Tworzenie aplikacji bazodanowych

wykład

Joanna Kołodziejczyk

2016

Plan wykładu

Liczba użytkowników

Jeden użytkownik

DBMS znajduje się na jednym komputerze i jest przeznaczona dla jednego użytkownika. Wykorzystywane do celów prywatnych i małych firm. Tylko jedna osoba może korzystać z bazy danych w czasie. Jeśli jeden użytkownik korzysta z bazy danych inni użytkownicy, muszą czekać, aż sesja zostanie zakończona.

Wielu użytkowników

Stosowane dla większych firm. Jest to baza danych, która jest dostępna za pośrednictwem sieci. Więcej niż jedna osoba może uzyskać dostęp i zmieniać dane w systemie. Większość używa rodzaj zamka w bazie danych, tak, że nie ma żadnych konfliktów między użytkownikami w trakcie dokonywania zmian. Dostęp z jednego lub wielu komputerów.

Systemy klient-serwer i bazy danych n-tier (wielowarstwowa architektura)

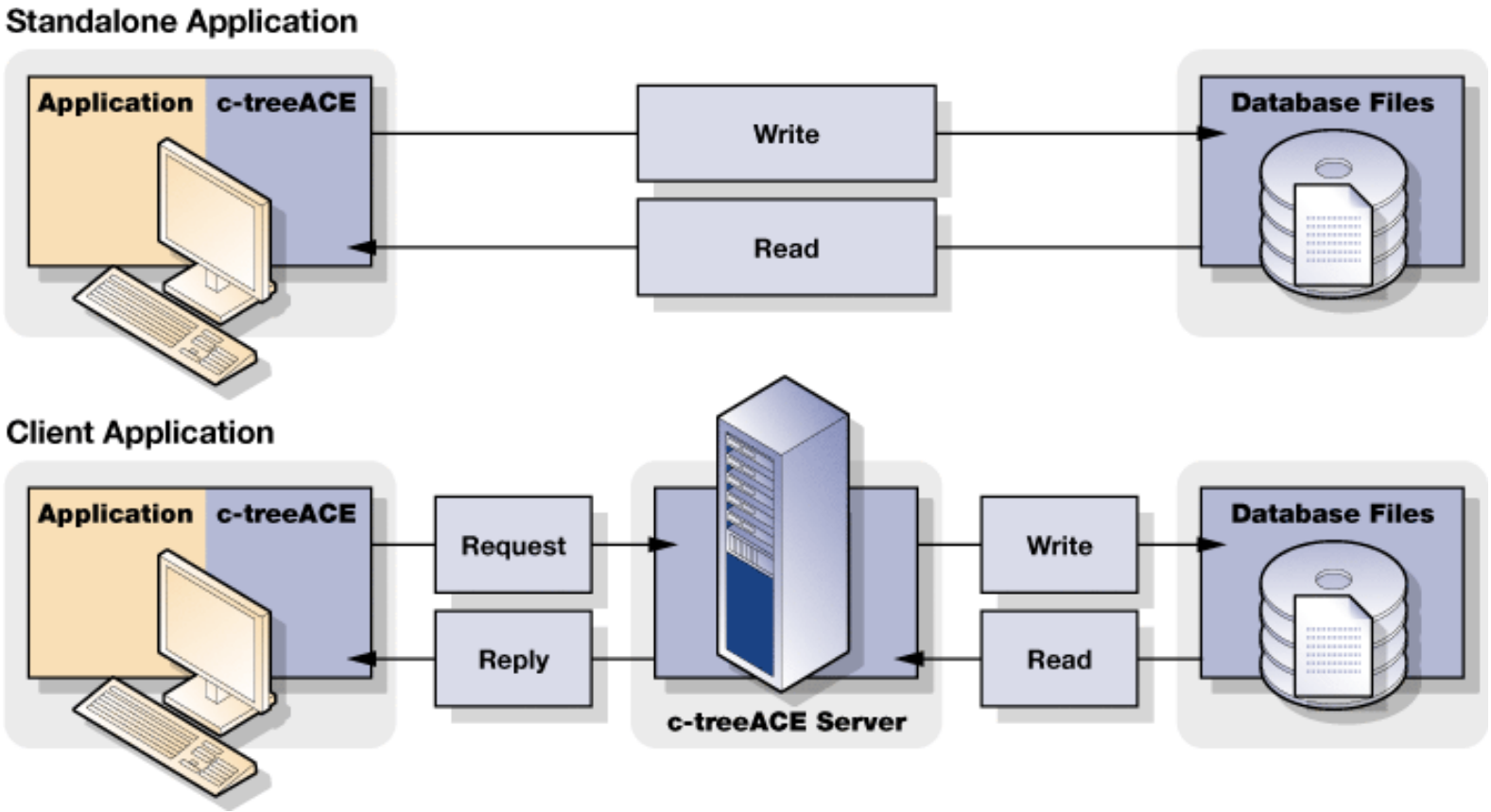
Architektura klient-serwer

Zasadniczo systemy baz danych klient-serwer to serwery, które przechowują i udostępniają zasoby innym komputerom. Gdy klient wysyła żądanie usługi do serwera, ten realizuje usługę. Klient jest określany jako front-end a serwer bazy danych jest określany jako back-end.

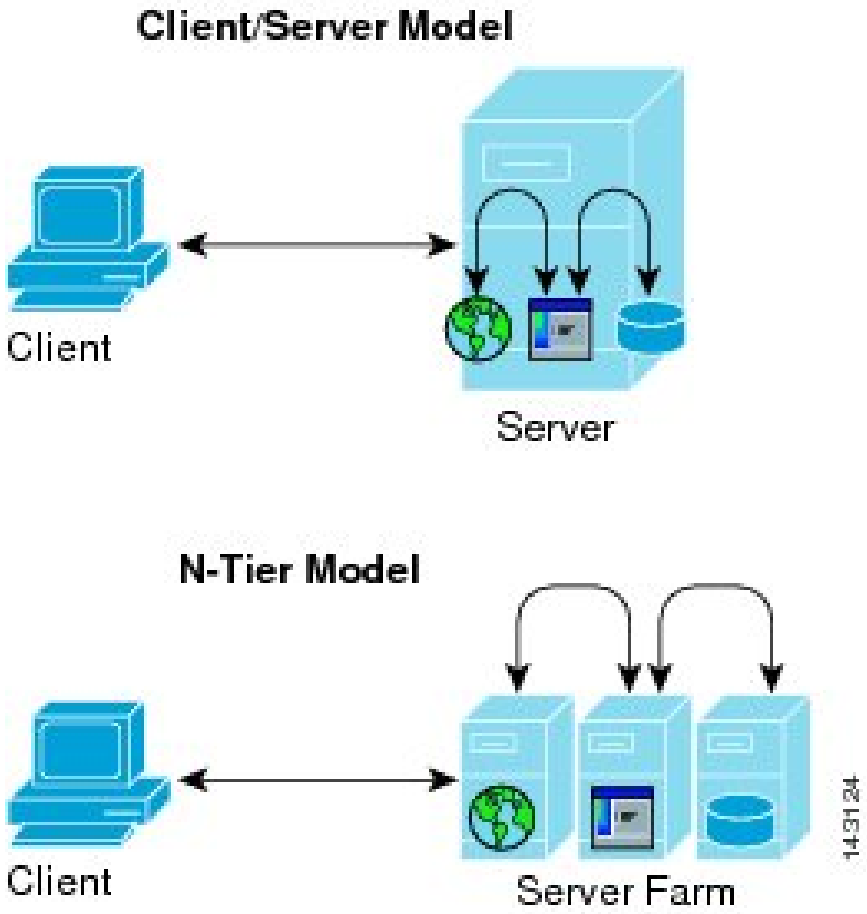
Architektura wielowarstwowa

Niektóre systemy baz danych klient-serwer mają nie tylko front-end i back-end, ale także pośrednie części nazywane warstwami. W tych rozwiązaniach klient i baza danych nigdy nie komunikują się bezpośrednio, wszystkie dane są przekazywane za pośrednictwem warstwy środkowej.

Systemy klient-serwer i bazy danych n-tier (wielowarstwowa architektura)



Systemy klient-serwer i bazy danych n-tier (wielowarstwowa architektura)



Bazy danych n-tier (wielowarstwowa architektura) — zalety

- Warstwa środkowa zapewnia warstwę abstrakcji, w ten sposób można zmienić części back-endu bez konieczności zmiany front-endu.
- Dobry sposób, aby rozdzielić odpowiedzialność.
- Mogą też być bardziej efektywne.
- Możliwość tworzenia efektywnych aplikacji w sieci do rozwiązań biznesowych.
- Middleware to istotna część pozwalająca aplikacjom po stronie klienta rozmawiać z serwerem, np. klient pocztowy.

Systemy scentralizowane i rozproszone

Systemy scentralizowane

Znajdują się na jednym serwerze/mainframe. Zaletą scentralizowanego systemu bazy danych jest to, że wszystkie informacje są w jednym miejscu. Wadą może być pojawienie się wąskiego gardła. Mając wszystkie informacje na jednym komputerze ułatwia się dostęp tylko niektórym użytkownikom, ale utrudnia innym, którzy chcą uzyskać dostęp do plików.

Systemy rozproszone

Dane podzielone pomiędzy wiele komputerów w sieci. Jedną z zalet systemów rozproszonych baz danych jest to, że dostęp do bazy danych można uzyskać z dowolnego komputera w sieci, nawet jeśli wszystkie informacje, nie są na jednym komputerze. Jest to preferowany typ, ponieważ informacje można łatwo znaleźć. Zapewnia również, że prawie niemożliwa jest utrata wszystkich danych.

In-memory database

Baza danych w pamięci (IMDB) to baza danych, której dane są przechowywane w pamięci głównej, aby ułatwić szybsze czasy reakcji. In-memory database są również czasami określane jako Main Memory Data Base lub MMDBs i stają się coraz bardziej popularne i wykorzystywane do obliczania wysokiej wydajności (HPC) i aplikacji Big Data. Aplikacje, z urządzeń sieci telekomunikacyjnych i sieci reklam mobilnych, często korzystają z IMDB.

In-memory database — co przyczynia się do popularności

- 64-bitowe obliczenia
- wielordzeniowe procesory
- ceny RAMu

Bazy przechowywane są w formie skompresowanej i nierelacyjnej. Szybciej przetwarzają zapytania. Mają wewnętrzne algorytmy optymalizacji i wykonują mniej instrukcji procesora. Ułatwiają też analizę danych, gdyż wszystkie dane przechowywane są w jednej bazie.

Plan wykładu

Definicja problemu

Budując aplikację webową należy podjąć decyzję, jaka architektura serwerów jest najlepsza dla środowiska. Czynniki, które mają wpływ na decyzję to:

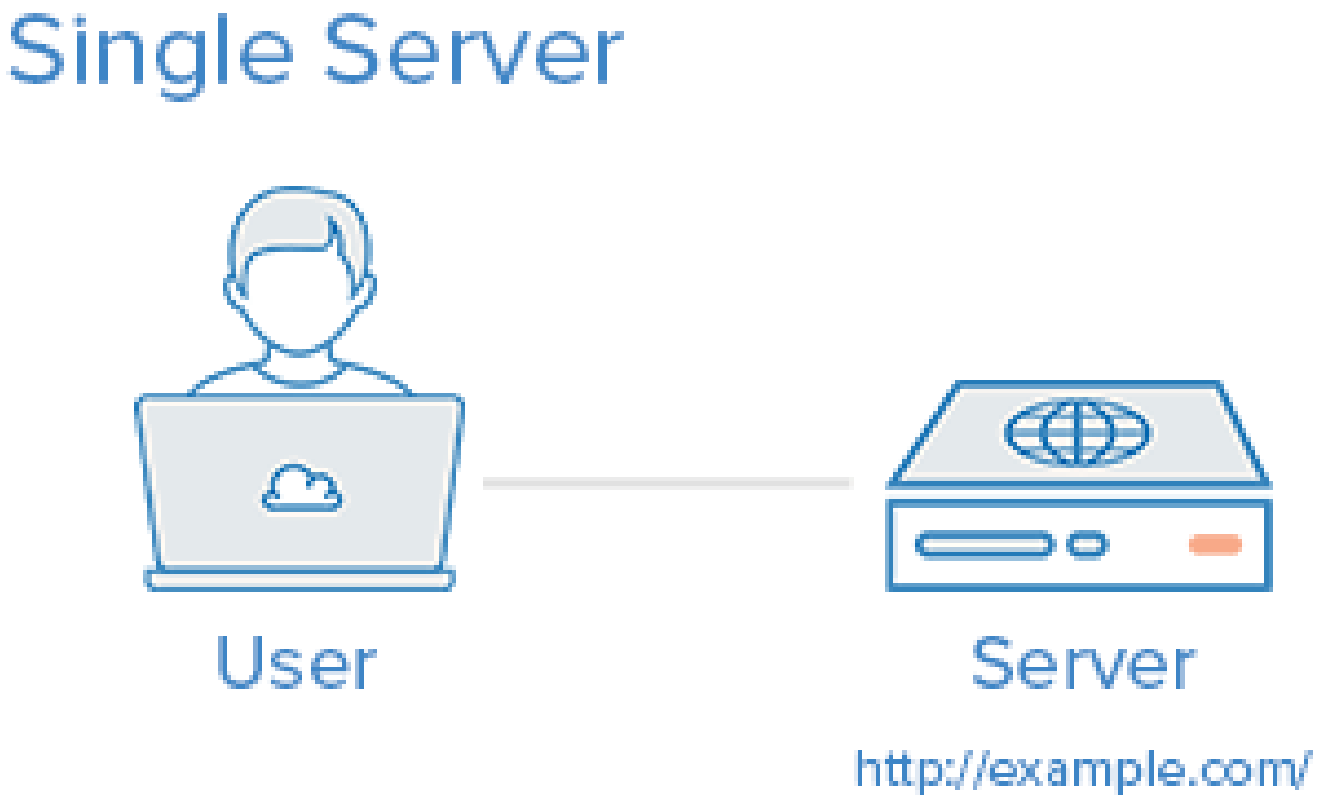
- wydajność,
- skalowalność,
- dostępność,
- niezawodność,
- koszty,
- łatwość zarządzania.

Rozwiązanie jednoserwerowe

Całe środowisko znajduje się na pojedynczym serwerze. Dla typowej aplikacji internetowej, która zawiera serwer WWW, serwer aplikacji i serwer bazy danych. Typowym przykładem tej konfiguracji jest stos LAMP, co oznacza Linux, Apache, MySQL, PHP na jednym serwerze.

Przypadek użycia: dobry do tworzenia szybkich aplikacji, jest to najprostsza możliwa konfiguracja, ale oferuje niewielkie możliwości skalowalności i oddzielenia komponentów.

Rozwiązanie jednoserwerowe



Rozwiązanie jednoserwerowe

Zalety

- Prosty

Wady

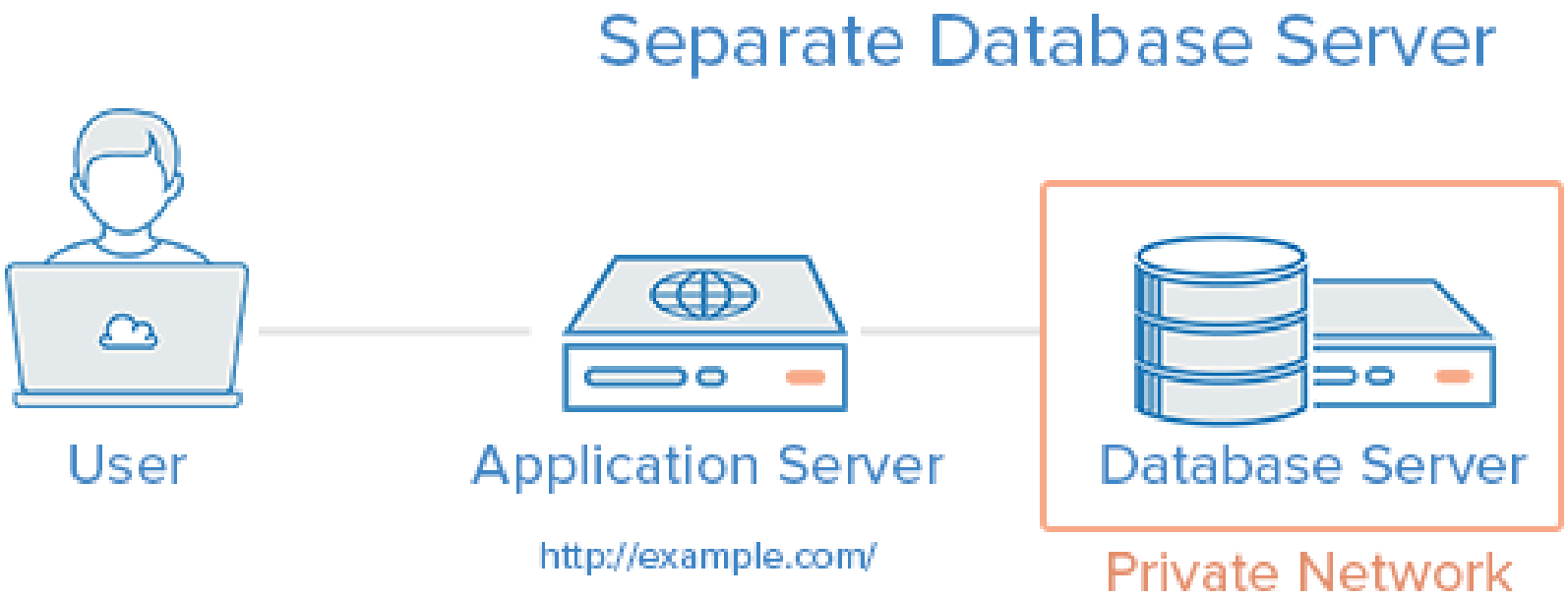
- Aplikacje i baza danych walczą o te samych zasobów serwera (procesor, pamięć, I / O, etc.), które, oprócz możliwych słabych osiągów, mogą sprawić, że trudno będzie określić źródło niskiej wydajności (aplikacja lub bazy danych).
- Niemożliwe skalowanie poziome.

Rozwiązanie: niezależny serwer bazy danych

System zarządzania bazą danych (DBMS) można oddzielić od reszty środowiska by wyeliminować dzielenie zasobów między aplikacją a bazą danych, oraz do zwiększenia bezpieczeństwa poprzez usunięcie bazy danych z DMZ (strefa zdemilitaryzowana bądź ograniczonego zaufania) lub publicznym internecie.

Przypadek użycia: Dobry do szybkiego tworzenia aplikacji, ale baza danych oddzielona od aplikacji nie walczy o te same zasoby systemowe.

Rozwiązanie: niezależny serwer bazy danych



Rozwiązanie: niezależny serwer bazy danych

Zalety

- Aplikacje i bazy danych nie walczą o te same zasoby serwera (procesor, pamięć, I / O, itd.).
- Możliwość pionowego skalowania każdej elementu oddzielnie, poprzez dodanie większej ilości zasobów na dowolny serwer, który potrzebuje zwiększonej pojemności.
- W zależności od konfiguracji, może zwiększyć bezpieczeństwo poprzez usunięcie bazy danych z DMZ.

Wady

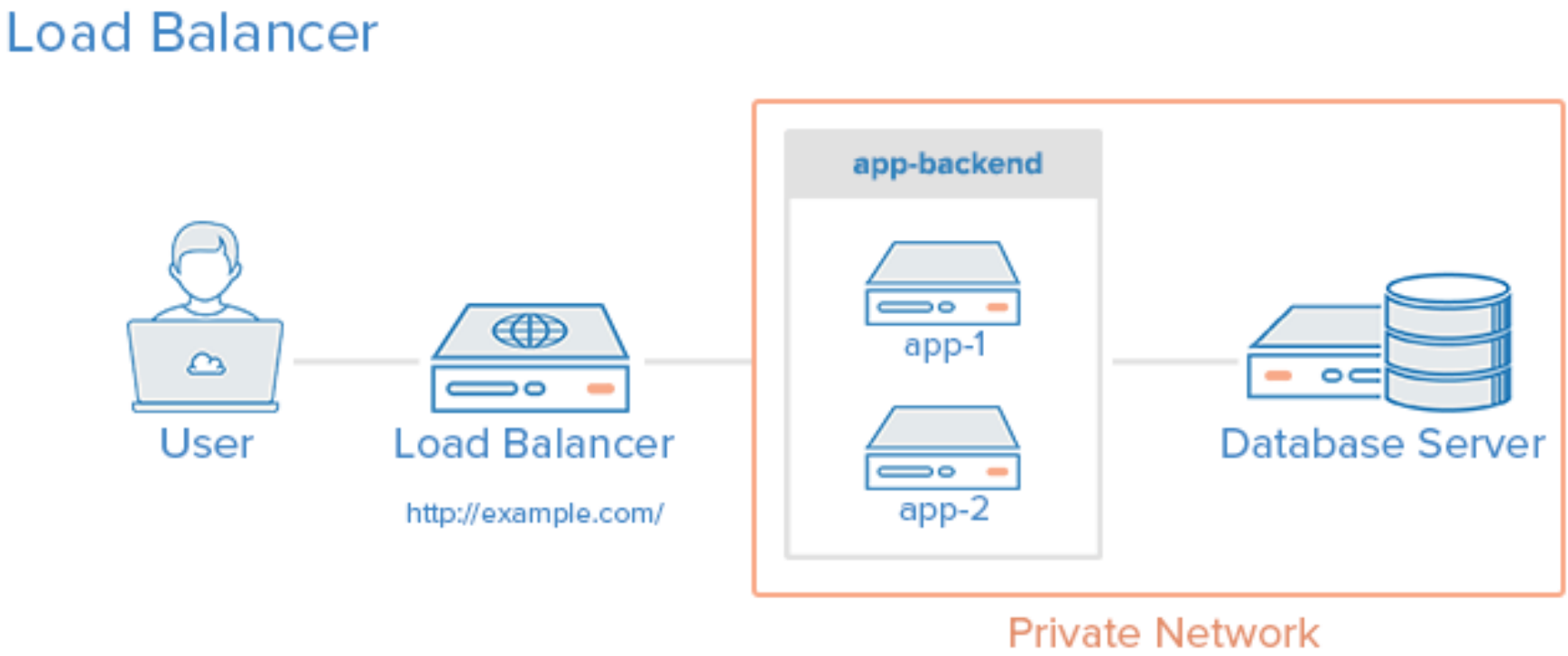
- Nieco bardziej skomplikowane rozwiązanie niż jednoserwerowa konfiguracja.
- Możliwe problemy z wydajnością, gdy połączenie sieciowe pomiędzy dwoma serwerami jest wysokiej latencji (czyli serwery są geograficznie odległe od siebie), lub przepustowość jest zbyt niska dla ilości przesyłanych danych.

Rozwiązanie: równoważenie obciążenia — reverse proxy

Równoważenie obciążenia może być dodawane do środowiska serwera, aby poprawić wydajność i niezawodność poprzez dystrybucję obciążenia na wiele serwerów. Jeśli jeden z serwerów zostanie wyłączony, pozostałe serwery będą obsługiwać ruch przychodzący, dopóki serwer nie zostanie „uzdrowiony". Może on być również stosowany do obsługi wielu aplikacji w tej samej domenie i na tym samym porcie, za pomocą warstwy 7 (warstwy aplikacji) — reverse proxy.
Przykłady oprogramowania zdolnego do równoważenia obciążenia reverse proxy : HAProxy, Nginx, i Varnish.

Przypadek użycia: Przydatne w środowisku, które wymaga skalowania poprzez dodawanie większej liczby serwerów, znany również jako poziome skalowanie.
`http://blogs.technet.com/b/plitpromicrosoftcom/archive/2013/05/14/reverse-proxy-w-iis.aspx`

Rozwiązanie: równoważenie obciążenia — reverse proxy



Rozwiązanie: równoważenie obciążenia — reverse proxy

Zalety

- Umożliwia skalowanie poziome, czyli zwiększenie pojemności środowiska poprzez dodanie większej liczby serwerów.
- Może chronić przed atakami DDoS poprzez ograniczenie połączeń klientów do rozsądnej ilości i częstotliwości.

Wady

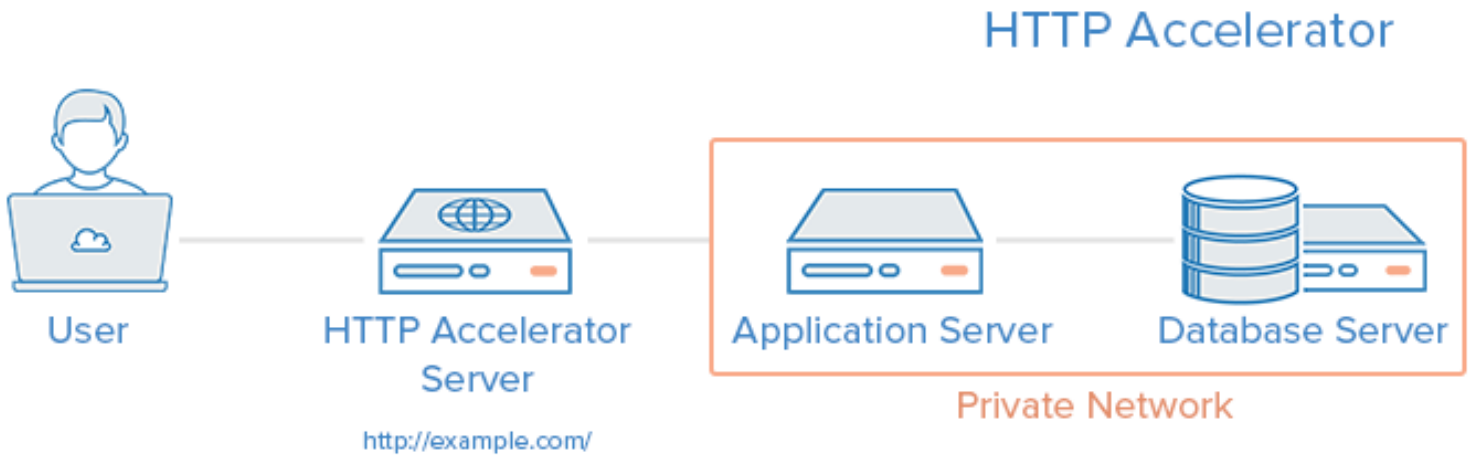
- Równoważenie obciążenia może stać się wąskim gardłem wydajności, jeśli nie ma wystarczających środków, lub jeśli jest słabo skonfigurowany.
- Może wprowadzać złożoność w systemie, która wymaga dodatkowych decyzji, takich jak, gdzie wykonywać kodowanie SSL w połączeniach szyfrowanych i jak obsługiwać aplikacje wymagające sticky session (przekierowanie zapytania do tego samego fizycznie urządzenia, które rozpoczęło sesję).

Rozwiązanie: HTTP Accelerator (buforowanie reverse proxy)

Akcelerator HTTP lub buforowanie reverse proxy, może być stosowane w celu zmniejszenia czasu potrzebnego do przekazania zawartości do użytkownika za pomocą różnych technik. Główną techniką przyjętą przez akcelerator HTTP jest buforowanie odpowiedzi z serwera WWW lub serwera aplikacji w pamięci, więc przyszłe zapytania o tej samej treści mogą być obsługiwane bardzo szybko. Realizuje się mniej zapytań z web serwerów lub serwerów aplikacji.
Przykłady oprogramowania zdolnego do akceleracji HTTP: Vernish, Squid, Nginx.

Przypadek użycia: Przydatne w środowisku dynamicznych aplikacjach WWW z dużą treścią, lub z wieloma powszechnie używanymi plikami.

Rozwiązanie: HTTP Accelerator (buforowanie reverse proxy)



Rozwiązanie: HTTP Accelerator (buforowanie reverse proxy)

Zalety

- Zwiększenie wydajności witryny poprzez zmniejszenie obciążenia procesora na serwerze WWW, poprzez buforowanie i kompresję, co zwiększa możliwości użytkownika.
- Może być użyty jako równoważenie obciążeń reverse proxy.
- Niektóre programy buforowania mogą chronić przed atakami DDoS.

Wady

- Wymaga dostrojenia, aby uzyskać najlepszą wydajność.
- Jeśli wskaźnik cache-hit jest niski, może to zmniejszyć wydajność.

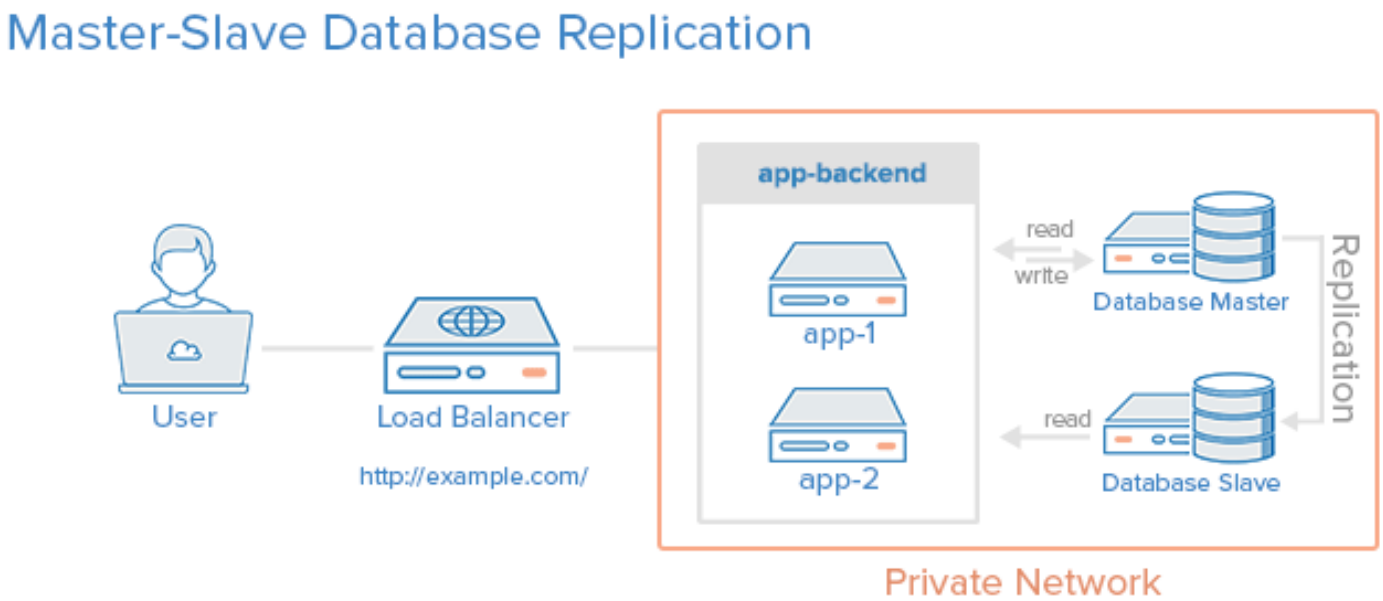
Rozwiązanie: replikacja bazy danych w architekturze master-slave

Jednym ze sposobów poprawy wydajności systemu bazy danych, która wykonuje wiele czytań w porównaniu do liczby zapisów, takich jak CMS, jest użycie replikacji bazy danych w architekturze master-slave. Replikacja master-slave wymaga jednego węzła głównego oraz jednego lub więcej węzłów podrzędnych. W tej konfiguracji, wszystkie aktualizacje są wysyłane do węzła głównego i odczyty mogą być rozłożone na wszystkie węzły.

Przypadek użycia: dobry dla zwiększenia wydajności odczytu bazy danych w warstwie aplikacji.

Rozwiązanie: replikacja bazy danych w architekturze master-slave

Rozwiązanie z pojedynczym węzłem typu slave



Rozwiązanie: replikacja bazy danych w architekturze master-slave

Zalety

- Poprawia wydajność odczytu bazy danych poprzez rozproszenie odczytu w węzłach slave.
- Może poprawić wydajność zapisu poprzez wyłącznie węzła master tylko do aktualizacji/zapisu (nie spędza czasu na obsługiwaniu żądań odczytu).

Wady

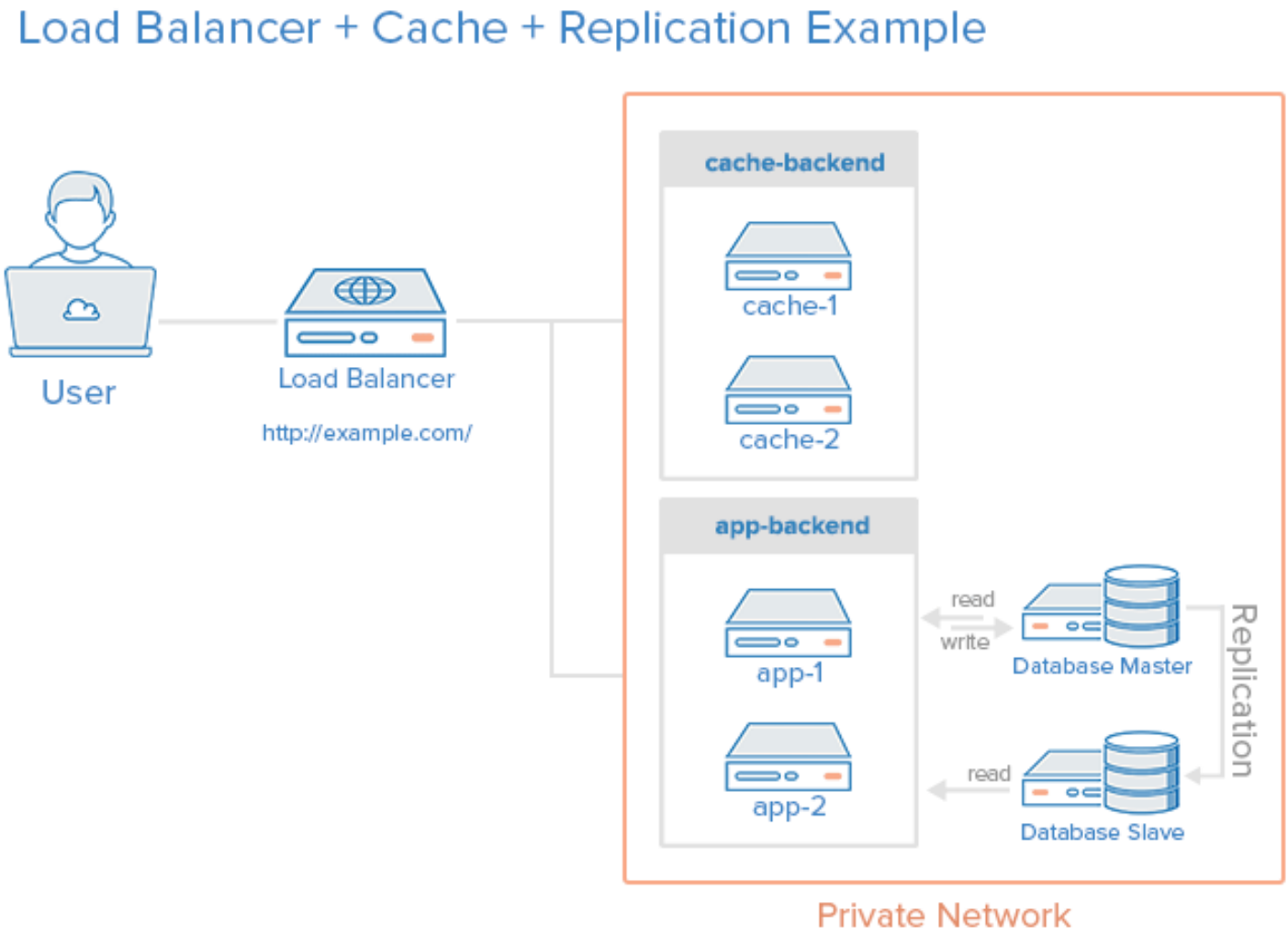
- Aplikacja korzystająca z bazy danych musi mieć mechanizm określający, które węzły służą do zapisu, a które do odczytu.
- Aktualizacje węzłów slave są asynchroniczne, więc istnieje ryzyko, że ich zawartość nie będzie aktualna.
- Jeśli węzeł główny ulegnie awarii, nie można wykonywać wpisów do bazy danych, aż problem nie zostanie rozwiązany.

Przykład złożonego rozwiązania

Możliwe jest równoważenie obciążenia serwerów buforujących, oprócz serwerów aplikacyjnych i używanie replikacji bazy danych w pojedynczym środowisku. Celem jest połączenie tych technik, aby czerpać korzyści z każdej bez wprowadzania zbytniej złożoności.

Załóżmy, że równoważenie obciążenia jest skonfigurowane do obsługi żądań statycznych (takich jak obrazy, CSS, JavaScript, itp) i wysyła te zapytania bezpośrednio do serwerów buforowania a inne żądania wysyła do serwerów aplikacji.

Przykład złożonego rozwiązania



Jak obsługuje się zapytanie użytkownika — przykład

1. Użytkownik żąda dynamicznej zawartości z http://example.com/ (load balancer).
2. Load balancer przesyła zapytanie do do app-backend
3. App-backend czyta z bazy danych i zwraca zawartość do load balancer.
4. Load balancer zwraca wymagane dane dla użytkownika

Jak obsługuje się zapytanie użytkownika o zawartość statyczną — przykład

1. Load balancer sprawdza cache-backend aby zobaczyć, czy żądana zawartość jest w pamięci podręcznej (cache-hit), czy nie (cache-miss).
2. Jeśli cache-hit: zwrócenie żądanej treści do load balancer i przejść do kroku 7. Jeśli cache-miss: serwer cache przekazuje żądanie do app-backend, przez load balancer.
3. Load balancer przekazuje żądanie aż do app-backend
4. App-backend odczytuje z bazy danych, a następnie zwraca zawartość do load balancer.
5. Load balancer przesyła odpowiedź do cache-backend
6. Cache-backend buforuje zawartość i zwraca go do load balancer.
7. Load balancer zwraca wymagane dane dla użytkownika

Przykład złożonego rozwiązania — wady

To środowisko ma jeszcze dwa słabe punkty (równoważenie obciążenia i serwer baz danych jako master), ale oferuje wszystkie inne korzyści, niezawodność i wydajności, które zostały opisane przy niezależnych rozwiązaniach.

Plan wykładu

Opracowanie strony internetowej z bazą danych online

Strony internetowe mogą zacząć swoje istnienie jako statyczna strona informacyjna, dopóki właściciel strony nie zda sobie sprawy, że trzeba umieścić kilka danych strukturalnych.
Dane te mogą być proste w konstrukcji, lub złożone. Tak czy inaczej dane można pozyskać online i zintegrowane ze stroną.

Plan wykładu

Do opracowania tej części wykładu wykorzystano

1. „5 Common Server Setups For Your Web Application" by Mitchell Anicas `https://www.digitalocean.com/community/tutorials/5-common-server-setups-for-your-web-application`
2. Wprowadzenie do baz danych `https://en.wikibooks.org/wiki/Introduction_to_Computer_Information_Systems/Database`