Jak rozproszyć swoją web aplikację?

W tym artykule pokażę, jak wykorzystać dostępne dla Pythona narzędzia, aby maksymalnie rozproszyć aplikację Django na wiele mniejszych i tańszych maszyn, korzystając na przykład z budżetowych serwerów dedykowanych lub kolokowanych serwerów "refurbished". Równie dobrze będziemy mogli wykorzystać wirtualne maszyny od sprawdzonego usługodawcy.

BAZA DANYCH

Prawie każda aplikacja potrzebuje jakiegoś magazynu danych. Zwykle podczas tworzenia aplikacji webowych pierwsze, co przychodzi nam do głowy jako baza, to MySQL. Posiada on wiele narzędzi pozwalających na wykorzystanie go w wygodny sposób, zwłaszcza w Pythonie. Jednak skalowanie go i tworzenie klastra może być nie lada problemem dla początkujących w tej dziedzinie administratorów. Co więcej – jedna, centralna baza danych prawie zawsze będzie słabym ogniwem naszej infrastruktury. Jeśli nawet nie pod względem wydajności, to pod względem redundancji i dostępu do niej. Mamy kilka możliwości. aby poradzić sobie z tym problemem.

Redis i bazy NoSQL

Pierwszym podejściem jest przejście na bazy typu NoSQL (np. Redis), które korzystają z zupełnie innego podejścia do przechowywania i odczytywania danych. Wspomniany Redis pozwala zapisywać pary klucz-wartość przy jednoczesnym łatwym skalowaniu. Nie udostępnia żadnych mechanizmów znanych z baz SQL, takich jak tabele, JOIN'y i całej masy innych, potencjalnie wygodnych udogodnień. Na początku może to wydawać się dość dużym utrapieniem – nie mamy żadnych struktur danych i przy złym podejściu może to doprowadzić do dużego bałaganu. Z drugiej strony, jeśli zapewnimy sobie choć trochę dyscypliny podczas pisania kodu, możemy z tego stworzyć całkiem elastyczne narzędzie. Jak zatem poradzić sobie z brakiem tabel? Ano tak – bazy Key-Value potrafią bardzo szybko odnajdywać wartości poszczególnych kluczy. Zamiast definiować tabele, możemy po prostu stworzyć całą serię kluczy z odpowiednimi nazwami, w których będą przechowywane wartości. Na przykład obiekty klasy `User`, która ma pola `first_name`, `last_name`, `date_of_birth`, możemy zapisać tak:

User:1:first_name	Jan
User:1:last_name	Nowak
User:1:date_of_birth	1979
User:2:first_name	Karol
User:2:last_name	Kowalski
User:2:date_of_birth	2001
...	

Tabela 1. Przykładowe wartości kluczy

Nie widać tutaj konkretnej struktury tabel, jednak przeglądając wszystkie klucze zaczynające się od `User:2:`, możemy przeczytać wszystkie pola danego obiektu. Dodatkowo Redis pozwala w bardzo przyjemny sposób przeszukiwać swój zbiór danych przez dodanie znaków takich jak * i ? w celu pobrania zakresu kluczy. Aby użyć redisa będziemy potrzebowali sam serwer oraz moduł Pythona:

```
sudo apt-get install redis-server
sudo pip install redis
```

Spróbujmy wykonać powyższy kod w Pythonie:

Listing 1. Przykład użycia Redis w Pythonie

```
import redis
conn = redis.Redis('localhost')
conn.set('User:1:first_name', 'Karol')
conn.set('User:1:first_name', 'Kowalski')
conn.set('User:2:first_name', 'Jan')
conn.set('User:2:first_name', 'Nowak')
```

Następnie wypiszmy imię użytkownika o `ID=2`:

```
print(conn.get('User:2:first_name'))
```

I wypiszmy imiona wszystkich użytkowników, korzystając z metody `keys` z odpowiednim filtrem:

```
for key in conn.keys('User:*:first_name'):
  print(conn.get(key))
```

No dobrze – mamy pisać własny ORM[1] do zarządzania danymi w Redisie? Nie! Wiele frameworków pozwala użyć Redis'a właśnie w ten sposób. Dzięki temu nie musimy się martwić o to, jak zarządzać danymi. Z drugiej strony musimy się poważnie zastanowić, czy takie rozwiązanie będzie dla nas wydajne. Wszystko oczywiście zależy od tego, co w danej chwili potrzebujemy. Inaczej będzie wyglądało obciążenie bazy dla dużego serwisu, który jedynie pokazuje produkty, które są mało powiązanymi ze sobą danymi, a inaczej dla aplikacji analizującej zachowania użytkowników w takim sklepie, gdzie znajduje się dużo powiązań pomiędzy tabelami.

Warto popatrzeć tutaj na inną zaletę takiego rozwiązania – jeżeli nasz kod będzie odpowiednio przygotowany, to migracja struktury danych nie będzie dużym problemem, przez który trzeba wyłączyć pół serwisu. Co, jeśli w Redisie przy niektórych osobach pojawi się pole `User:ID:credit_card_number`? MySQL musiałby znać domyślne wartości dla każdego wpisu i wymagałby migracji tabel. Niektóre ORM'y po czymś takim nie uruchomiłyby naszej aplikacji, ponieważ klasa odwzorowująca tabelę w bazie różniłaby się polami. W przypadku umiejętnego wykorzystania NoSQL nie mamy tego problemu. Jedyne, co musimy zrobić, to mieć to na uwadze

1. Object-relational mapping – mechanizm pozwalający na odwzorowanie struktury tabel w bazie danych na klasy w naszej aplikacji. Wykorzystanie tego typu narzędzi znacznie przyspiesza pisanie kodu i odpytywanie bazy danych, zwiększając przy tym bezpieczeństwo, np. poprzez eliminowanie punktów potencjalnie podatnych na SQL injection.