

Program szkolenia:

Projektowanie i implementacja wysokowydajnych aplikacji w języku PHP

Informacje:

Nazwa:	Projektowanie i implementacja wysokowydajnych aplikacji w języku PHP
Kod:	PHP-performance
Kategoria:	PHP
Grupa docelowa:	developerzy
Czas trwania:	3 dni
Forma:	60% wykłady / 40% warsztaty

Szkolenie zostało przygotowane z myślą o programistach i architekach PHP, przed którymi zostało postawione zadanie zaprojektowania i implementacji wysokowydajnej aplikacji webowej.

Szkolenie zawiera szereg praktycznych rozwiązań opracowanych na podstawie doświadczeń w dojrzałych projektach, obsługujących duży wolumen ruchu i działających w oparciu o rozbudowane infrastruktury serwerowe.

Zalety szkolenia:

- Kruczki, które dadzą Ci realną przewagę
- Dostęp do wiedzy eksperckiej architekta systemów dużej skali (największy polski portal społecznościowy jak i największy komunikator)

Szczegółowy program:

1. Wprowadzenie

1.1. Dlaczego wydajność aplikacji ma znaczenie?

1.2. Flow obsługi requestu przez PHP

1.3. Wydajność a skalowalność, mity i fakty

2. Profilowanie i pomiary wydajności aplikacji

2.1. Metody i narzędzia przeprowadzanie testów obciążeniowych

2.1.1. ApacheBenchmark

2.1.2. Tsung

2.1.3. Selenium

2.1.4. JMeter

2.2. Monitorowanie aplikacji i serwera

2.2.1. NewRelic

2.2.2. Graphite / StatsD / Diamond

2.2.3. xhprof

2.3. Pułapki mikrooptymalizacji

3. Przyspieszanie aplikacji PHP poprzez konfigurację stosu technologicznego

3.1. Dobór właściwego oprogramowania

3.1.1. Wersja PHP a performance aplikacji

3.1.2. Wybór web-serwera i związane z tym konsekwencje

3.1.2.1. Apache2 + mod_php

3.1.2.2. Lighttpd

3.1.2.3. nginx + php-fpm

3.2. Opcode caching

3.2.1. Zasada działania i wpływ na performance aplikacji

3.2.2. Rozwiązania

3.2.2.1. APC

3.2.2.2. XCache

3.2.2.3. opcache

3.2.3. Przyspieszanie aplikacji z włączonym opcode cache

3.2.3.1. Wykorzystanie buildów kodu źródłowego

3.2.3.2. Unikanie operacji fstat i związane z tym konsekwencje

3.2.4. Pozostałe aspekty

3.2.4.1. Wykorzystanie modułów opcode jako shared-memory user-cache

3.2.4.2. Monitorowanie skuteczności cache

3.2.4.3. Przegląd możliwych problemów i metod ich rozwiązania

4. Frameworki PHP

4.1. Przegląd popularnych frameworków PHP i ich natywnej wydajności

4.2. Tuning wydajności aplikacji poprzez wykorzystanie cache

4.2.1. Cache danych z bazy danych

4.2.2. Cache requestów HTTP

4.3. Tuning wydajności aplikacji poprzez jej architekturę

5. Optymalizacja schematu bazy danych i zapytań SQL (MySQL)

5.1. Silniki składowania danych i ich możliwości

5.2. Identyfikacja wąskich gardeł bazy danych

5.3. Podstawowa optymalizacja schematu bazy danych

5.4. Podstawowa optymalizacja zapytań SQL

6. Cache danych

6.1. Wprowadzenie do cache'owania danych

6.1.1. Cache hit vs miss

6.1.2. Algorytmy cache

6.1.3. Inwalidacja i czas życia cache

6.2. Narzędzia

6.2.1. Memcached

6.2.2. Redis

6.2.3. AWS ElastiCache

6.3. Strategie budowania cache

6.3.1. Przechowywanie danych per query vs per row

6.4. Przegląd typowych use-case'ów związanych z cache danych

6.5. Optymalizacja komunikacji pomiędzy aplikacją a serwerami cache

7. Odciążanie baz danych

7.1. Eliminacja części zapytań SQL i przenoszenie ich do dedykowanych rozwiązań

7.1.1. Full-Text-Search

7.1.1.1. Przegląd problemów wynikających z implementacji w języku SQL

7.1.1.2. Narzędzia

7.1.1.3. Full-Text-Search w MySQL 5

7.1.1.4. Sphinx

8. Cache requestów HTTP

8.1. Wprowadzenie do cache'owania danych

8.1.1. Nagłówki HTTP i ich obsługa w przeglądarce internetowej

8.1.2. Efektywność cache a obciążenie serwera aplikacji

8.2. Narzędzia

8.2.1. Nginx

8.2.2. Squid

8.2.3. Varnish

8.3. Konfiguracja i wykorzystanie serwera Varnish w stosie technologicznym

8.3.1. Zasada działania

8.3.2. Skuteczność i wydajność

8.3.3. Instalacja, konfiguracja oraz uruchomienie serwera Varnish

8.3.4. Wprowadzenie do języka VCL i jego możliwości

8.3.5. Podstawowe aspekty konfiguracji serwera Varnish i integracji z chronioną aplikacją

9. Wydajność frontendu aplikacji

9.1. Podstawowe techniki zwiększające szybkość działania aplikacji w przeglądarce internetowej

9.2. Optymalizacja komunikacji klient-serwer

10. Przetwarzanie danych w modelu asynchronicznym

10.1. Zmiana miejsca przetwarzania dużych ilości danych jako metoda przyśpieszenia aplikacji

10.2. Systemy kolejkowe

10.2.1. Zasada działania

10.2.2. Przegląd typowych use-case'ów związanych z przetwarzaniem asynchronicznym

10.2.3. Narzędzia

10.2.3.1. Gearmand

10.2.3.2. RabbitMQ

10.2.3.3. Redis

10.2.3.4. AWS Simple Queue Service

10.2.4. Konfiguracja i wykorzystanie serwera RabbitMQ w stosie technologicznym

10.2.4.1. Instalacja i uruchomienie

10.2.4.2. Podstawowe aspekty pracy i możliwości serwera RabbitMQ

10.2.4.3. Generowanie i przetwarzanie wiadomości przesyłanych za pomocą kolejek

11. Skalowanie aplikacji

11.1. Korzyści wynikające z rozproszenia aplikacji pomiędzy większą liczbę serwerów

11.2. Podstawowe aspekty pracy z rozproszoną aplikacją

11.2.1. Kierownie ruchem użytkowników pomiędzy instancjami aplikacji

11.2.2. Utrzymanie sesji użytkownika

12. Dedykowane rozwiązania zewnętrzne

12.1. HHVM, maszyna wirtualna dla języka PHP