Problematyka rozproszonych systemów - przypadek *Apache Cassandra*

Marek Lewandowski
Wydział Elektroniki i Technik Informacyjnych, Politechnika Warszawska
marek.m.lewandowski@gmail.com

Streszczenie. *Artykuł omawia nierelacyjne bazy danych i twierdzenia z nimi związane takie jak* CAP, BASE*. W szczególności omawia cechy szczególne i budowę bazy* Apache Cassandra. Apache Cassandra *jest kanoniczną nierelacyjną bazą danych, której zrozumienie znacząco ułatwia zrozumienie większości pozostałych nierelacyjnych baz danych.*

1. Motywacja. Nierelacyjne bazy danych powstały, w celu rozwiązania nowego rodzaju problemów, którym nie były w stanie sprostać tradycyjne bazy danych. Problemy to bardzo duża, szybko przyrastająca ilość danych oraz potrzeba obsługi równie dużego obciążenia systemu.

Ilość danych, wyrażana może być, aż peta bajtach ($1PB = 1000TB$), co może sprawiać problemy jeśli chcielibyśmy obsłużyć taką ilość danych jednym komputerem. W związku z tym poruszamy się po dziedzinie klastrów serwerów, a nie pojedynczych maszyn. Pojedyncza maszyna to również pojedynczy punkt awarii. Takie i inne problemy adresowane są przez nierelacyjne bazy danych.

Często podczas omawiania nierelacyjnych baz danych przytacza się przypadek firmy *Amazon*. *Amazon* oświadczył, że wzrost czasu odpowiedzi w ich serwisie o 0.1 sekundy powoduje spadek sprzedaży o 1%. Wysoka wydajność i niski czas dostępu są możliwe dzięki nierelacyjnym bazom danych oraz innym pobocznym technikom. Warto zatem pamiętać o biznesowej wartości tych rozwiązań, wiedzieć jak działają i jakim prawom podlegają.

2. Spójność i dostępność W klasycznych bazach danych, często słyszymy o tym, że baza jest spójna. Taka spójność oznacza, że wszystkie mechanizmy w bazie takie jak więzy integralności, klucze obce, ograniczenia na kolumny są spełnione. Chcę przedstawić inną definicję spójności.

Spójność oznacza jedną wersję danych, a więc wszystkie takie same dane, są zawsze w identycznej wersji. Dostępność rozumiana jest jako zdolność systemu do obsługi żądania zapisu/odczytu danych i zwrócenia odpowiedzi.

ACID charakteryzuje klasyczne bazy danych, które posiadają bardzo wysoką spójność. *BASE* to akronim, który należy omówić.

- większość danych dostępna przez cały czas (ang. Basically Available)
- dane wystarczająco świeże (ang. Soft state)
- osiągnięcie spójności odsunięte w czasie, ale osiągalne (ang. Eventually consistent)

ACID i *BASE* reprezentują dwa podejścia projektowe po przeciwnych końcach spektrum spójność-dostępność. Naturalnie *ACID* znajduje się po stronie spójności i reprezentuje klasyczne podejście do baz danych. *BASE* znajduje się po przeciwnej stronie spektrum i reprezentuje systemy z dużą dostępnością. Dzięki tym modelom wybór pomiędzy dostępnością, a spójnością jest jawny.

3. Twierdzenie CAP. Brewer opublikował w 1999 roku [2] twierdzenie przedstawiające wybór dwóch z trzech cech w rozproszonych bazach danych. Dwie cechy już znamy. Należy wytłumaczyć czym jest odporność na partycje.

Partycja w rozprosznym systemie występuje, wtedy gdy część węzłów w klastrze nie ma łączności z co najmniej jednym węzłem, inaczej mówiąc klaster jest podzielony na dwie części, które nie mogą się ze sobą komunikować. Taka sytuacja może wystąpić w co najmniej dwóch przypadkach. Część maszyn może ulec awarii i tym samym zniknie z klastra na długi czas, a więc będzie niedostępna z punktu widzenia pozostałych maszyn. Inną możliwością jest brak łączności spowodowany awarią sieci. Z punktu widzenia maszyn w klastrze obie sytuacje są identyczne i nie da się ich rozróżnić. W obu tych sytuacjach występuje partycja rozdzielająca maszyny na dwie grupy. Twierdzenie Brewera mówi, że można mieć jedynie dwie cechy z poniższych:

- spójność,
- dostępność,
- odporność na partycje.

W przypadku rozproszonych systemów brak odporności na partycje to pojedyczny punkt awarii. Wystarczy sobie wyobrazić, że pojedyncza maszyna ulega awarii, a baza nie jest w stanie dalej funkcjonować. Z tego powodu rozważa się wybór pomiędzy dostępnością, a spójnością. Odporność na partycje powinna być zawsze. Można zatem mówić o binarnym wyborze.

Binarny wybór został zaprezentowany w oryginalnej postaci twierdzenia. Przez ponad dekadę powstało wiele typów systemów, które skupiają się na innych kombinacjach właściwości.

3.1. Twierdzenie dzisiaj. Wokół twierdzenia pojawiło się wiele nieporozumień. Przez lata odkryto sporo niuansów. Właściwości nie są binarne, a bliżej im do wartości ciągłych. Dostępność jest w zakresie procentowym. Istnieje wiele poziomów spójności. Węzły w klastrze mogą nie zgadzać się co do tego czy partycja faktycznie występuje. Autor opublikował 12 lat później artykuł [1], który rozwiewa większość wątpliwości.

Dzisiaj możemy przedstawić twierdzenie w lepszy sposób. Podczas wystąpienia partycji niemożliwa jest idealna spójność i idealna dostępność. Dzięki specjalnej obsłudze systemu podczas partycji, można mieć więcej niż tylko jedną cechę. Wymaga to użycia odpowiednik technik. Jedną z technik jest ograniczenie możliwych operacji, tylko do tych, które nie zniszczą spójności. Przykładem rozprosznego systemu, który stosuje taką technikę jest *Google Docs*, który podczas braku połączenia (partycji), udostępnia tylko część narzędzi tesktowych.

Jeśli partycja nie występuje system może mieć idealną dostępność i spójność. Dopiero podczas wystąpienia partycji należy dokonać wyboru pomiędzy spójnością, a dostępnością lub mieć specjalny tryb operacji podczas partycji. Nie dokonanie wyboru w krótkim czasie to wybór spójności, ponieważ żądanie nie zostanie obsłużone w odpowiednim czasie, a więc z perspektywy klienta system nie będzie dostępny.

4. Parametry nierelacyjnych baz danych.

4.1. Reguła spójności.

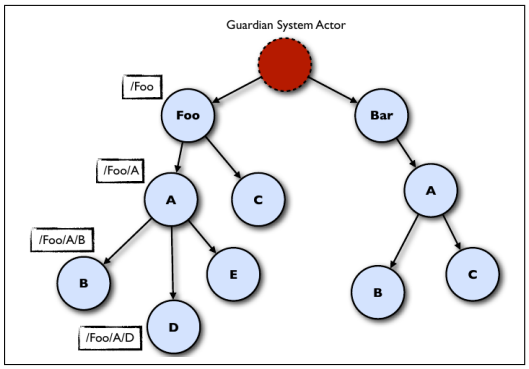
5. Apache Cassandra.

5.1. Liniowa skalowalność.

5.2. Natywny model danych.

5.3. Model CQL.

5.4. Przypadki użycia.



Rysunek 1. A picture of the same gull looking the other way!

6. Motywacja.

Literatura

[1] E Brewer. CAP twelve years later: How the "rules" have changed. *Computer*, 45(2):23–29, 2012.

[2] Armando Fox and Eric A. Brewer. Harvest, yield, and scalable tolerant systems. In *In HotOS-VII*. Society Press, 1999.