo prosty sposób, używając dostarczanego z PHP narzędzia PECL (*https://pecl.php.net/*). W tym celu w oknie Terminalu kontenera należy wprowadzić następujące polecenia:

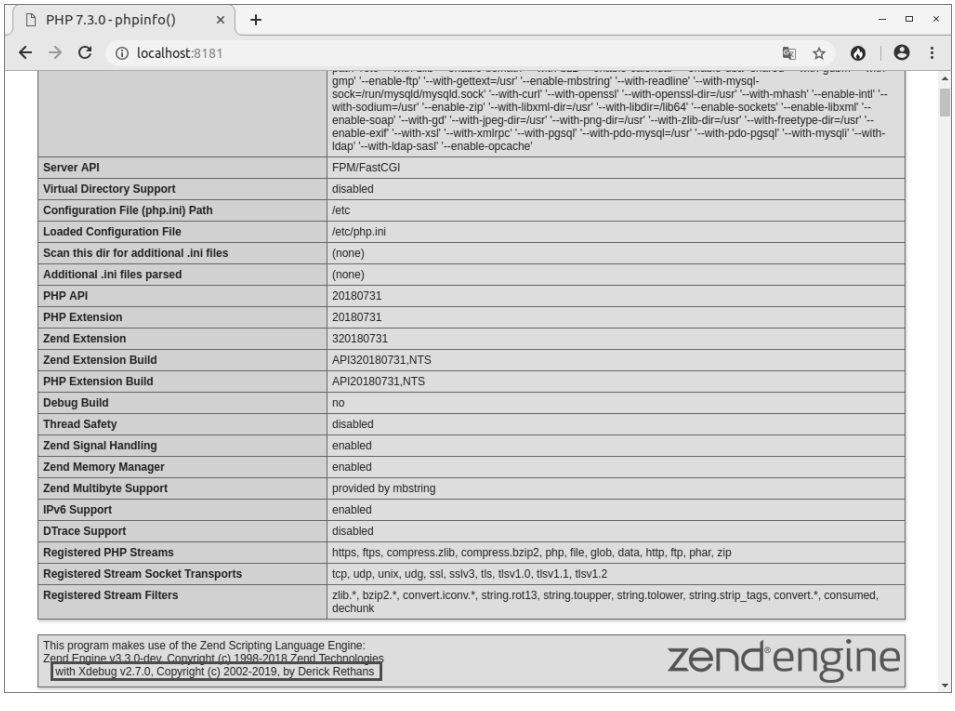
```
# pecl install xdebug
# echo -e "zend_extension=$( php -i | grep extensions | awk '{print $3}'
↪)/xdebug.so\n" >> /etc/php.ini
# echo -e "xdebug.remote_enable = 1\n" >> /etc/php.ini
# echo -e "xdebug.remote_enable_trigger = 1\n" >> /etc/php.ini
# echo -e "xdebug.remote_connect_back = 1\n" >> /etc/php.ini
# echo -e "xdebug.idekey = PHPSTORM\n" >> /etc/php.ini
# echo -e "xdebug.profiler_enable = 1\n" >> /etc/php.ini
# echo -e "xdebug.profiler_enable_trigger = 1\n" >> /etc/php.ini
# /etc/init.d/php-fpm restart
# tail -50 /etc/php.ini
```


Kup książkę Poleć książkę

Ostatnie wiersze pliku /etc/php.ini w Twoim kontenerze powinny teraz wyglądać następująco:

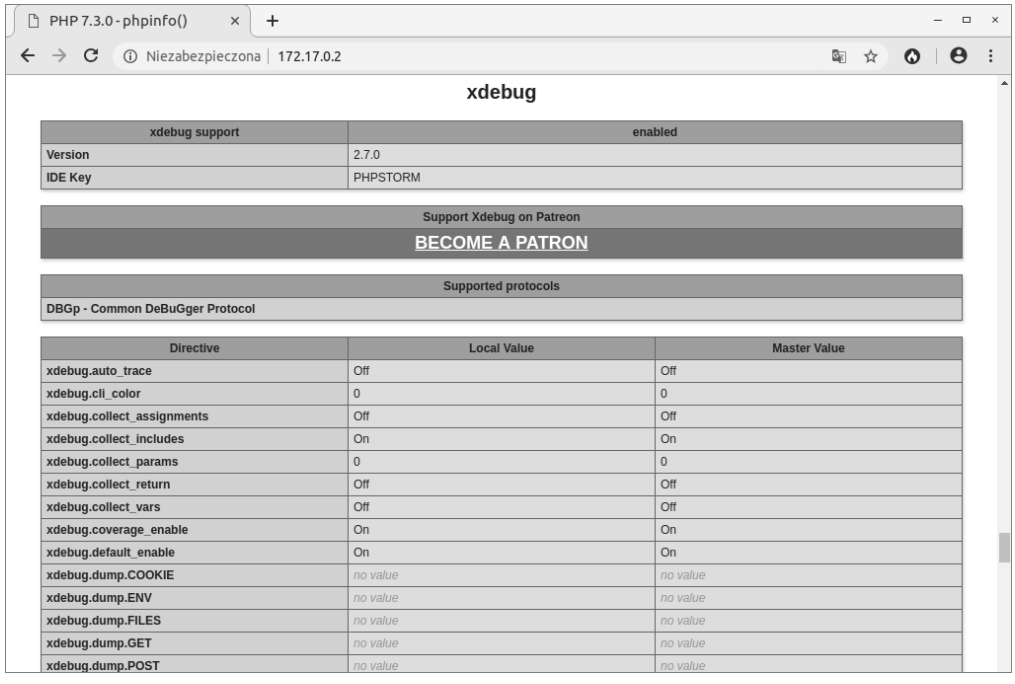


Na koniec odśwież w swojej ulubionej przeglądarce stronę *http://localhost:8181*. Powinna teraz zawierać informację o załadowaniu rozszerzenia xdebug i przedstawiać się tak:



Kup książkę Poleć książkę

Jeśli przewiniesz stronę do dołu, zobaczysz sekcję *xdebug*:



xdebug

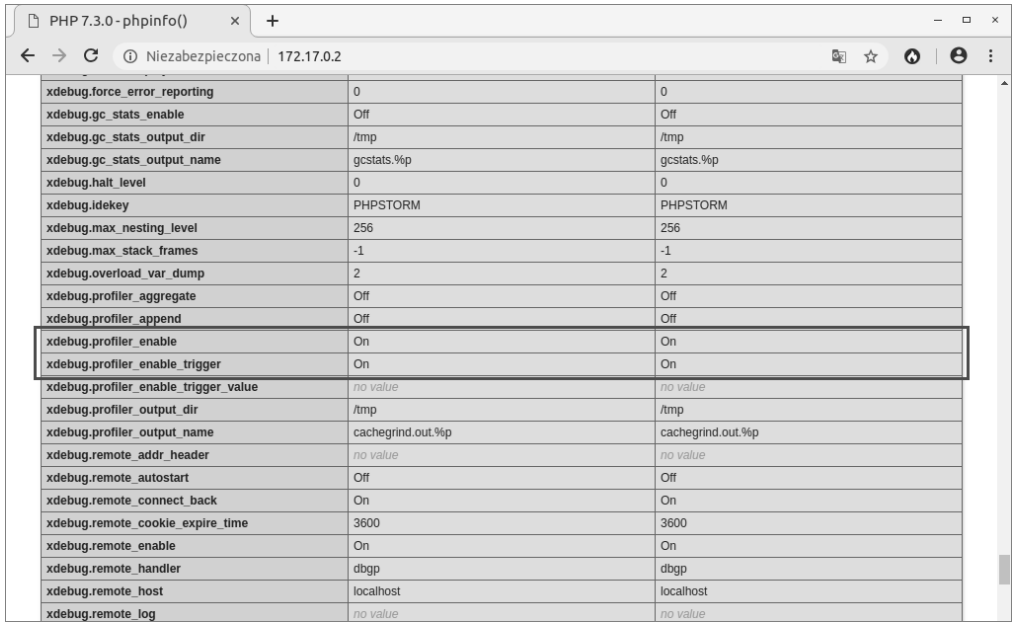
xdebug support	enabled
Version	2.7.0
IDE Key	PHPSTORM

Support Xdebug on Patreon
BECOME A PATRON

Supported protocols
DBGp - Common DeBuGger Protocol

Directive	Local Value	Master Value
xdebug.auto_trace	Off	Off
xdebug.cli_color	0	0
xdebug.collect_assignments	Off	Off
xdebug.collect_includes	On	On
xdebug.collect_params	0	0
xdebug.collect_return	Off	Off
xdebug.collect_vars	Off	Off
xdebug.coverage_enable	On	On
xdebug.default_enable	On	On
xdebug.dump.COOKIE	*no value*	*no value*
xdebug.dump.ENV	*no value*	*no value*
xdebug.dump.FILES	*no value*	*no value*
xdebug.dump.GET	*no value*	*no value*
xdebug.dump.POST	*no value*	*no value*

Można też zauważyć, że są aktywne opcje profilowania:



xdebug.force_error_reporting	0	0
xdebug.gc_stats_enable	Off	Off
xdebug.gc_stats_output_dir	/tmp	/tmp
xdebug.gc_stats_output_name	gcstats.%p	gcstats.%p
xdebug.halt_level	0	0
xdebug.idekey	PHPSTORM	PHPSTORM
xdebug.max_nesting_level	256	256
xdebug.max_stack_frames	-1	-1
xdebug.overload_var_dump	2	2
xdebug.profiler_aggregate	Off	Off
xdebug.profiler_append	Off	Off
xdebug.profiler_enable	On	On
xdebug.profiler_enable_trigger	On	On
xdebug.profiler_enable_trigger_value	*no value*	*no value*
xdebug.profiler_output_dir	/tmp	/tmp
xdebug.profiler_output_name	cachegrind.out.%p	cachegrind.out.%p
xdebug.remote_addr_header	*no value*	*no value*
xdebug.remote_autostart	Off	Off
xdebug.remote_connect_back	On	On
xdebug.remote_cookie_expire_time	3600	3600
xdebug.remote_enable	On	On
xdebug.remote_handler	dbgp	dbgp
xdebug.remote_host	localhost	localhost
xdebug.remote_log	*no value*	*no value*

Kup książkę Poleć książkę

Skonfigurujemy teraz program PHPStorm w roli serwera debugowania. Pozwoli to używać tego środowiska IDE jako centrum sterowania sesjami debugera.

Zanim zaczniemy, udostępnij cały folder *fasterweb* jako katalog główny serwera WWW, wprowadzając w kontenerze następujące polecenia:

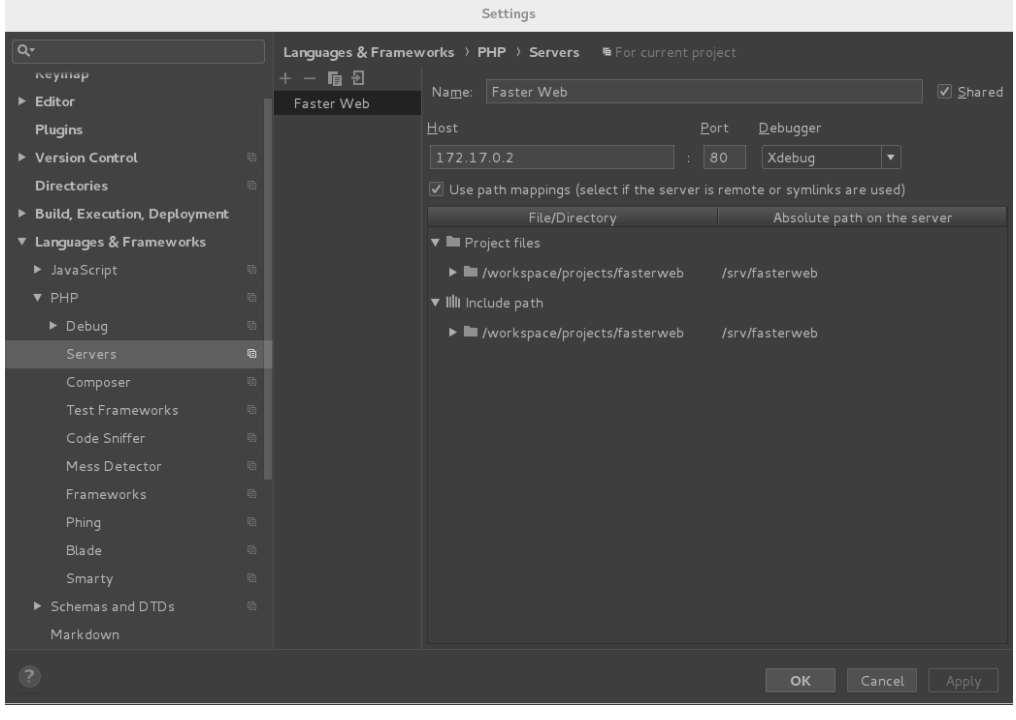
```
# rm /srv/www
# ln -s /srv/fasterweb /srv/www
# cd /srv/www
```

Uruchom teraz program *PHPStorm* i ustaw jako katalog główny projektu folder *fasterweb*. W tym celu wybierz polecenie *Create New Project from Existing Files* (utwórz nowy projekt z istniejących plików), zaznacz *Source files are in a local directory* (pliki źródłowe znajdują się w lokalnym katalogu), jako *Project root* (katalog główny projektu) wybierz folder *fasterweb*, a następnie kliknij przycisk *Finish*.

Po utworzeniu projektu wybierz z menu *File* (plik) polecenie *Settings* (ustawienia). W sekcji *Languages & Frameworks* (języki programowania i frameworki) rozwiń pozycję *PHP* i kliknij *Servers* (serwery). Wprowadź odpowiednie informacje zgodnie ze specyfiką własnych ustawień. Opcja *Host* ma zawierać adres IP kontenera Linux for PHP. Jeśli nie jesteś pewny co do tego adresu, w wierszu poleceń kontenera wprowadź następującą komendę:

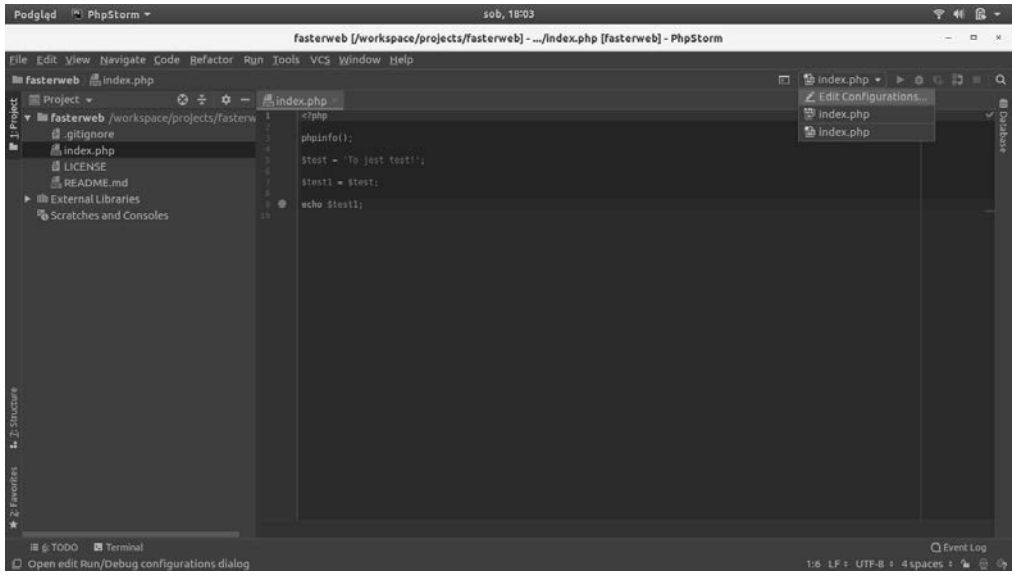
```
# ifconfig
```

Na koniec zatwierdź ustawienia, klikając przyciski *Apply* (zastosuj) i *OK*:

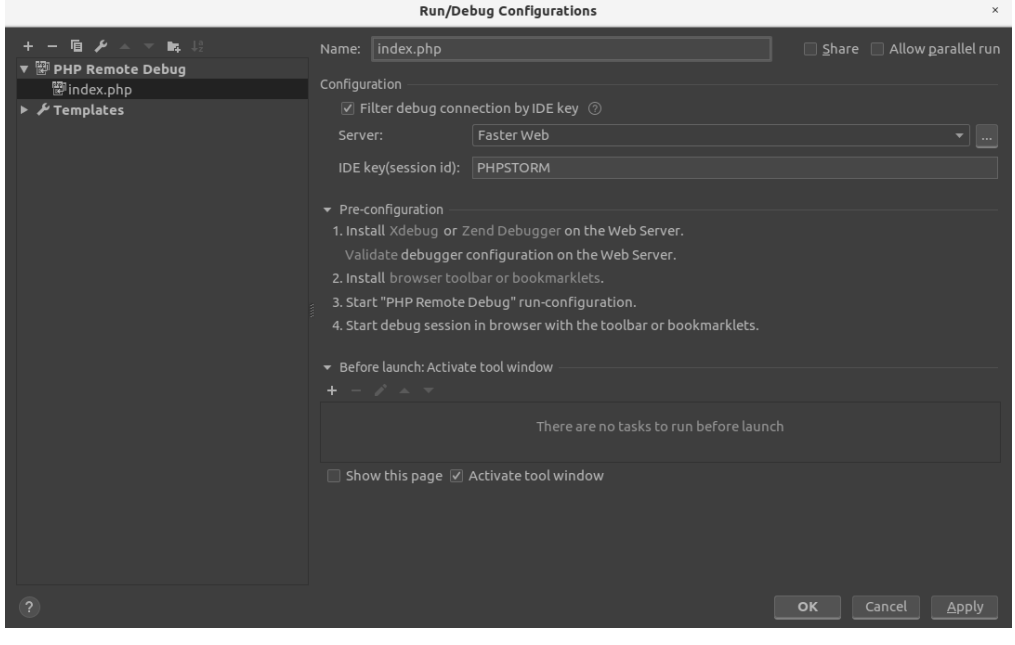


Kup książkę Poleć książkę

Teraz w menu *Run* (uruchom) poszukaj pozycji *Edit Configurations…* (edycja konfiguracji). Polecenie to można również znaleźć po prawej stronie okna środowiska IDE:

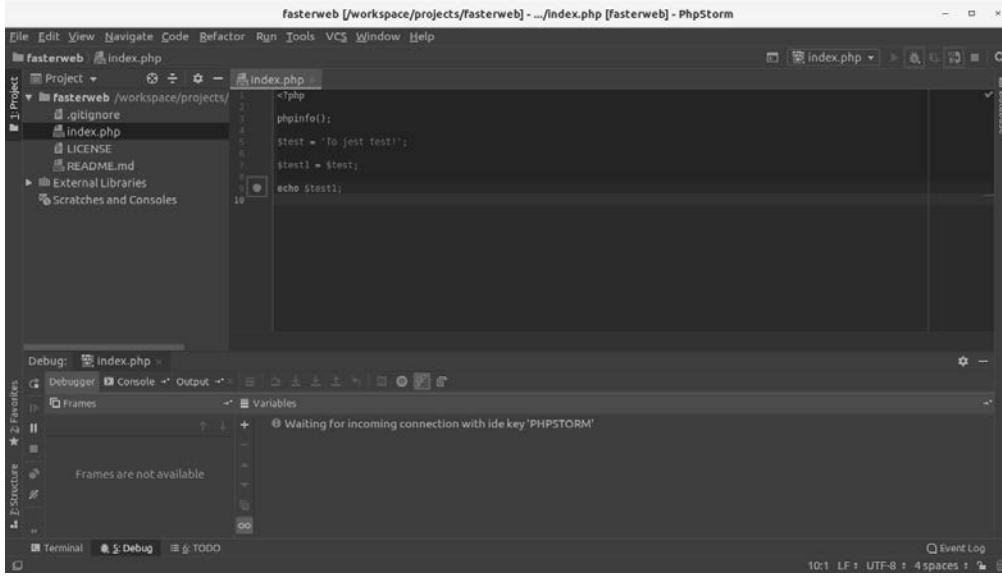


Klikając zielony znak plus w lewym górnym narożniku okna możesz dodać pozycję *PHP Remote Debug* (zdalne debugowanie aplikacji PHP). Wybierz serwer, który dodałeś w poprzednim kroku i upewnij się, czy właściwość *Ide key(session id)* [klucz środowiska IDE(identyfikator sesji)] ma wartość *PHPSTORM*:



Kup książkę Poleć książkę

Możesz już uaktywnić serwer debugowania środowiska PHPStorm, klikając w menu u góry po prawej stronie przycisk *Listen to debugger connections* (nasłuchuj połączeń z debugera), ustawić punkt przerwania, klikając wolne miejsce po prawej stronie dowolnego numeru wiersza pliku *index.php* oraz uruchomić narzędzie debugowania, korzystające z utworzonej w poprzednim kroku konfiguracji:



Jeśli w górnym prawym narożniku ekranu brakuje paska narzędzi, należy w menu *View* (widok) zaznaczyć pozycję *Toolbar* (pasek narzędziowy), a na ekranie pokażą się przyciski. Mają one też odpowiedniki w poleceniach menu *Run*.

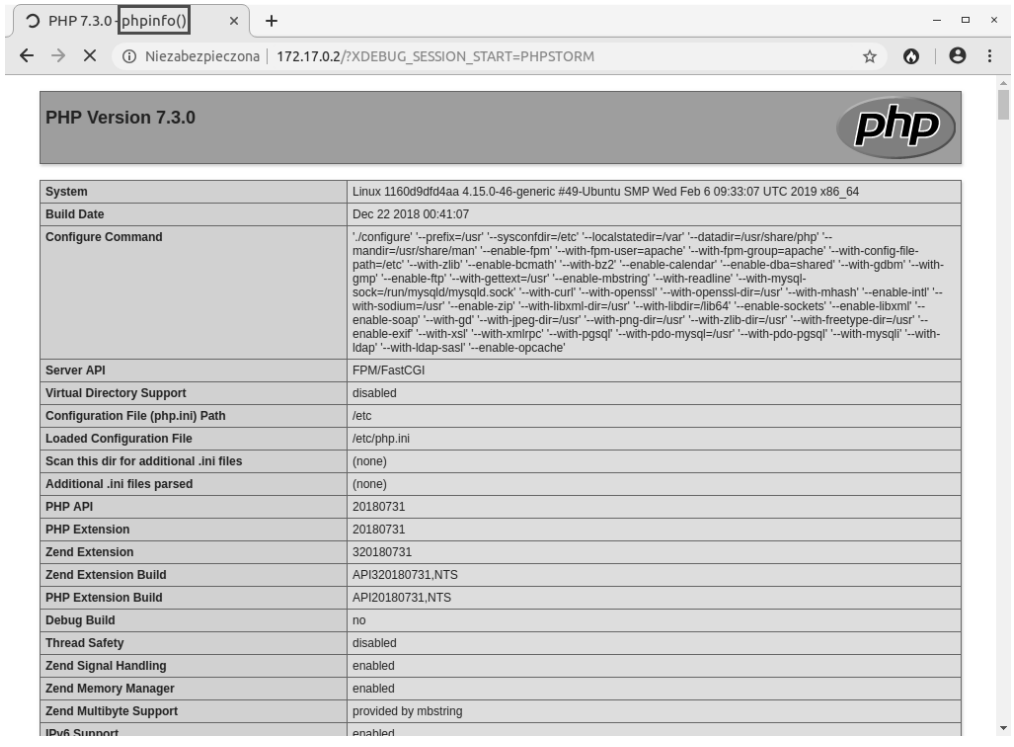
Otwórz teraz swoją ulubioną przeglądarkę i wczytaj tę samą stronę WWW, wprowadzając adres IP kontenera Docker: *http:///[IP_ADDRESS]/?XDEBUG_SESSION_START=PHPSTORM*.

Można dostrzec, że przeglądarka wpadła w nieskończoną pętlę, oczekując na wznowienie lub zakończenie sesji debugowania (patrz pierwszy rysunek na następnej stronie).

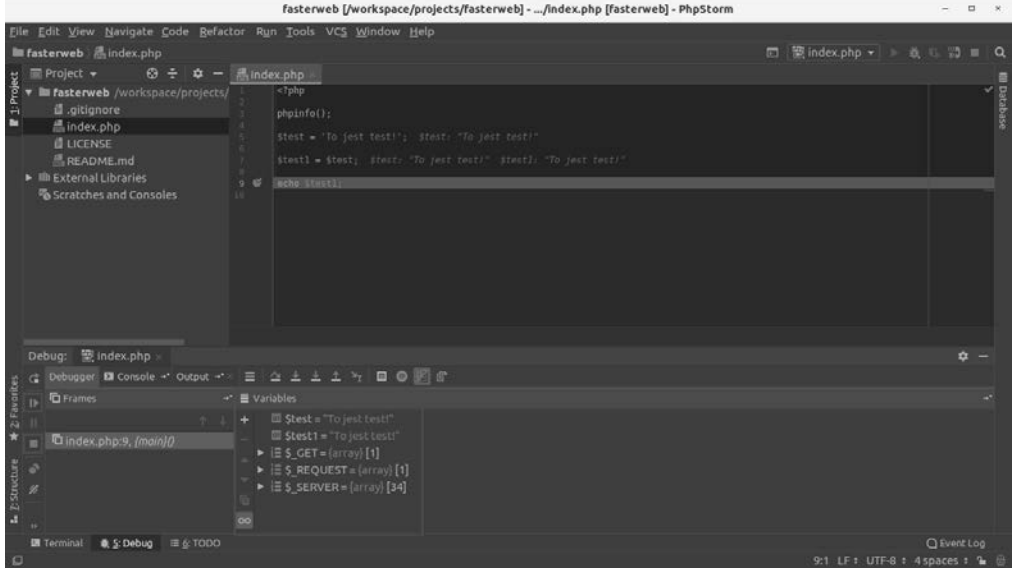
Zapewne zauważyłeś też, że w środowisku IDE pokazały się informacje dotyczące debugowania. Można stamtąd również kontrolować sesję i wznawiać wykonanie kodu.

Zanim wznowisz wykonywanie aplikacji, wciskając w tym celu znajdujący się po lewej stronie ekranu zielony przycisk odtwarzania, sprawdź dokładnie zawartość zmiennych. Aby zakończyć sesję debugowania, wystarczy kliknąć znajdujący się w tej samej grupie ikon różowy przycisk zatrzymania (patrz drugi rysunek na następnej stronie).

Kup książkę Poleć książkę



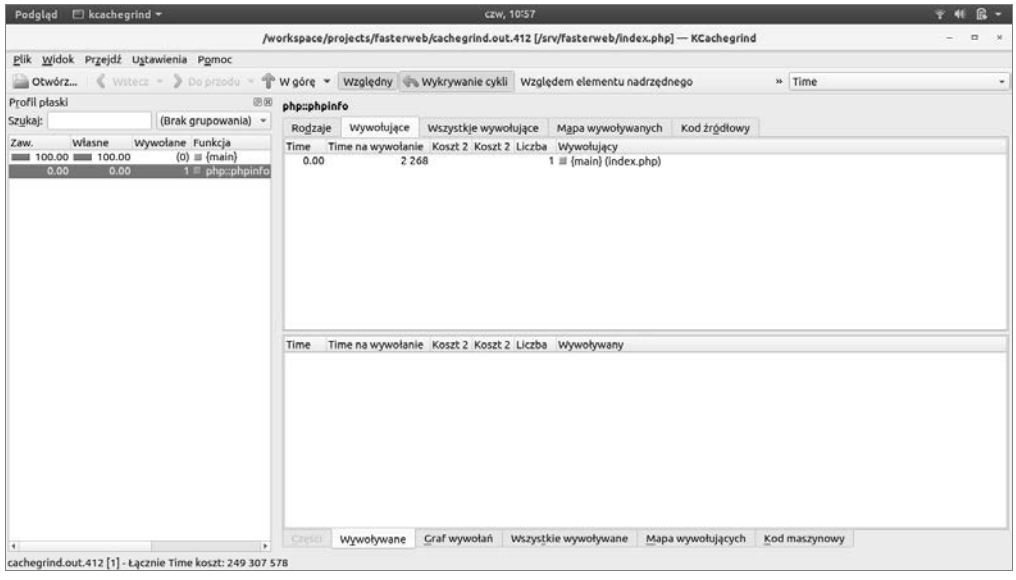
System	Linux 1160d9dfd4aa 4.15.0-46-generic #49-Ubuntu SMP Wed Feb 6 09:33:07 UTC 2019 x86_64
Build Date	Dec 22 2018 00:41:07
Configure Command	'./configure' '--prefix=/usr' '--sysconfdir=/etc' '--localstatedir=/var' '--datadir=/usr/share/php' '--mandir=/usr/share/man' '--enable-fpm' '--with-fpm-user=apache' '--with-fpm-group=apache' '--with-config-file-path=/etc' '--with-zlib' '--enable-bcmath' '--with-bz2' '--enable-calendar' '--enable-dba=shared' '--with-gdbm' '--with-gmp' '--enable-ftp' '--with-gettext=/usr' '--enable-mbstring' '--with-readline' '--with-mysql-sock=/run/mysqld/mysqld.sock' '--with-curl' '--with-openssl' '--with-openssl-dir=/usr' '--with-mhash' '--enable-intl' '--with-sodium=/usr' '--enable-zip' '--with-libxml-dir=/usr' '--with-libdir=/lib64' '--enable-sockets' '--enable-libxml' '--enable-soap' '--with-gd' '--with-jpeg-dir=/usr' '--with-png-dir=/usr' '--with-zlib-dir=/usr' '--with-freetype-dir=/usr' '--enable-exif' '--with-xsl' '--with-xmlrpc' '--with-pgsql' '--with-pdo-mysql=/usr' '--with-pdo-pgsql' '--with-mysqli' '--with-ldap' '--with-ldap-sasl' '--enable-opcache'
Server API	FPM/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc
Loaded Configuration File	/etc/php.ini
Scan this dir for additional .ini files	(none)
Additional .ini files parsed	(none)
PHP API	20180731
PHP Extension	20180731
Zend Extension	320180731
Zend Extension Build	API320180731,NTS
PHP Extension Build	API20180731,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	enabled
Zend Memory Manager	enabled
Zend Multibyte Support	provided by mbstring
IPv6 Support	enabled



Po zakończeniu sesji debugowania możesz zajrzeć do katalogu */tmp* kontenera. W pliku o nazwie *cachegrind.out* powinny być dane wyjściowe profilera. Możesz bezpośrednio przejrzeć ten plik w edytorze tekstu albo posłużyć się wyspecjalizowanym oprogramowaniem, takim jak

Kup książkę Poleć książkę

Kcachegrind, które zainstalujesz przy pomocy menedżera pakietów swojej dystrybucji systemu Linux. Oto przykładowe dane wyjściowe oglądane za pomocą programu Kcachegrind:



Oprócz tych programów, których będziemy używać do optymalizacji przykładowych kodów w następnych rozdziałach, zawsze możesz dodatkowo korzystać z narzędzia do profilowania *xdebug*. A w następnym rozdziale przyjrzymy się właśnie bardziej zaawansowanym programom profilującym, między innymi **Blackfire.io**.

Jeśli skończyłeś testowanie narzędzia *xdebug*, możesz z powrotem ustawić jako katalog główny serwera WWW folder *r01*:

```
# rm /srv/www
# ln -s /srv/fasterweb/r01 /srv/www
# cd /srv/www
```

Przyjrzyjmy się teraz narzędziom do testowania szybkości bazy SQL.

Testowanie szybkości bazy SQL

Chociaż serwer PostgreSQL bywa często uważany za najszybszy na świecie — zaraz po bazie danych Oracle — system zarządzania relacyjnymi bazami danych, to **MariaDB** (odgałęzienie serwera MySQL) wciąż należy do najszybszych i najpopularniejszych systemów bazodanowych, zwłaszcza w przypadku stosowania prostych zapytań SQL. Omawiając w tej książce optymalizacje języka SQL, będę się więc głownie odnosił do bazy MariaDB.

Do sprawdzania wydajności serwera MariaDB będziemy używać narzędzia *mysqlslap*, dostarczanego z serwerem MySQL od wersji 5.1.4. Aby rozpocząć testy, załadujemy najpierw demonstracyjną bazę danych **Sakila**. W wierszu poleceń kontenera wprowadź następujące komendy:

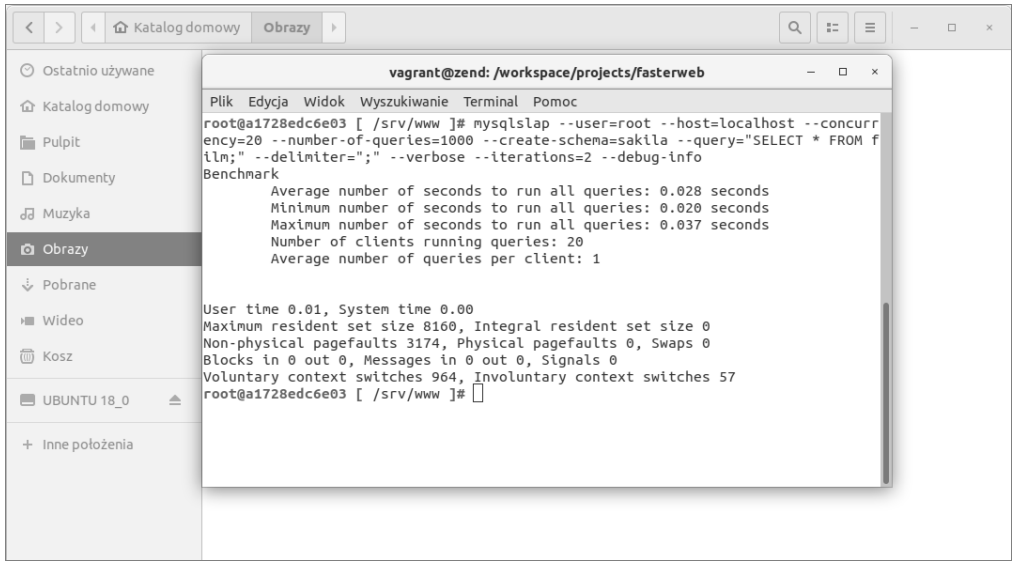
Kup książkę Poleć książkę

```
# wget -O sakila-db.tar.gz \
> https://downloads.mysql.com/docs/sakila-db.tar.gz
# tar -xzvf sakila-db.tar.gz
# mysql -uroot < sakila-db/sakila-schema.sql
# mysql -uroot < sakila-db/sakila-data.sql
```

Po załadowaniu bazy danych możesz uruchomić pierwszy test porównawczy:

```
# mysqlslap --user=root --host=localhost --concurrency=20 --number-of-queries=1000
↪--create-schema=sakila --query="SELECT * FROM film;" --delimiter=";" --verbose
↪--iterations=2 --debug-info
```

Uzyskane wyniki powinny być podobne do tych:



Testy możesz powtórzyć przy wyższym poziomie współbieżności celem porównania wyników:

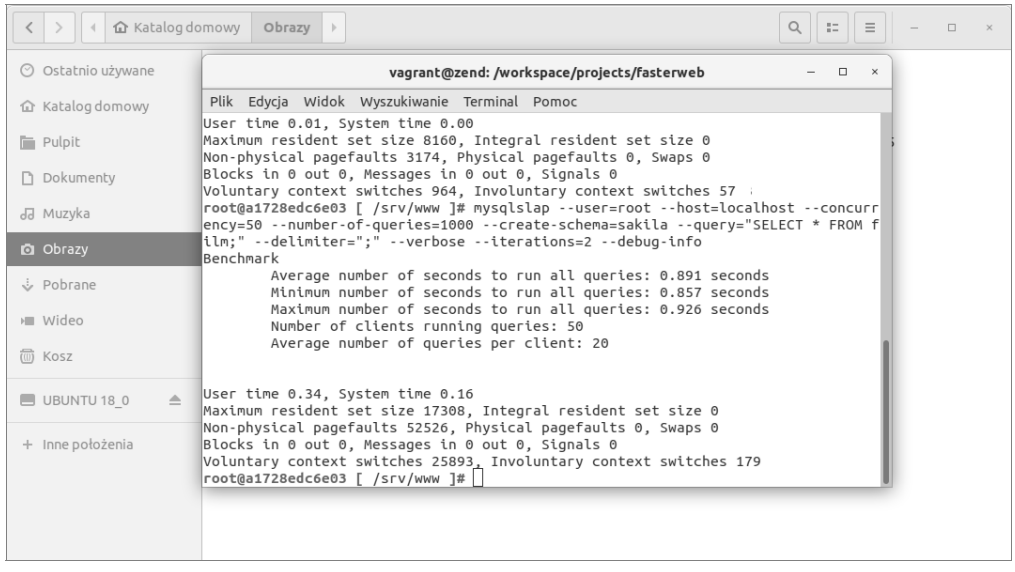
```
# mysqlslap --user=root --host=localhost --concurrency=50 --number-of-queries=
↪1000 --create-schema=sakila --query="SELECT * FROM film;" --delimiter=";"
↪--verbose --iterations=2 --debug-info
```

Oto wyniki drugiego testu (patrz pierwszy rysunek na następnej stronie).

Wyniki testów wskazują, że przy zapytaniu obejmującym całą tabelę zawierającą około 1000 pozycji wydajność drastycznie się zmniejsza, jeśli do serwera trafia ponad 50 jednoczesnych zapytań.

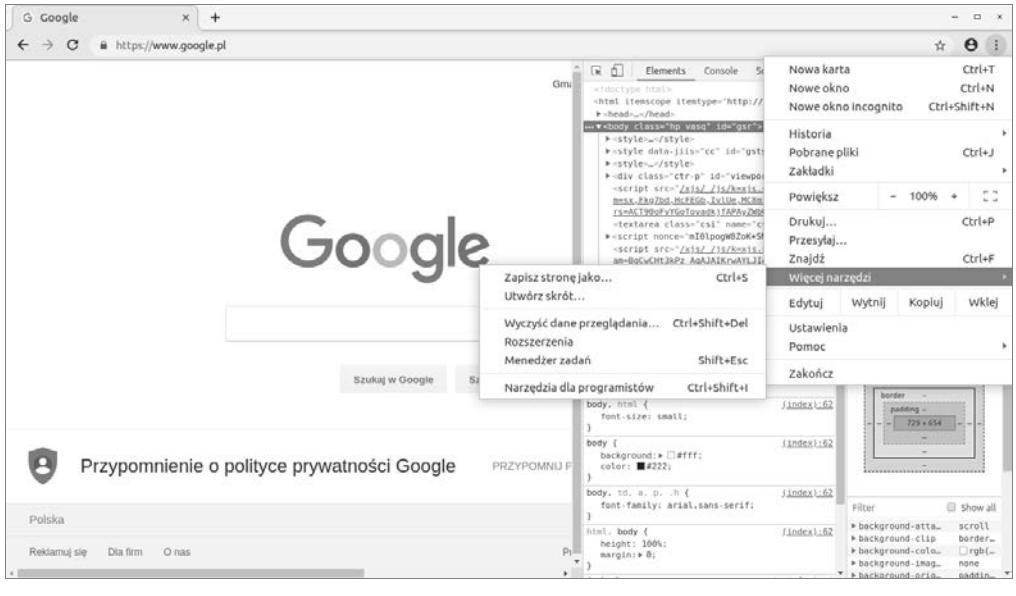
O użyteczności tego typu testów oraz wielu innych, bardziej zaawansowanych narzędzi, przekonasz się podczas omawiania optymalizacji zapytań SQL w rozdziałach poświęconych temu zagadnieniu.

Kup książkę Poleć książkę



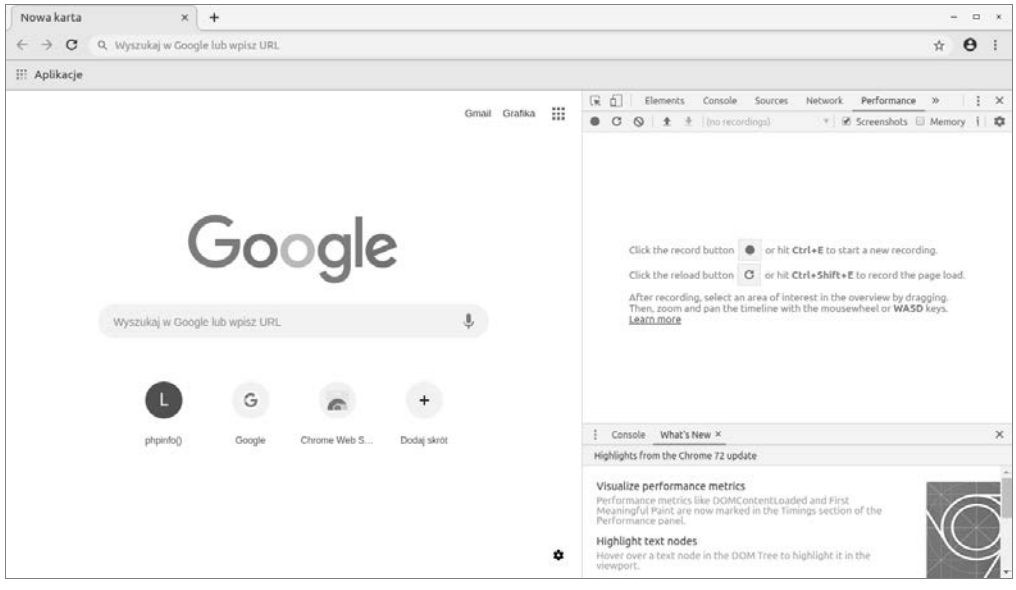
Narzędzia dla programistów JavaScript

W celu zmierzenia wydajności i profilowania kodu JavaScript zamieszczonego w tej książce będziemy korzystać z narzędzi dla programistów, które są zintegrowane z przeglądarką Google Chrome. Chrome zapewnia rejestrację zdarzeń w linii czasu oraz śledzenie czasu wykonywania operacji, co pozwala identyfikować „wąskie gardła" w kodzie JavaScript. Aby uaktywnić te narzędzia, kliknij symbol trzech kropek w prawym górnym narożniku przeglądarki i z podmenu *Więcej narzędzi* wybierz pokazaną poniżej opcję *Narzędzia dla programistów*:



Kup książkę Poleć książkę

Użycie profilera sprowadza się do kliknięcia przycisku nagrywania i odświeżenia profilowanej strony. Można potem przeanalizować wyniki w celu identyfikacji potencjalnych problemów z kodem:



W rozdziałach 7., „Język JavaScript i programowanie sterowane zagrożeniami" oraz 8., „Funkcyjny JavaScript", w większym stopniu wykorzystamy to narzędzie do mierzenia wydajności i ogólnej optymalizacji kodu JavaScript.

Podsumowanie

W tym rozdziale zdefiniowałem pojęcie Szybszej Sieci, pokazałem, dlaczego jest ważne i jak odróżnić je od czystej wydajności. Opisałem, jak w celu pomiaru szybkości zainstalować i skonfigurować narzędzia do testów porównawczych i profilowania, a także pokazałem, jak z nich korzystać.

W następnym rozdziale odkryjesz automatyczne profilowanie za pomocą narzędzia Blackfire.io. Nauczysz się też monitorowania, instalując i konfigurując zestaw programów TICK Stack oraz Grafana na fikcyjnym serwerze produkcyjnym, wdrożonym na drugim kontenerze Docker.

Kup książkę Poleć książkę

Bibliografia

[1] *https://googleblog.blogspot.ca/2009/06/lets-make-web-faster.html*

[2] Bondi André B., *Foundations of Software and System Performance Engineering: Process, Performance Modeling, Requirements, Testing, Scalability, and Practice*, Addison-Wesley, Upper Saddle River 2015.

[3] Meier J. D. et al., *Performance Testing Guidance for Web Applications*, Microsoft Corporation, Redmond 2007.

[4] *https://www.smashingmagazine.com/2015/11/why-performance-matters-part-2-perception-management/*

[5] *https://speakerd.s3.amazonaws.com/presentations/2ece664392024e9da39ea82e3d9f1139/perception-performance-ux-confoo-3-4.pdf*

Kup książkę Poleć książkę

Kup książkę Poleć książkę

Skorowidz

Kup książkę Poleć książkę

G

H

I

J

K

L

M

Kup książkę Poleć książkę

N

O

P

R

S

Kup książkę Poleć książkę

Ś

T

U

W

Z

Kup książkę Poleć książkę


PROGRAM
PARTNERSKI
GRUPY HELION
1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄŻKI
3. ZBIERAJ PROWIZJĘ
Zmień swoją stronę WWW w działający bankomat!
Dowiedz się więcej i dołącz już dzisiaj!
http://program-partnerski.helion.pl
GRUPA
Helion

Przekraczaj granice wydajności. Projektuj szybszy kod!

Termin *Szybsza Sieć* (ang. *Faster Web*) powstał stosunkowo niedawno. Koncepcja ta skupia się na rozwijaniu wszystkich elementów technologii internetowych oraz interfejsu użytkownika, tak aby przyśpieszyć interakcję między klientem a serwerem oraz działanie samej aplikacji. Efektem tych optymalizacji powinna być poprawa jakości pracy systemu. Aby w pełni przyswoić ideę Szybszej Sieci, trzeba dobrze zrozumieć takie pojęcia jak wydajność, efektywność i postrzegana wydajność. Konieczne jest również opanowanie większości nowych technologii webowych.

Dzięki tej książce dowiesz się, w jaki sposób poprawić działanie każdej aplikacji WWW, aby odpowiadała kryteriom Szybszej Sieci. Wkrótce rozpoczniesz pracę z najnowszymi narzędziami do mierzenia wydajności, profilowania i monitorowania dla PHP, MySQL i JavaScriptu. Omówiono tu wszystkie istotne zagadnienia związane z ideą Szybszej Sieci, w tym optymalizację kodu PHP 7, programowanie asynchroniczne i programowanie funkcyjne w JavaScripcie. Sporo miejsca poświęcono najlepszym strategiom optymalizacji. Opisano też techniki profilowania skryptów, pomiaru wydajności bazy danych, optymalizacji zapytań SQL oraz podnoszenia wydajności serwera WWW. Przekonasz się, że dzięki optymalizacji można przekraczać granice efektywności aplikacji!

W tej książce między innymi:

- wprowadzenie do koncepcji Szybszej Sieci i jej formalne aspekty
- monitorowanie i profilowanie w środowisku ciągłej integracji
- potencjał struktur danych PHP 7
- optymalizacja baz danych i nowoczesne techniki języka SQL
- najnowsze osiągnięcia JavaScriptu
- wpływ odpowiedniego projektu interfejsu użytkownika na wydajność aplikacji

Andrew Caya — od 30 lat zajmuje się programowaniem, specjalizuje się w PHP, kodował też w C, C++, Perlu. Otrzymał tytuły Zend Certified PHP Engineer i Zend Certified Architect. Jest twórcą dystrybucji Linux for PHP i głównym programistą popularnego rozszerzenia do systemu Joomla! Obecnie jest prezesem w założonej przez siebie firmie Foreach Code Factory oraz autorem i recenzentem technicznym w wydawnictwie Packt Publishing.

Helion
helion.pl
0 801 339900
0 601 339900
INFORMATYKA W NAJLEPSZYM WYDANIU

Sprawdź nasze szkolenia!
SZKOLENIA
AKADEMIA IT & BUSINESS
WWW.SZKOLENIA.HELION.PL

KOD KORZYŚCI
Sięgnij po więcej!

ISBN 978-83-283-5521-7
9 788328 355217
Cena: 44,90 zł

Packt