

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Modelování a simulace - 6. Počítačové služby

Porovnávání SQL a JAVA přístupů do databáze

autor?28. ledna 2018

Obsah

| | | |
|----------|--|----------|
| 1 | Úvod | 1 |
| 2 | Zdroje faktů | 1 |
| 2.1 | Průběh sběru dat | 1 |
| 2.1.1 | Přesnost vyhledávání | 2 |
| 2.1.2 | RAM paměť | 2 |
| 2.1.3 | Vygenerování databáze | 3 |
| 2.1.4 | Automatizace pomocí BASH | 3 |
| 2.1.5 | Naměření hodnot JAVA | 4 |
| 2.1.6 | Zpracování naměřených hodnot | 4 |
| 2.2 | Ověření funkčnosti modelu | 4 |
| 2.3 | Petriho síť | 4 |
| 3 | Závěr | 5 |

1 Úvod

V této práci je řešen projekt do předmětu **IMS - Modelování a simulace** [1] vyučovaném na Fakultě informačních technologií Vysokého učení technického v Brně [2]. Konkrétně se jedná o zadání **6. Počítačové služby** [3].

Tato práce se věnuje problematice doby vyhledávání v databázi. Zaměříme se na porovnávání přístupu do databáze výhradně přes SQL dotazy a přístupu do databáze spojeného s cacheováním (jednotlivých řádků vyhledávané tabulky) na počítači klienta.

V našich experimentech se zaměřujeme zjištění za jakých podmínek je efektivnější pro vyhledávání použít spíše databázový server a nebo ve veškerých datech vyhledávat až na straně klienta. Vzhledem k tomu, že na tyto časy hraje roli několik faktorů výběr nemusí být na první pohled hned jasný. V kapitole Experimenty (viz. Experimenty) jsou sepsány jednotlivé krajní i více obecné (realističtější) případy při práci s databází.

2 Zdroje faktů

Jako model jsme si vybrali databázi Postgresql ve verzi 9.5.10 [6]. Pro přístup do této databáze jsme zvolili naprogramování aplikace v jazyce JAVA [7] ve verzi JDK-1.8.0_151 [8], ve které jsme si naprogramovali komunikaci se serverem. Programy pro sběr dat z této komunikace běželi na virtuálním stroji Ubuntu 16.04.3 LTS [9] a samotné posílání jednotlivých dotazů bylo zautomatizované pomocí scriptu psaném v GNU Bash version 4.3.48(1)-release (x86_64-pc-linux-gnu) [10].

2.1 Průběh sběru dat

Pro přístup k datům z databáze a měření doby přístupu k datům, jsme se rozhodli, že bude vhodné, aby se vyhledávací dotazy vytvářeli na virtuální počítači odděleného od virtuálního počítače s databázovým serverem. Vytvořili jsme tedy 2 virtuální počítače s těmito parametry:

| VM1SERVER | |
|-----------|-------------------|
| CPU | 2 jádra - 4,2 GHz |
| RAM | 2048 MB |
| SSD | 25 GB |
| OS | Ubuntu - 16.04.3 |

| VM2CLIENT | |
|-----------|-------------------|
| CPU | 2 jádra - 4,2 GHz |
| RAM | 8192 MB |
| SSD | 25 GB |
| OS | Ubuntu - 16.04.3 |

Parametry byli vybrány, tak aby splňovali minimální systémové požadavky a zároveň bylo co nejvíce místa pro nacachování prohledávaných tabulek.

Tyto parametry jsme zvolili, tak aby splňovali požadavky operačního systému Ubuntu [12], databázového serveru Postgresql [13] a zároveň aby se na straně klienta spustila JAVA s námi vybranými velikostmi RAM pro měření. Virtuální diskové prostory byli fyzicky uloženy na SSD disku, který během testování nepřesáhl 20% zatížení.

OS parametry Operační systém ubuntu [9] jsme si vybrali z důvodu jednoduché instalace jednotlivých aplikací potřebných pro tento sběr dat, nízké náročnosti na hardwarové požadavky a jednoduchou obsluhu. Jako databázový systém nám posloužil PostgreSQL [6]. Tento systém jsme zvolili z důvodu

jednoduché instalace, nízkých nároků na hardware a The PostgreSQL Licence (mirně modifikované Open Source licence) [14]. Při výběru s jakou databází budeme pracovat jsme si nezvolili Oracle [15] z důvodu, že již není volně k dispozici pro komerční použití a naše výsledky by nebyli dostatečně využitelné.

Na prvním virtuálním počítači běžel server (dále jen VM1SERVER) s databází PostgreSQL a měl za úkol zpracovávat přijaté SQL dotazy a odpovídat na ně. Druhý virtuální počítač znázorňující klienta (dále jen VM2CLIENT) se postupně připojoval na databázový server a posílal dotazy.

Na VM2CLIENT tedy běžel BASH [10] script *functions.sh*, který automaticky spouštěl námi naprogramované Javové dotazy z */dist/testApp.jar*.

V rámci automatických testů se také před každým měřením musel zaslat požadavek pro vymazání cache paměti v databázi na straně serveru, toto je vyřešeno připojením přes OpenSSH rozhraní (OpenSSH_7.2p2 Ubuntu-4ubuntu2.4, OpenSSL 1.0.2g 1 Mar 2016) [11] *sshclear.sh* připojením se na VM1SERVER a odtud zavoláním scriptu *clearcache.sh*.

2.1.1 Přesnost vyhledávání

Jako hodnotu pro vyhledávání jsme si zvolili index (od 1 do velikosti tabulky po 1). Rozhodli jsme se filtraci provádět podle začátku řetězce jeho hodnoty. Tato filtrace byla zvolena z toho důvodu, že se nejvíce podobá přístupu do databáze k vyhledání určité položky, aneb jak s databází pracuje normální uživatel. Při práci s databází byla tato filtrace prováděna příkazem LIKE 'prefix%' [19] a v JAVA `startsWith(String prefix)` [18] (dále se na obě funkce budeme zároveň odkazovat jako na pseudofunkci LIKEE(prefix)).

Příklady LIKEE() nad řetězci:

LIKEE("AH")nad" AHOJ"vyjdejakoTrue

LIKEE("1")nad"10"vyjdejakoTrue

LIKEE("0")nad"10"vyjdejakoFalse

LIKEE("1")nad"1"vyjdejakoTrue

Tudíž se dá jednoduše určit v našem případě, že když pracujeme s indexovanou tabulkou nad jejím indexem u každého řádku zjistíme jestli je začátkem a na konci vrátíme hodnotu pro kolik řádků to platí:

LIKEE("1")100tabulka => 12

LIKEE("1")1000tabulka => 111

LIKEE("10")100tabulka => 2

LIKEE("10")500tabulka => 11

LIKEE("100")500tabulka => 1

LIKEE("100")30000tabulka => 111

LIKEE("1000")500tabulka => 0

2.1.2 RAM paměť

Pro výběr s jakými hodnoty RAM paměti budeme pracovat jsme byli limitováni fyzickým hardwarem, proto jsme na VM1SERVER usoudili, že nebude potřeba větší než minimální Systémem a Databází požadovaná viz: **OS parametry 2.1**. Pro VM2CLIENT jsme měli 8GB paměti. Nyní stačilo vyhledat pro jaké RAM paměti nám JAVA běžící na VM2CLIENT dovolí s maximální velikostí tabulek. Zjistili jsme, že pro JAVA při paměti 512MB dokáže nad našimi tabulkami (ve formátu viz: 2.1.3) pracovat s maximálně 150 000 tabulkami. Na konec jsme zvolili RAM 1024, 2048 a 4096. Tyto RAM paměti bez problému pojmu i naši největší zvolenou tabulku (250 000).

2.1.3 Vygenerování databáze

Pro generaci potřebných dat jsme si vybrali csv generátor [16] jednotlivé řádky jsme se rozhodli selectovat pomocí indexu("seq"), bylo ovšem zapotřebí, aby se tabulka podobala tabulkám se kterými se pracuje v reálném životě, proto jsme si zvolili, že každý řádek bude obsahovat hodnoty pro tyto sloupce:

*"seq, first, last, age, street, city, state, zip,
dollar, pick, date, email, digid, latitude, longitude,
pick2, string, domain, float, ccnumber, bool, yn"*

Například:

*1, Jesse, Watts, 51, HusfoTerrace, Livemil, HI, 75091, \$2932.33,
YELLOW, 7/13/1993, sajuvubug@uj.net,
147889110758, 12.63144, -166.12131, UP, W4P9nY0yubdKsQu)sxI,
boto.co.uk, -275370638258.9952, 6304284025402256, true, N*

Velikost jednotlivých řádků tabulek by neměla ovlivnit poměr časů vyhledávání, ale slouží k tomu, aby jsme vygenerovali vstupní data k simulaci ve kterých uvidíme větší hodnoty s menší pravděpodobností vzniklých nepřesností.

Tyto data jsme umístili na VM1SERVER, tak že jsme na něm spustili databázi a připojili jsme se na ní pomocí PG ADMIN 3 [17] a nahráli zde tabulky ze kterých jsme plánovali získávat data.

Tabulky ze kterých jsme zkoumali data měli velikosti:

| TABLES |
|--------|
| 100 |
| 500 |
| 1000 |
| 2000 |
| 5000 |
| 7500 |
| 10000 |
| 20000 |
| 30000 |
| 50000 |
| 75000 |
| 100000 |
| 200000 |
| 250000 |

Původně jsme plánovali pracovat s tabulkami velkými až 10 000 000, to jsme ovšem velice rychle zavrhlí z důvodu velké časové náročnosti výpočtů.

2.1.4 Automatizace pomocí BASH

Z bash scriptu byl volán JAVA program [link] postupně po jednom tak aby byla provedena kombinace všech vstupních parametrů z předchozí sekce(RAM, TABLES, LIKEE). Naměřené hodnoty jsme ukládali do formátu csv. Při

2.1.5 Naměření hodnot JAVA

JAVA DELALA TO A TO A VYHODILA TO JAKO VYSLEDEK V TAK A TAK 6 DOB VYHODILA A TY SE ZPRACOVALI

Pro získávání dat jsme vytvořili program v jazyce JAVA (viz. příložený kód. . .). V tomto programu se připojujeme na databázový server. Zkoumáme zde reálný systém na zaklade hodnot nad nim .

BLBLBAALBLAL

2.1.6 Zpracování naměřených hodnot

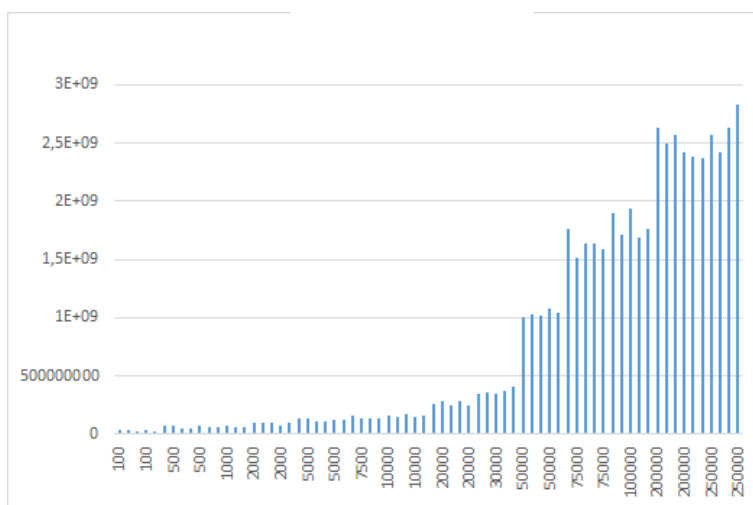
Výstupní *csv* soubory jsme si převedli do formátu *xlsx* kde jsme nad jednotlivými časy vytvářeli grafy. Na obrázku číslo 1 můžete vidět dobu vytváření objektů všech prvků z tabulky na základě její velikosti. Z grafu je jasné vidět, jak se jednotlivé časy mění při změně velikosti tabulky.

Normální rozložení Potřebovali jsme na základě těchto hodnot vytvořit funkce podle který by jsme mohli určit přibližnou hodnotu daného prvku v určitém bodě, u který nemáme hodnoty naměřené. Rozhodli jsme,

JAK????????????????????????????????????

že veškeré grafy jsou normálního rozložení. U každého jednotlivého grafu bylo zapotřebí zvážit zda není vhodné ho rozdělit na více intervalů, kde by vytvořená funkce počítala přesněji. Hodnoty ze zvoleného intervalu jsme přes [20]

z nich vytvořili funkce, které jsme si poté zkontrolovali zda opravdu nabývají podobných hodnot. Přes [link] jsme si zapsali průměrnou odchylku jednotlivých bodů od vytvořené funkce. Rozhodli jsme se vytvořit na základě střední hodnoty a rozptylu



Obrázek 1: LIKEE-1-CREATE ALL - Vertikální osa značí čas v nanosekundách a horizontální značí velikost tabulky.

2.2 Ověření funkčnosti modelu

Validita modelu byla ověřována porovnáváním dat, které jsme vygenerovali s daty reálnými.

2.3 Petriho síť

Obrázek 2: Petriho síť

Obrázek 3: DBModel a JavaModel

3 Závěr

//RAM STUFFFF

Potom nějaká věta o tom, že v naměřených datech, jsme zjistili, že nakonec měla RAM jen (určitý vliv) a to jen u vybírání celých nebo jen likee 1 filtraci

Pozn. Pokud používáme přesné filtrování pomocí databáze, tak bychom mohli i pro větší tabulky zvolit menší RAM.

//STUFFFF

procházením po jednotlivých řádcích. Následné vygenerované data analyzujeme a modelujeme v simlibu[4, 5]

Literatura

- [1] IMS - Modelování a simulace: [online]. [vid. 2017-12-06].
URL <<http://www.fit.vutbr.cz/study/course-1.php.cs?id=12167>>
- [2] Fakulta informačních technologií Vysokého učení technického v Brně: [online]. [vid. 2017-12-06].
URL <<http://www.fit.vutbr.cz/>>
- [3] Zadání č.6: [online]. [vid. 2017-12-06].
URL <<http://perchta.fit.vutbr.cz:8000/vyuka-ims/42>>
- [4] Simlib: [online]. [vid. 2017-12-06].
URL <<http://www.fit.vutbr.cz/~peringer/SIMLIB/>>
- [5] Simlib-3.04-20171004.tar.gz: [online]. [vid. 2017-12-06].
URL <<http://www.fit.vutbr.cz/~peringer/SIMLIB/source/>>
- [6] PostgreSQL: [online]. [vid. 2017-12-06].
URL <<https://www.postgresql.org/?&>>
- [7] Java: [online]. [vid. 2017-12-06].
URL <<https://java.com/en/download/>>
- [8] JDK-1.8.0_151: [online]. [vid. 2017-12-06].
URL <<http://www.oracle.com/technetwork/java/javase/8u151-relnotes-3850493.html>>
- [9] Ubuntu 16.04.3 LTS: [online]. [vid. 2017-12-06].
URL <<http://fridge.ubuntu.com/2017/08/05/ubuntu-16-04-3-lts-released/>>
- [10] GNU Bash: [online]. [vid. 2017-12-06].
URL <<https://www.gnu.org/software/bash/>>
- [11] OpenSSH: [online]. [vid. 2018-01-05].
URL <<https://man.openbsd.org/ssh.1>>
- [12] Ubuntu system requirements: [online]. [vid. 2018-01-05].
URL <<https://help.ubuntu.com/community/Installation/SystemRequirements>>
- [13] PostgreSQL system requirements: [online]. [vid. 2018-01-05].
URL <https://www.commandprompt.com/blog/postgresql_minimum_requirements/>
- [14] The PostgreSQL Licence: [online]. [vid. 2018-01-05].
URL <<https://opensource.org/licenses/postgresql>>
- [15] Oracle Database Server: [online]. [vid. 2018-01-05].
URL <<https://www.oracle.com/database/index.html>>
- [16] Csv generator: [online]. [vid. 2018-01-05].
URL <<http://www.convertcsv.com/generate-test-data.htm>>
- [17] PG Admin 3 v. 1.22.2: [online]. [vid. 2018-01-05].
URL <<https://www.postgresql.org/ftp/pgadmin/pgadmin3/v1.22.2/win32/>>
- [18] JAVA funkce Startswith(): [online]. [vid. 2018-01-05].
URL <https://www.tutorialspoint.com/java/java_string_startswith.htm>

- [19] SQL dotaz LIKE: [online]. [vid. 2018-01-05].
URL <https://www.w3schools.com/sql/sql_like.asp>
- [20] Online Polynomial Regression: [online]. [vid. 2018-01-05].
URL <<http://www.xuru.org/rt/PR.asp#CopyPaste>>