MySQL
Optymalizacja przez kompilację.

Paweł Szołtysek

Spis treści

Streszczenie

Dokument ten przedstawia problem optymalizacji szczególnego modelu relacyjnego bazy danych - strukturalnego języka zapytań MySQL. Skupia się na jednej z metod - optymalizacji przez kompilację. W dokumencie został zamieszczony krótki wstęp teoretyczny, a także ogólnodostępne w sieci wyniki takiej optymalizacji.

1 Wstęp

Systemy bazodanowe w dzisiejszym świecie są jednym z najważniejszych zastosowań komputerów. Śmiało można stwierdzić, że na ich istnieniu opiera się olbrzymia ilość projektów informatycznych, a także znaczna większość

witryn w sieci Internet.
Z tego powodu zadanie optymalizacji takich systemów jest bardzo ważne i wykorzystywane przez wiele różnorodnych osób i organizacji, a jakość jego wykonania może nawet przełożyć się na sukces lub porażkę całych projektów.
Opracowanie to opiera się na systemie MySQL z kilku względów.

- Jest to bodaj najczęściej wykorzystywany system bazodanowy na świecie.
- Jest to system typu open-source, za jego wykorzystywanie do celów niekomercyjnych nie trzeba płacić.
- Wyróżnia go doskonała wydajność oraz skalowalność.
- Wykorzystuje się go wszędze tam, gdzie duże znaczenie ma prędkość przetwarzania danych.

Należy jednak dodać, że większość rzeczy poruszonych poniżej może być zastosowana do dowolnych innych systemów dostępnych na dzisiejszym rynku.

2 Metody optymalizacji a tworzenie baz danych

Optymalizacja baz danych jest procesem długotrwałym, tzn. powinno się zwracać uwagę na wydajność na każdym kroku ich tworzenia. Generalnie proces życia pojedynczej bazy danych można podzielić na trzy etapy:

Projektowanie

Implementacja

Użytkowanie oraz wprowadzanie modyfikacji

Każdy z nich ma własne możliwości optymalizacyjne, które prowadzą do minimalizacji np. miejsca zajmowanego przez strukturę lub czasu wykonywania poszczególnych zapytań czy transakcji.

3 Optymalizacja przez kompilację

Pierwszym krokiem implementacji jest odpowiedni wybór silnika bazy danych, po czym dokonywana jest jego instalacja.
W procesie zwiększania wydajności systemu MySQL, konfiguracja serwera i

tuning kompilacji są o tyle ważne, że nie ma uniwersalnych parametrów, które pozwoliłyby uznać je za bardzo dobre we wszystkich przypadkach użycia (dobrze pokazuje to [1]). Dzięki temu uzyskano też wysoką skalowalność.
Cechą omawianego systemu jest otwarty kod, co pozwala na dowolne modyfikowanie parametrów kompilacji, która będzie wykonana i zainstalowana na maszynie. Optymalizacja przez kompilację opiera się na odpowiednim konfigurowaniu ich w taki sposób, który pozwoli na osiągnęcie lepszych wyników dla szczególnego przypadku danych oraz systemu operacyjnego. Taki rodzaj optymalizacji daje (jak później przedstawimy) znaczne efekty.
Wystarczy zastosować analogię do taśm produkcyjnych tych samych samochodów, gdzie jedna z nich będzie produkowała samochody niechlujnie, ale w dużej ilości, a druga z nich będzie ich robiła mniej, ale każdy z samochodów będzie dogłębnie testowany.
Podobnie jest z samymi kompilatorami. Jeden z nich będzie wykonywał swoją pracę lepiej, a inny znów gorzej - należy więc znaleźć taki, który będzie stosunkowo dobry. W [3] pada nawet stwierdzenie, że *By just using a better compiler and/or better compiler options you can get a 10-30% speed increase in your application*, co zwraca uwagę także na ewentualne ustawienia kompilatora.
Dobra ilustracja tego faktu znajduje się w [1], gdzie możemy znaleźć takie porównanie:

Rysunek 1: Różnica w jakości kompilacji między Sun Studio 10 a Sun Studio 11 dla MySQL 5.0.15 DBT2 na Solaris 10.

z którego wprost wynika, że wydajność systemu wzrosła o 13% tylko dzięki użyciu innego (nowszego) kompilatora.

3.1 Wybór kompilatora

Kiedy mówimy o kompilatorze C++ dla linuxa mamy zwykle na myśli `gcc` (http://www.gnu.org/software/gcc/gcc.html). Jednakże powinniśmy mieć na względzie także inne dostępne na rynku. Szczególnie, jeśli pracujemy na procesorze Pentium, zalecanym jest używanie `pgcc`, czyli opartego na wcześniejszym `gcc` kompilatora zoptymalizowanego na działanie właśnie na tych CPU. Według [5] tylko z tego powodu (przy tych samych ustawieniach) możemy uzyskać wzrost wydajności rzędu 10%.
Należy też zwrócić uwagę na kompilator `icc`, który w testach [2] potrafił osiągnąć ponad dwuipółkrotnie lepszą efektywność (dla szczególnych typów procesorów) względem `gcc`.

3.2 Dobór odpowiednich parametrów serwera MySQL

Jedną z zalet optymalizacji przez kompilację (i głównym powodem faktycznego wpływu jej na późniejszy efekt w postaci wzrostu szybkości) jest możliwość doboru odpowiednich ograniczeń tworzonego serwera MySQL.

3.2.1 Ograniczenie ilości character sets

MySQL, wersji 5.0.54-log, oferuje domyślnie 36 zestawów znaków w 128 różnych dialektach. To dużo.
Jednak fakt jest taki, że zwykle w obrębie jednej bazy danych nie potrzebujemy więcej niż jednego dialektu konkretnego zestawu znaków. Pozostałe są więc całkowicie rezydentne, spowalniając i zabierając niepotrzebnie zasoby. Można to łatwo ograniczyć poprzez przełącznik

```
-with-extra-charsets=none
```

który w zaprezentowanej formie zainstaluje tylko domyślny zestaw znaków `latin1`.

3.2.2 Statyczna kompilacja mysqld

Statyczna kompilacja mysqld (tj. taka, w której biblioteki nie są współdzielone) także wpływa na zwiększenie się wydajności systemu. Aby to zrobić, należy wpisać

```
-with-mysqld-ldflags=-all-static
```

dzięki temu wynik będzie zajmował nieco więcej miejsca, ale będzie pracował znacznie szybciej (15% według [3]).

3.2.3 Pomijanie wyjątków C++

MySQL nie używa wyjątków C++, więc nie muszą być one brane pod uwagę podczas procesu kompilacji.

```
-fno-exceptions
```

[4] zauważa, że taka zmiana daje znaczny wzrost osiągów systemu.

3.2.4 Nieużywanie wskaźników frame

Kompilacja, która nie używa *frame pointers*

```
-fomit-frame-pointer -ffixed-ebp
```

jest według [3] kilka procent szybsza używając kompilatora `gcc` na Linux-x86.

3.3 Przykładowa konfiguracja kompilera

Przeglądając sieć można kilka razy zobaczyć następującą konfigurację kompilera jako zalecaną:

```
CFLAGS="-O6 -mpentiumpro -fomit-frame-pointer" CXX=gcc CXXFLAGS="-O6
-mpentiumpro -fomit-frame-pointer -ffixed-ebp -felide-constructors
-fno-exceptions -fno-rtti" ./configure --prefix=/usr/local
--enable-assembler --with-mysqld-ldflags=-all-static --disable-shared
-with-extra-charsets=none
```

Jest ona ogólnie używana w celu zwiększenia prędkości użytkowania całego systemu. Jest to propozycja dla kompilera `gcc` dla systemu linux.
Znaczenie poszczególnych przełączników można znaleźć w [6].
Dla systemu operacyjnego Solaris 10 optymalizację bazy danych (dla tabel typu InnoDB) można znaleźć w [1].

3.4 Pułapki kompilacji MySQL

Po wykonaniu kompilacji należy dokładnie zbadać, czy została ona wykonana poprawnie i czy na pewno jest optymalna (tj. czy zawiera dokładnie tyle cech ile potrzebujemy). Przykładem są tutaj silniki tabel. Zaproponowana w 3.3 konfiguracja będzie obsługiwała tylko cztery podstawowe typy silników - CSV, MRG_MYISAM, MEMORY i MyISAM. I tak, aby dodać silnik InnoDB, należy przy kompilacji dopisać

```
--with-plugins=innodatabase
```

do MySQL 5.1, lub

`--with-plugins=innodb`

dla MySQL 5.0.54. Wszystkie dostępne silniki zostaną wkompilowane w system przy użyciu

`--with-plugins=max`

ale z oczywistych względów wydajnościowych nie jest to zalecane. Można też użyć przełącznika

`-sql_mode=NO_ENGINE_SUBSTITUTION`

który nie zamieni standardowych silników.

4 Znaczenie optymalizacji przez kompilację

Analiza teoretyczna potwierdza, że kompilacja serwera MySQL potrafi znacznie podnieść szybkość jego działania. Jest to więc bardzo ważny krok przy jego optymalizacji.
Należy jednak pamiętać, że jest to tylko jedna bitwa w całej wojnie o wydajność bazy danych. Aby osiągnąć pełen sukces, trzeba mieć wzgląd na osiągi na każdym z etapów tworzenia takiej bazy.

Literatura

[1] Luojiia Chen: *MySQL InnoDB Performance Tuning for the Solaris 10 OS*,
http://developers.sun.com/solaris/articles/mysql_perf_tune.html

[2] Peter Zaitsev: *Improving MySQL Server Performance with Intel C++ Compiler*,
http://www.mysqlperformanceblog.com/files/presentations/LinuxWorld2005-Intel.pdf.

[3] *MySQL 5.0 Reference Manual*,
http://dev.mysql.com/doc/refman/5.0/en/index.html.

[4] *MySQL Presentations: Optimizing MySQL*,
http://dev.mysql.com/tech-resources/presentations/presentation-oscon2000-20000719/.

[5] *Optimizing MySQL*,
http://www.devshed.com/c/a/MySQL/Optimizing-MySQL/.

[6] *Using the GNU Compiler Collection*,
http://gcc.gnu.org/onlinedocs/gcc-4.3.0/gcc.pdf.