Łukasz Strobin
Adam Niewiadomski
Politechnika Łódzka

WIELOKROTNE PRZYSPIESZENIE DZIAŁANIA APLIKACJI POPRZEZ ZASTOSOWANIE TECHNOLOGII NIERELACYJNYCH BAZ DANYCH

Wprowadzenie

Relacyjny model danych został wprowadzony przez Codda w 1970 r. i jest cały czas najczęściej wybieranym modelem danych. Technologia ta jednak pochodzi z czasów, w których ilość danych do przetwarzania była znacznie mniejsza, przez co nie jest ona przystosowana do aktualnych wymagań aplikacji internetowych, gdzie ilość danych wyraża się niejednokrotnie w petabajtach (10^15 bajtów). Dodatkowo aplikacje internetowe, takie jak Facebook, są używane równocześnie przez niespotykaną wcześniej liczbę użytkowników. Ten rosnący problem zapoczątkował ruch, którego celem jest zaprojektowanie i wprowadzenie nowoczesnych technologii, które sprostają aktualnym wymaganiom. Takie firmy jak Google (BigTable[1]) czy Amazon (Dynamo[2]) mocno przyczyniły się do przyspieszenia idei NoSQL, czyli „Not Only SQL". W początkowej fazie tego ruchu wzięły udział również takie firmy jak Facebook (Cassandra), Apache (HBase) i LinkedIn (Voldemort)[3]. W celu dokładniejszego wprowadzenia do tematyki NoSQL czytelnik jest proszony o zapoznanie się z pracami przeglądowymi Jing Hana[4] oraz Nayaka i in.[5].

Celem niniejszego artykułu jest doświadczalne sprawdzenie wydajności baz nierelacyjych, czyli szybkości wykonywania operacji dla dużego obciążenia bazy oraz dla wielu użytkowników jednocześnie używających bazy.

---

1 F. Chang, *BigTable: A Distributed Storage System for Structured Data*, OSDI'06 Proceedings of the 7th USENIX Symposium on Operating System Design and Implementation – Vol. 7, 2006.

2 G. DeCandia, *Dynamo: Amazon's Highly Avaiable Key-Value Store*, SOSP'07 Proceedings of twenty-first ACM SIGOPS symposium on Operating System Principles, 2007.

3 E. Ellis, *NoSQL Ecosystem*, 2010, http://www.rackspacecloud.com/blog/2009/11/09/nosql-ecosystem.

4 Jing Han, E. Haihong, Guan Le, Jian Du, *Survey on NoSQL database*, „Pervasive Computing and Applications (ICPCA)" 2011.

5 A. Nayak, A. Poriya, D. Poojary, *Type of NOSQL Databases and its Comparison with Relational Databases*, „International Journal of Applied Information Systems" 2013, Vol. 5, No. 4.

Aby sprawdzić wydajność tych SZBD, zostały przeprowadzone doświadczenia wydajnościowe trzech typów baz danych – relacyjnej (MySQL), dokumentowej (MongoDB) oraz bazy danych typu klucz-wartość (Redis). Potwierdzono doświadczalnie wyższą wydajność tych baz oraz fakt, że różnica wydajności jest tym większa, im większe jest obciążenie bazy danych.

Wykazano również doświadczalnie przewagę grafowych baz danych nad relacyjnymi bazami danych dla kwerend złożonych (tj. takich, które wymagają wykonania wielu operacji złączenia).

Na przykładzie aplikacji rekomendującej wykazano, że zastosowanie nierelacyjnych baz danych może wielokrotnie zwiększyć szybkość działania. Jest to szczególnie istotne w przypadkach, w których czas odpowiedzi systemu jest kwestią kluczową.

1. Dokumentowe bazy danych i bazy typu klucz-wartość

Ten punkt stanowi krótkie wprowadzenie do dwóch typów nierelacyjnych baz danych – bazy dokumentowej oraz bazy typu klucz-wartość. Krótko omówiono model danych, porównano zapytania oraz scharakteryzowano możliwości, jakie oferują te bazy danych.

1.1. Model danych w dokumentowej bazie danych

Bardzo popularną implementacją dokumentowej bazy danych jest MongoDB. W tym podpunkcie są zaprezentowane najistotniejsze cechy tej bazy w porównaniu do bazy relacyjnej.

Głównym pojęciem w dokumentowej bazie danych jest tzw. dokument. Jest to zbiór danych zorganizowanych według standardu wybranego przez daną implementację SZBD (np. XML, YAML, JSON, BSON). W przetestowanej bazie danych, MongoDB, dokument jest zapisany w postaci BSON. Poniżej przedstawiono przykład dokumentu w bazie MongoDB:

```
{
   Imię: "Imie1",
   Nazwisko: "Nazwisko1",
   Adres: "Adres1"
 }
```

Model danych w MongoDB nie posiada sztywnego schematu. Dokumenty są przechowywane w kolekcjach, które nie narzucają struktury dokumentów. Wynika z tego, że:

- dokumenty w tej samej kolekcji nie muszą mieć takiego samego zbioru pól,
- takie same pola w różnych dokumentach mogą posiadać różne typy danych.

W praktyce większość dokumentów w danej kolekcji ma podobną, lecz nie identyczną, strukturę. Brak sztywnego schematu oznacza, że jest możliwe takie modelowanie dokumentów, które jest zbliżone do obiektów w aplikacji. Łatwe jest również modyfikowanie struktury bazy danych już przy działającej aplikacji, co w przypadku relacyjnej bazy danych może być bardzo uciążliwe.

Tak jak w modelu relacyjnym, dokumenty mogą być ze sobą powiązane. W MongoDB możliwe są dwa rozwiązania, aby powiązać dokumenty:

- zagnieżdżenie – zalecane przy relacjach typu jeden-do-jednego oraz przy relacjach jeden-do-wielu. O zagnieżdżeniu mówi się, jeśli jeden obiekt ma bezpośrednią referencję do drugiego obiektu;
- referencja – zalecane przy relacjach wiele-do-wielu. Obiekty posiadają jedno pole o takiej samej wartości. Po stronie aplikacji leży, aby te dane poprawnie zinterpretować

Konwersja istniejącej już relacyjnej bazy danych w dokumentową bazę danych jest możliwa, ponieważ wszystkie struktury bazy relacyjnej mogą być łatwo przetłumaczone do struktur dokumentowej bazy danych. Można wykonać tę operację pisząc własny program, lecz dostępne są również darmowe programy służące do tego celu, np. Mongify[6]. Program ten wymaga minimalnej konfiguracji i jest bardzo łatwy w użyciu. W pierwszym kroku jest pobierany schemat relacyjnej bazy danych do pliku, w którym użytkownik może określić zależności między obiektami w dokumentowej bazie danych. Następnie dane są automatycznie eksportowane z bazy relacyjnej do bazy dokumentowej.

1.2. Model danych w bazie klucz-wartość

Bazy typu klucz-wartość posiadają bardzo prosty model danych – jest to zbiór par klucz-wartość, co w językach programowania ma odpowiednik w tablicy. Pobieranie danych w takim modelu odbywa się poprzez podanie klucza, zatem możliwości operacji na danych są bardzo ograniczone, w szczególności wybieranie rekordów gdzie wartość spełnia dany warunek. W przypadkach, gdzie nie jest potrzebne pobieranie wyników na podstawie wartości, bazy tego typu mają natomiast znaczną przewagę w szybkości operacji, oraz przy dużych obciążeniach bazy danych. Warto również podkreślić, że w większości przypadków implementacji tego typu baz danych (np. Redis), wartość danego wpisu w bazie danych nie musi być ciągiem znaków – może być listą, zbiorem lub referencją do innego obiektu. Niektóre implementacje tego typu baz danych (np. Scalaris) oferują transakcyjność, przez co możliwe jest zbudowanie bardziej złożonej aplikacji, np. odpowiednika Wikipedii[7].

---

[6] A. Kalek, *Mongify*, 2011, www.mongify.com.

[7] N. Kruber, *Scalaris: Scalable Web Applications with a Transactional Key-Value Store*, 2012, http://2012.berlinbuzzwords.de/sites/2012.berlinbuzzwords.de/files/slides/Scalaris-nkruber-bbuzz12.pdf.

Konwersja z bazy relacyjnej do bazy typu klucz-wartość nie zawsze jest możliwa. Przez prosty model danych trzeba każdy przypadek rozpatrywać osobno i dopasować sposób reprezentacji danych do konkretnego zastosowania, a w szczególności do typów zapytań, jakie obsługiwać ma baza danych. Często, aby zapewnić szybkość i użyteczność danych w bazie typu klucz-wartość niezbędne jest wprowadzenie redundancji, ponieważ nie istnieje możliwość pobierania danych według wartości, tylko klucza. Gdy zatem jest wymagane pobieranie danych według dwóch atrybutów, niezbędne jest ustalenie każdego atrybutu jako klucza, przez co jest wymagane powtórzenie danej informacji kilkukrotnie.

2. Porównanie wydajności wybranych SZBD dla baz relacyjnych i nierelacyjnych

Porównano wydajność trzech typów baz danych – bazy relacyjnej, bazy dokumentowej oraz bazy typu klucz-wartość. Zastosowano typowe implementacje tych baz danych – MySQL, MongoDB (baza dokumentowa) oraz Redis (baza typu klucz-wartość).

2.1. Testy wydajnościowe przy różnym stopniu obciążenia

Testy wydajnościowe zostały przeprowadzone z wykorzystaniem frameworka YCSB (Yahoo Cloud Serving Benchmark)[8]. Framework ten pozwala na własną implementację czterech typów operacji: UPDATE, INSERT, READ oraz SCAN. Następnie możliwe jest określenie procentowego udziału danej operacji w tzw. obciążeniu (*workload*), np. możliwe jest określenie, że obciążenie ma składać się w 50% z operacji odczytu i w 50% z operacji wstawiania. Dzięki temu wydajność bazy danych może być przetestowana w różnych sytuacjach. Dodatkową możliwością oferowaną przez framework YCSB jest ustalanie ilości operacji na sekundę, dzięki czemu zostaje sprawdzona wydajność bazy danych przy ustalonym ruchu.

Do testów wykorzystano cztery typy obciążeń:

- obciążenie A – 50% operacji aktualizacji, 50% operacji odczytu,
- obciążenie B – 50% operacji aktualizacji, 50% operacji odczytu,
- obciążenie C – 100% operacji odczytu,
- obciążenie D – 80% operacji odczytu, 20% operacji wstawiania.

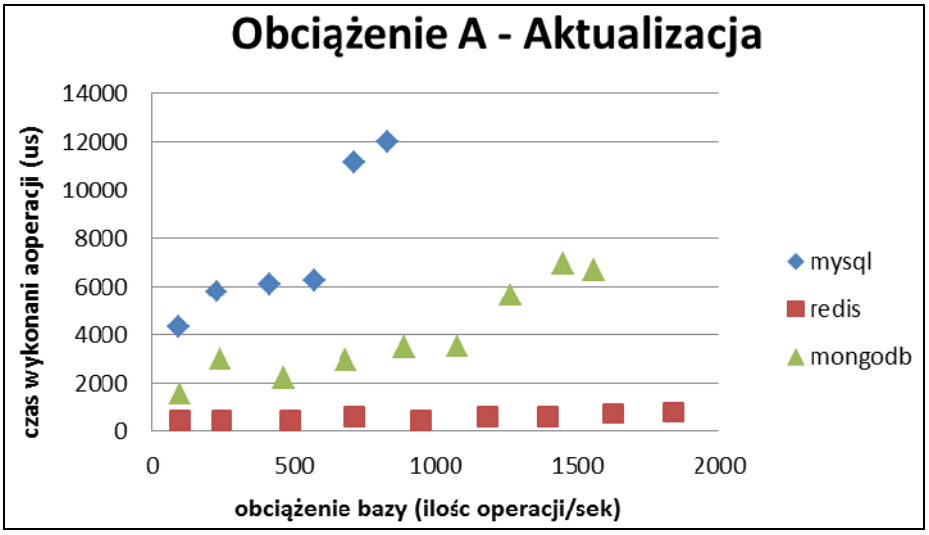
W eksperymencie jest mierzone opóźnienie, czyli czas wykonania danej operacji, w zależności od aktualnie wykonywanych operacji na bazie danych.

[8] B.F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, R. Sears, *Benchmarking Cloud Serving Systems with YCSB*, AMC New York, New York 2010.

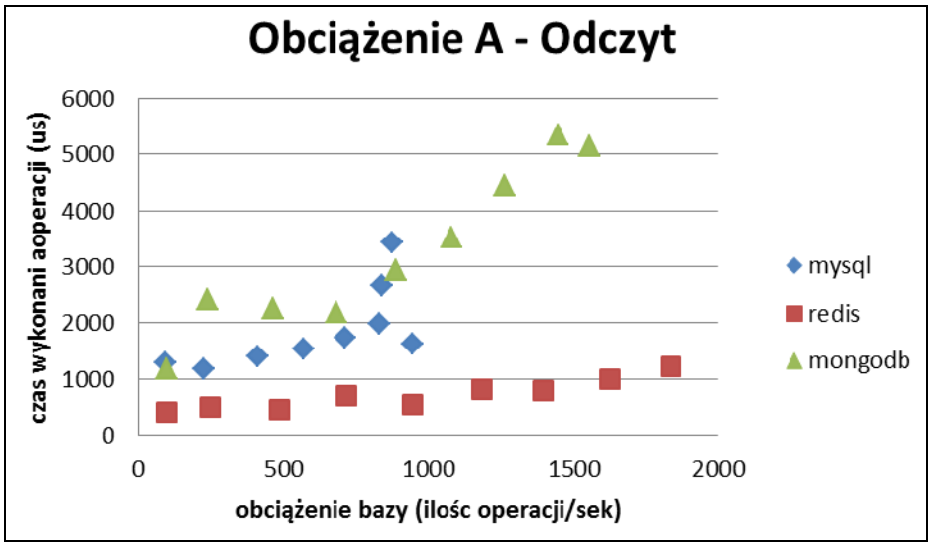
Środowisko testowe

Doświadczenie przeprowadzono w wirtualnym środowisku z użyciem programu VMWare Player. Systemami operacyjnymi bazy danych był Ubuntu Server 14. Parametry techniczne komputera: Procesor: Intel® Core i7-2720QM CPU @ 2.20GHz, RAM: 16GB, system operacyjny Windows 7. Wirtualny serwer bazy danych miał przyporządkowane 512MB pamięci RAM.

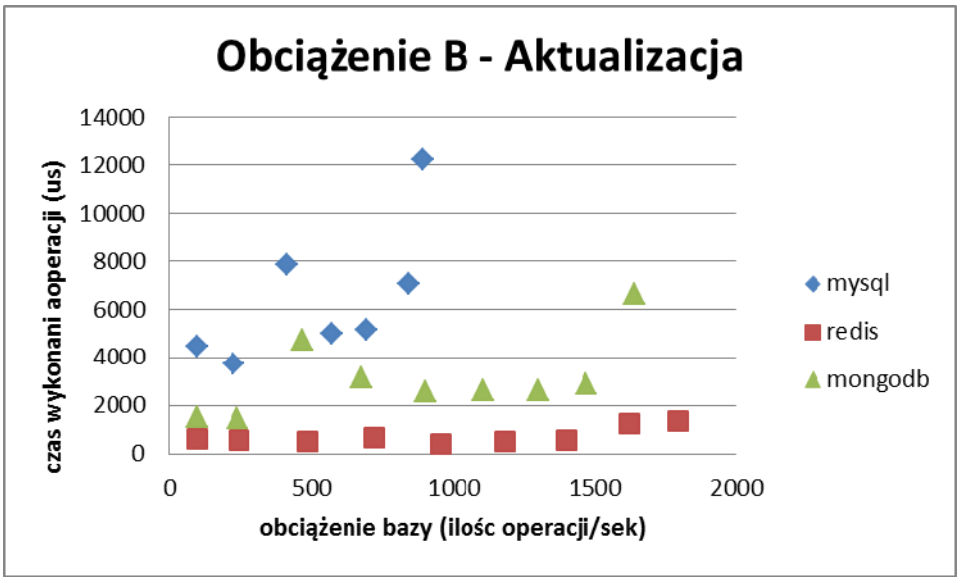
Wyniki eksperymentu



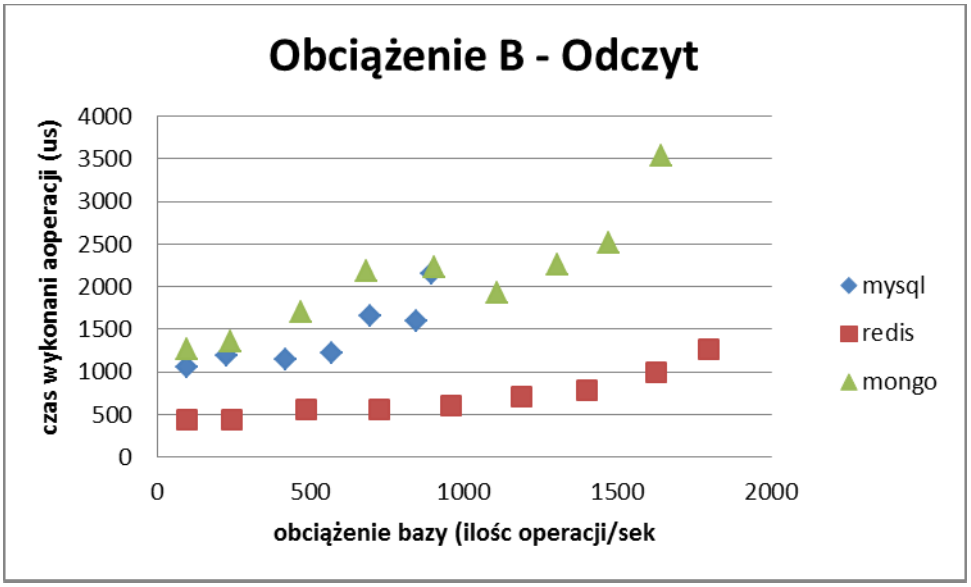
Rys. 1. Czas wykonania operacji, obciążenie A, operacja Aktualizacji



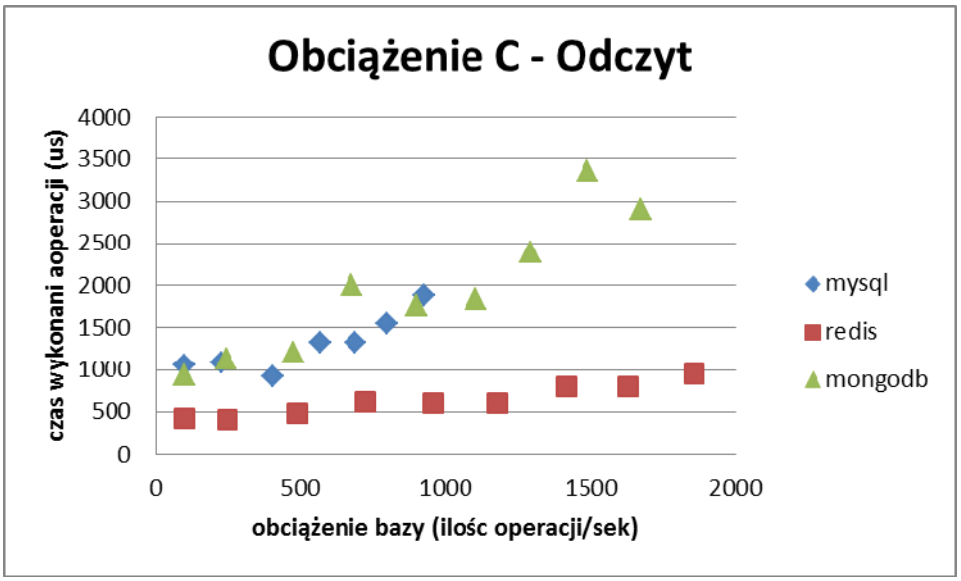
Rys. 2. Czas wykonania operacji, obciążenie A, operacja Odczytu



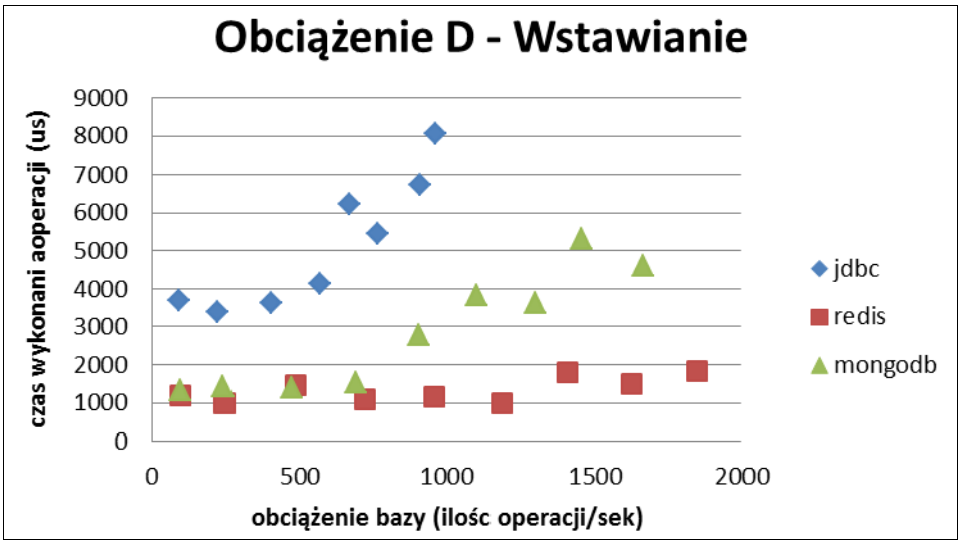
Rys. 3. Czas wykonania operacji, obciążenie B, operacja Aktualizacji



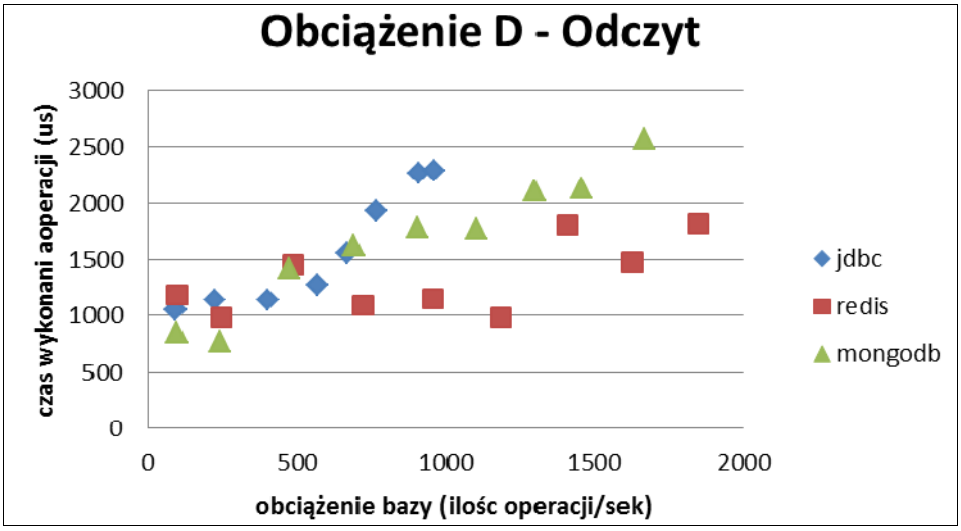
Rys. 4. Czas wykonania operacji, obciążenie B, operacja Odczytu



Rys. 5. Czas wykonania operacji, obciążenie C, operacja Odczytu



Rys. 6. Czas wykonania operacji, obciążenie D, operacja Wstawiania



Rys. 7. Czas wykonania operacji, obciążenie D, operacja Odczytu

Przedstawione wyniki potwierdzają główną tezę ruchu NoSQL, czyli fakt, że relacyjne bazy danych znacznie tracą na wydajności w przypadkach dużego obciążenia bazy danych. Sytuacja taka może wystąpić np. w aplikacji internetowej podczas godzin szczytu. Jak widać na rys. 1, 3 oraz 6, dla obciążenia ok. 500 operacji na sekundę baza MySQL zaczyna działać bardzo wolno, „zatykać się" i wraz ze wzrostem obciążenia czas oczekiwania użytkownika na wykonanie operacji rośnie bardzo szybko. Konsekwencją przeciążenia takiej bazy danych może być jej zawieszenie i wyłączenie, a w konsekwencji utrata informacji, co w niektórych przypadkach może mieć poważne konsekwencje.

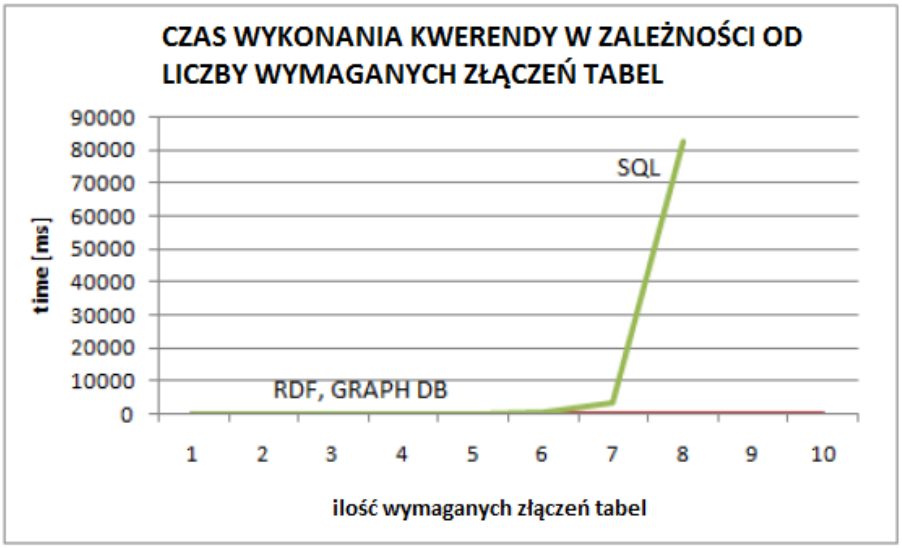
Z analizy wyników otrzymanych dla dokumentowej bazy danych (MongoDB) wynika, że baza ta dużo lepiej zachowuje się w przypadkach znacznego obciążenia (rys. 1, 3 oraz 6). Podczas gdy baza MySQL zaczyna się zawieszać, dokumentowa baza danych tylko trochę traci na szybkości.

Kolejnym, bardzo istotnym wnioskiem wynikającym z przeprowadzonych doświadczeń jest bardzo duża wydajność bazy Redis, czyli nierelacyjnej bazy danych typu klucz-wartość. Baza Redis również zachowuje bardzo dobrą wydajność nawet dla bardzo dużego obciążenia bazy.

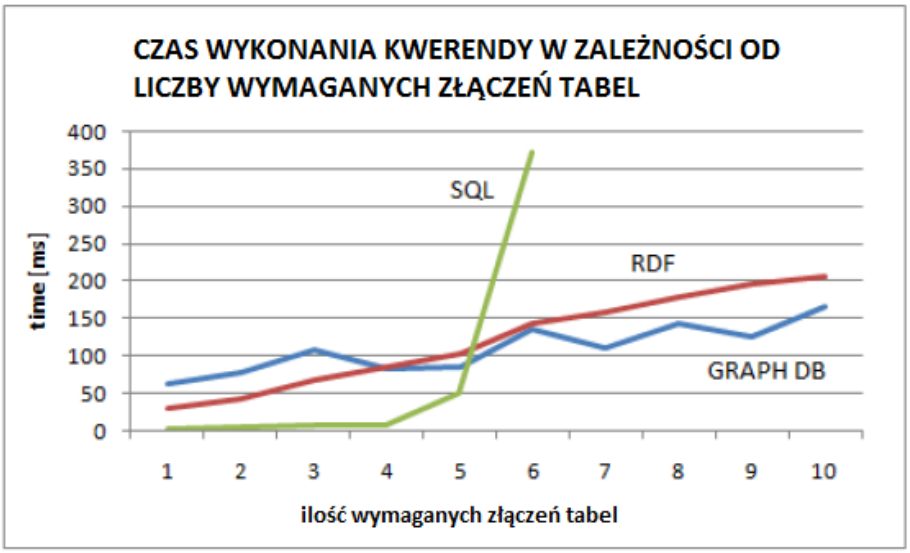
2.2. Porównanie wydajności SZBD dla bazy relacyjnej i grafowej dla zapytań złożonych

Poza badaniem wydajności przedstawionym w poprzednich podpunktach został również przeanalizowany inny aspekt – wydajność zapytań, do których realizacji niezbędne jest wykonanie wielu złączeń tabel. Została porównana wydajność bazy relacyjnej MySQL oraz dwóch baz grafowych – Neo4j oraz

Open RDF[9]. Dokładniejszy opis grafowych baz danych oraz opis przeprowadzonego eksperymentu znajduje się w artykule Autorów[10].



Rys. 8. Czas wykonania kwerendy w zależności od ilości operacji złączenia



Rys. 9. Czas wykonania kwerendy w zależności od liczby wymaganych operacji złączenia, zbliżenie na początek wykresu

[9] L. Strobin, A. Niewiadomski, *New Hierarchical Data Model for Efficiency in NoSQL Databases* [in:] *Computer Methods in Practice*, eds. A. Cader, M. Yatsymirsky, K. Przybyszewski, Computer Science, Exit, Warszawa 2012.

[10] Ibid.

3. Porównanie wydajności systemu rekomendującego opartego na bazie relacyjnej i bazie typu klucz-wartość

W poprzednim punkcie przedstawiono wyniki eksperymentów wydajnościowych przeprowadzonych za pomocą frameworka YCSB. Jak wynika z przedstawionych doświadczeń, zdecydowanie najszybszą bazą danych jest Redis, która jest przykładem nierelacyjnej bazy danych typu klucz-wartość.

Przykładowym zastosowaniem, w którym istotne znaczenie ma szybkość otrzymania odpowiedzi systemu są systemy rekomendujące. Są one wykorzystywane w takich obszarach jak:

- reklamy celowane, np. wyświetlane w sąsiedztwie e-maili,
- proponowanie produktów w sklepach internetowych,
- sugerowanie filmów, muzyki itp.,
- proponowanie miejsc, np. restauracji. Szybkość otrzymania rekomendacji jest szczególnie istotna w przypadku rekomendacji na urządzenia mobilne, np. dla poruszającego się samochodu.

System rekomendujący zaimplementowano w technologii Java z wykorzystaniem frameworka Mahout (mahout.apache.org). Użyto mechanizmu rekomendacyjnego opartego na podobieństwie użytkowników (tzw. *user-based recommendation system*[11]). W takim systemie rekomendacji najpierw jest obliczane podobieństwo między użytkownikami (na podstawie ich ocen danych obiektów), następnie danemu użytkownikowi są rekomendowane obiekty, które podobni do niego użytkownicy oceniają wysoko.

System rekomendacyjny oparty na podobieństwie użytkowników do działania potrzebuje dużego zbioru danych, który zawiera informacje nt. preferencji użytkowników. Do zbudowania systemu użyto zbioru danych udostępnionego przez zespół GroupLens[12], składającego się z 1 mln rekordów, które zawierają dane pochodzące od 6 tys. użytkowników, oceniających wybrane z spośród zbioru 4 tys. filmów. Otrzymany zbiór danych zaimplementowano w trzech testowanych wcześniej bazach danych – typowej relacyjnej bazie MySQL, dokumentowej bazie danych MongoDB oraz nierelacyjnej bazie typu klucz-wartość Redis.

Jak opisano w punkcie 1, bazy te posiadają zupełnie różny model danych. Jak wspomniano wcześniej, dane do systemu rekomendacji składają się z trójek:

- ID użytkownika – ID filmu – ocena

[11] M. Pazzani, *Content based Recommendation Systems* [in:] *The Adaptive Web: Methods and Strategies of Web Personalization*, eds. P. Brusilovsky, A. Kobsa, W. Nejdl, Springer Verlag, Berlin Heidelberg 2007.

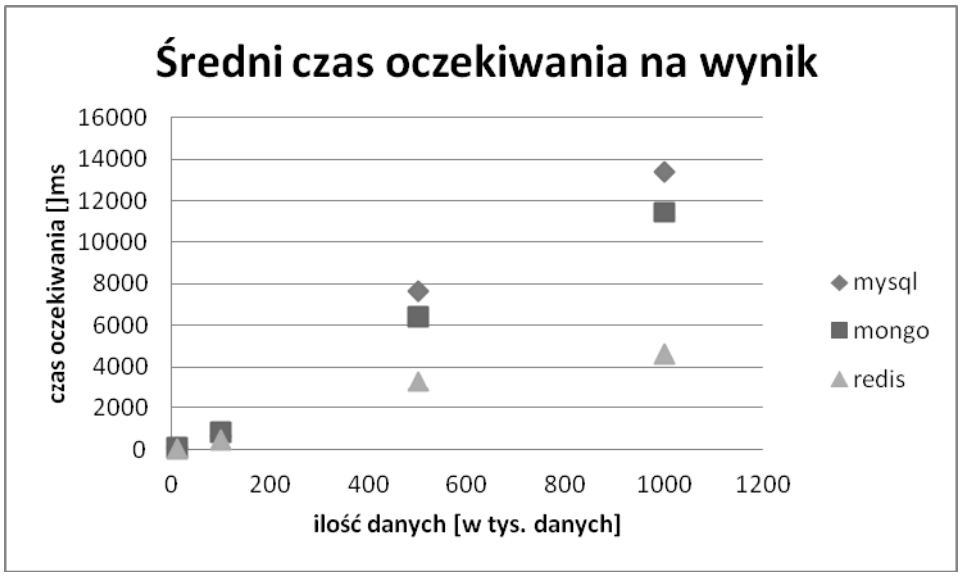
[12] GroupLens, 2010, http://www.grouplens.org.

W modelu relacyjnym dane te zostały zgromadzone w jednej tabeli, składającej się z trzech kolumn. Na każdej kolumnie zostały założone indeksy. W bazie typu klucz-wartość dane te zostały natomiast zamodelowane w zupełnie inny sposób:

- ID Użytkownika -> zbiór par (ID Filmu – ocena)
- ID Film -> zbiór par (ID Użytkownika – ocena)

Jak widać, przy takim zamodelowaniu danych pojawia się redundancja – ocena, jaką przyznał dany użytkownik danemu filmowi jest zapisana w bazie dwukrotnie.

Przeprowadzono badanie dla różnych ilości danych: 10 tys. trójek, 100 tys. trójek, 500 tys. trójek oraz 1 mln trójek, dla każdego rodzaju bazy danych. Poniżej na rys. 10, przedstawiono średni czas oczekiwania na rekomendację w zależności od wielkości bazy danych.



Rys. 10. Czas oczekiwania na rekomendację w zależności od wielkości bazy dla różnych baz danych

Jak wynika z przedstawionych wyników, zastosowanie nierelacyjnej bazy Redis znacznie zwiększyło szybkość działania programu rekomendującego. Warto zauważyć, że zysk ze stosowania takiej bazy rośnie wraz ze wzrostem ilości danych do przetworzenia. W przypadku 1 mln rekordów w bazie stosowania bazy Redis przyspiesza działanie programu ok. 3-krotnie, co jest bardzo istotnym przyspieszeniem. Jak widać na wykresie, dla 2-krotnie mniejszej bazy, zysk jest już 2-krotny. Dla jeszcze mniejszych zbiorów danych, zysk bazy Redis jest natomiast jeszcze mniejszy, aczkolwiek cały czas występuje.

Wydajność zastosowanej dokumentowej bazy danych, MongoDB, jest zbliżona do relacyjnej bazy MySQL. Oferuje tylko ok 10% zysku czasowego. Warto jednak zauważyć, że zgodnie z wynikami poprzednich doświadczeń dokumentowa baza MongoDB znacznie lepiej skaluje się wraz z rosnącym obciążeniem.

Podsumowanie

Jak wynika z przeprowadzonych doświadczeń, w niektórych aspektach nierelacyjne bazy danych mają znaczną przewagę nad typowymi, relacyjnymi bazami danych. Przeprowadzono wiele doświadczeń wydajnościowych z wykorzystaniem benchmarku, jak również porównano szybkość działania systemu rekomendującego.

Porównując bazę relacyjną (MySQL) i dokumentową (MongoDB), należy zwrócić uwagę na kila aspektów. Wykresy na rys. 2, 4 i 6 wykazują, że dla operacji odczytu wydajność baz jest porównywalna. Wykresy na rys. 1, 3 i 6 wyraźnie wykazują z kolei przewagę bazy dokumentowej dla operacji zapisu. Przewaga rośnie wraz ze wzrostem obciążenia bazy danych, tj. ilości wykonywanych operacji na sekundę. Jak opisano wcześniej, może mieć to znaczenie szczególnie dla aplikacji internetowych, w których możliwe jest podłączenie wielu użytkowników jednocześnie. Biorąc pod uwagę model danych baza dokumentowa oferuje łatwość w modyfikacji struktury danych oraz łatwą możliwość reprezentowania złożonych wartości atrybutu (np. lista). Wadą baz dokumentowych jest brak możliwości wykonywania złączeń tabel, przez co wykonywanie złożonych zapytań wymaga przetwarzania w kodzie aplikacji. Jeśli chodzi z kolei o szybkość i wydajność baz danych, to zdecydowaną przewagę ma baza danych typu klucz-wartość (Redis). W tym przypadku operacje odczytu i zapisu są zdecydowanie najszybsze, jak również baza ta najlepiej zachowuje się w przypadku jej znacznego obciążenia, co jest przedstawione na wykresach na rys. 1-6. Wadą tego rozwiązania jest bardzo prosty model danych i bardzo ograniczone możliwości operacji na danych bezpośrednio w bazie danych, przez co jest wymagane przetwarzanie w kodzie aplikacji.

Wyniki otrzymane za pomocą benchmarku zostały potwierdzone poprzez zaimplementowanie aplikacji rekomendującej. Aplikację oparto na omawianych trzech typach baz danych i porównano szybkość otrzymania rekomendacji. Przeprowadzone doświadczenie wykazało, że system pracujący na podstawie bazy typu klucz-wartość działa 3 razy szybciej niż w przypadku bazy relacyjnej, dla zbiorów danych składających się z ok. 1 mln krotek. W przypadku więk-

szych zbiorów danych oraz większego obciążenia bazy danych, zysk ze stosowania tej technologii baz danych jest jeszcze większy.

Podsumowując, w artykule eksperymentalnie wykazano przewagi, jakie mają nierelacyjne bazy danych nad typowymi rozwiązaniami. Warto zwrócić uwagę na nierelacyjne bazy danych szczególnie w przypadku aplikacji internetowych, w których szybkość działania jest cechą kluczową.

Literatura

Chang F., *BigTable: A Distributed Storage System for Structured Data*, OSDI'06 Proceedings of the 7th USENIX Symposium on Operating System Design and Implementation – Vol. 7, 2006.

Cooper B.F., Silberstein A., Tam E., Ramakrishnan R., Sears R., *Benchmarking Cloud Serving Systems with YCSB*, AMC New York, New York 2010.

DeCandia G., *Dynamo: Amazon's Highly Avaiable Key-Value Store*, SOSP'07 Proceedings of twenty-first ACM SIGOPS symposium on Operating System Principles, 2007.

Ellis E., *NoSQL Ecosystem*, 2010, http://www.rackspacecloud.com/blog/2009/11/09/nosql-ecosystem.

GroupLens, 2010, http://www.grouplens.org/.

http://mongify.com/

http://www.grouplens.org/

Jing Han, Haihong E., Guan Le, Jian Du, *Survey on NoSQL database*, „Pervasive Computing and Applications (ICPCA)" 2011.

Kalek A., *Mongify*, 2011, www.mongify.com.

Kruber N., *Scalaris: Scalable Web Applications with a Transactional Key-Value Store*, 2012, http://2012.berlinbuzzwords.de/sites/2012.berlinbuzzwords.de/files/slides/Scalaris-nkruber-bbuzz12.pdf.

Nayak A., Poriya A., Poojary D., *Type of NOSQL Databases and its Comparison with Relational Databases*, „International Journal of Applied Information Systems" 2013, Vol. 5, No. 4.

Pazzani M., *Content based Recommendation Systems* [in:] *The Adaptive Web: Methods and Strategies of Web Personalization*, eds. P. Brusilovsky, A. Kobsa, W. Nejdl, Springer Verlag, Berlin Heidelberg 2007.

Strobin L., Niewiadomski A., *New Hierarchical Data Model for Efficiency in NoSQL Databases* [in:] *Computer Methods in Practice*, eds. A. Cader, M. Yatsymirsky, K. Przybyszewski, Computer Science, Exit, Warszawa 2012.

SUBSTANTIAL INCREASE IN APPLICATION PERFORMANCE THROUGH THE USE OF NoSQL DATABASE TECHNOLOGIES

Summary

The aim of this article is to experimentally verify higher efficiency of NoSQL databases over typical relational databases. After short introduction to NoSQL, document databases and key-value stores, several efficiency tests were conducted, with the use of YCSB testing benchmark. Results show that the higher the load on the database, the higher the efficiency benefit of using NoSQL, especially for insert and update operations, and for key-value stores. Afterwards, the database efficiency is compared in a real application – a user-based recommendation engine. In this case the usage of NoSQL can boost the application speed at least three times, but the predicted efficiency increase under greater loads is much higher.