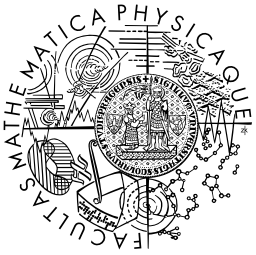

MATEMATICKO-FYZIKÁLNÍ FAKULTA
Univerzita Karlova

BAKALÁŘSKÁ PRÁCE

Martin Gora

Vylepšení agregace dotazovacího enginu pro grafové databáze

Katedra softwarového inženýrství

Vedoucí bakalářské práce: Mgr. Tomáš Faltín
Studijní program: Informatika
Studijní obor: Softwarové a datové inženýrství

Praha 2021

Prohlašuji, že jsem tuto bakalářskou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů. Tato práce nebyla využita k získání jiného nebo stejného titulu.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne ....

Podpis autora

Chtěl bych poděkovat mému vedoucímu Mgr. Tomáši Faltínovi za jeho pomoc, ochotu a nadšení při zpracovávání daného tématu. Déle bych chtěl poděkovat rodině, která mi poskytla zázemí pro práci a plnou podporu.

Název práce: Vylepšení agregace dotazovacího enginu pro grafové databáze

Autor: Martin Gora

Katedra: Katedra softwarového inženýrství

Vedoucí bakalářské práce: Mgr. Tomáš Faltín, Katedra softwarového inženýrství

Abstrakt: Abstrakt.

Klíčová slova: grafové databáze agregace dat proudové systémy

Title: Improvement of data aggregation in query engine for graph databases

Author: Martin Gora

Department: Department of Software Engineering

Supervisor: Mgr. Tomáš Faltín, Department of Software Engineering

Abstract: Abstract.

Keywords: graph databases data aggregation streaming systems

Obsah

Úvod

Tady ma byt text.

1. prvni

aaa Anděl (2007, Věta 4.22) aa

2. druha

3. treti

4. Experiment

Aby bylo možné porovnat stávající řešení s nově navrženým řešením na poli rychlosti zpracovávání dotazů, podrobili jsme zmíněná řešení experimentu. Vykonaný experiment proběhne na reálných grafech různé velikosti s uměle vygenerovanými vlastnostmi naležící vrcholům. Nad danými grafy provedeme vybrané množství dotazů, které nám umožní sledovat a porovnat chování řešení v různých situacích. Experiment bude zakončen diskuzí nad výsledky.

4.1 Příprava dat

Pro náš experiment jsme použili tři orientované grafy z databáze SNAP[1].

	#Vrcholů	**#Hran**
Amazon0601	403394	3387388
WebBerkStan	685230	7600595
As-Skitter	1696415	11095298

Tabulka 4.1: Vybrané grafy pro experiment

- **Amazon0601:** Jedná se o graf vytvořený procházením webových stránek Amazonu na základě featury „Customers Who Bought This Item Also Bought“ ze dne 1.6.2003. V grafu existuje hrana z i do j, pokud je produkt i často zakoupen s produktem j.
- **WebBerkStan:** Graf popisuje odkazy webových stránek domén `https://www.stanford.edu/` a `https://www.berkeley.edu/`. Vrcholem je webová stránka a hrana představuje hypertextový odkaz mezi stránkami.
- **As-Skitter:** Topologický graf internetu z roku 2005 vytvořený programem `traceroutes`. Ačkoliv je uvedeno, že daný graf je neorientovaný, vnitřní hlavička souboru uvádí opak, proto jsme se daný graf rozhodli přesto využít.

Samotné grafy obsahují pouze seznam hran. Abychom mohli dané grafy využít, bylo nutné je transformovat a vygenerovat k nim Properties na vrcholech. Při příkladu transformace budeme vycházet z následující ukázky hlavičky (graf Amazon0601):

```
# Directed graph (each unordered pair of nodes is saved once):
    Amazon0601.txt:
# Amazon product co-purchaisng network from June 01 2003
# Nodes: 403394 Edges: 3387388
# FromNodeId        ToNodeId
0        1
0        2
0        3
0        4
```

[1] Leskovec a Krevl (2014)

4.1.1 Transformace grafových dat

Výstupem transformace budou soubory popisující schéma vrcholů/hran NodeTypes.txt/EdgeTypes.txt a datové soubory vrcholů/hran Nodes.txt/Edges.txt. V našem případě graf bude obsahovat pouze jeden typ hrany a jeden typ vrcholu. Dané omezení ovlivňuje pouze vyhledávání vzoru, které není určující pro náš experiment.

Ukázka zvoleného schématu pro Nodes.txt/Edges.txt:

```
Soubor EdgeTypes.txt:
[
{ "Kind": "BasicEdge" }
]

Soubor NodeTypes.txt:
[
{ "Kind": "BasicNode" }
]
```

Generování souborů Edges.txt/Nodes.txt provádí program, který je obsahem přílohy zdrojových kódů A.1 v souboru GrapDataBuilder.cs. Výstupní soubor Edges.txt bude obsahovat hrany v rostoucím pořadí dle položky `FromNodeId` z originálního souboru s přidělenými `IDs` od hodnoty `ID` posledního vrcholu v souboru Nodes.txt. Samotný soubor Nodes.txt obsahuje setřizené vrcholy podle `ID` v rostoucím pořadí. Je nutné zmínit, že setřízení dat podle `ID` není nežádoucí, jelikož nezaručuje nic o seskupení vrcholů v daném grafu. Pro připomenutí zmíníme, že prvni sloupeček v datových souborech Edges.txt a Nodes.txt odpovídá unikátnímu `ID` v rámci celého grafu.

Následuje ukázka výstupních souborů transformace pro graf Amazon0601:

```
Soubor Edges.txt:
403395 BasicEdge 0 1
403396 BasicEdge 0 2
...

Soubor Nodes.txt:
0 BasicNode
1 BasicNode
...
```

4.1.2 Generování Properties vrcholů

Posledním krokem přípravy dat pro experiment je vygenerování Properties vrcholů. Jsme si vědomi, že nejideálnější způsob testování je graf s reálnými daty, nicméně dané omezení jsme se rozhodli aplikovat kvůli problematickému hledání vhodných dat, které nevyžadují netriviální transformaci do vhodného vstupního formátu. Proto pro každý vrchol náhodně vygenerujeme hodnoty tří Properties.

Property	Type	Popis
PropOne	integer	`Int32` s rozsahem $[0, 100000]$
PropTwo	integer	`Int32` s rozsahem [`Int32.MinValue`, `Int32.MaxValue`]
PropThree	string	délka $[2, 8]$ ASCII znaků s rozsahem $[33, 126]$

Tabulka 4.2: Generované Properties vrcholů

- **PropTwo** hodnoty jsou rovněž generováný střídavě kladně a záporně, aby nastal rovnoměrný počet záporných a kladných hodnot.
- **PropThree** hodnoty jsou pouze ASCII znaky z rozsahu $[33, 126]$. Dané omezení výplývá z vlastností dotazovacího enginu, aby bylo možné bez obtíží načíst datový soubor.

Na základě tabulky generovaných Properties 4.2 následuje ukázka upraveného souboru schématu pro vrcholy:

```
Soubor NodeTypes.txt:
[
{
"Kind": "BasicNode",
"PropOne": "integer",
"PropTwo": "integer",
"PropThree": "string"
}
]
```

Výsledné hodnoty Properties do souborů Edges.txt/Nodes.txt jsou vygenerovány pomocí programu, který používá generátor náhodných čísel. Program je obsažen v příloze zdrojových kódů A.1 v souboru PropertyGenerator.cs. Pro každý graf bylo použité jiné `Seed` pro inicializaci náhodného generátoru. Samotná `Seeds` byla vygenerována rovněž náhodně.

	Seed
Amazon0601	429185
WebBerkStan	20022
As-Skitter	82

Tabulka 4.3: Inicializační hodnoty náhodného generátoru pro PropertyGenerator.cs

Program generuje hodnoty definované ve statické položce `propGenerators` a zachovává jejich pořadí ve výsledném datovém souboru. Aby nedocházelo k omylům při opakování experimentů, uvádíme útržek kódu použité inicializace položky dle tabulky generovaných vlastností 4.2 pro všechny tři grafy:

```
static PropGenerator[] propGenerators = new PropGenerator[]
{
    new Int32Generator(0, 100_000, false),
    new Int32Generator(true),
    new StringASCIIGenerator(2, 8, 33, 126)
};
```

Timto jsme dokončili poslední nutný krok k vygenerování platných vstupních dat pro dotazovací engine. Použité grafy k transformaci a výsledné datové soubory jsou obsahem přílohy grafů pro experiment A.3

4.2 Výběr dotazů

Dotazy použité při experimentu dělíme do tří kategorií a to Match, Order by a Group by. Pro připomenutí zmíníme, že proměnné použité v jiných částech než Match způsobují ukládání daných proměnných do tabulky. Přidělené zkratky dotazům budou uváděný ve výsledcích experimentu namísto celých dotazů.

4.2.1 Dotazy Match

Každý dotaz provádí vyhledáváním vzoru v grafu. Níže zmíněné dotazy nám při experimentu pomohou oddělit čas agregací od času stráveném vyhledáváním vzoru.

Zkratka	Dotaz
M_Q1	select count(*) match (x) -> (y) -> (z);
M_Q2	select x match (x) -> (y) -> (z);
M_Q3	select x, y match (x) -> (y) -> (z);
M_Q4	select x, y, z match (x) -> (y) -> (z);

Tabulka 4.4: Dotazy Match

- **M_Q1** testuje pouze dobu strávenou vyhledáváním vzoru.
- **M_Q2** testuje vyhledávání společně s ukládáním proměnné x do tabulky výsledků.
- **M_Q3** testuje vyhledávání společně s ukládáním proměnné x a y do tabulky výsledků.
- **M_Q4** testuje vyhledávání společně s ukládáním proměnné x, y a z do tabulky výsledků.

4.2.2 Dotazy Order by

Zkratka	Dotaz
O_Q1	select y match (x) -> (y) -> (z) order by y;
O_Q2	select y, x match (x) -> (y) -> (z) order by y, x;
O_Q3	select x.PropTwo match (x) -> (y) -> (z) order by x.PropTwo;
O_Q4	select x.PropThree match (x) -> (y) -> (z) order by x.PropThree

Tabulka 4.5: Dotazy Order by

- **O_Q1** testuje třídění podle `ID` vrcholů y.
- **O_Q2** přidává do kontextu **O_Q1** overhead za porovnávání a ukládání další proměnné.
- **O_Q3** testuje třídění náhodně vygenerovaných hodnot Int32 (viz 4.2).
- **O_Q4** testuje třídění náhodně vygenerovaných řetězců (viz 4.2).

4.2.3 Dotazy Group by

Zkratka	Dotaz
G_Q1	select min(y.PropOne), avg(y.PropOne) match (x) -> (y) -> (z);
G_Q2	select min(y.PropOne), avg(y.PropOne) match (x) -> (y) -> (z) group by y;
G_Q3	select min(y.PropOne), avg(y.PropOne) match (x) -> (y) -> (z) group by y, x;
G_Q4	select min(x.PropOne), avg(x.PropOne) match (x) -> (y) -> (z) group by x.PropTwo;
G_Q5	select min(x.PropOne), avg(x.PropOne) match (x) -> (y) -> (z) group by x;
G_Q6	select min(x.PropOne), avg(x.PropOne) match (x) -> (y) -> (z) group by x, y;
G_Q7	select min(x.PropOne), avg(x.PropOne) match (x) -> (y) -> (z) group by x.PropOne;

Tabulka 4.6: Dotazy Group by

Pro volbu agregačních funkcí jsme zvolili `min(y.PropOne)` a `avg(y.PropOne)`, protože představují netriviální práci narozdíl od funkcí `sum`/`count`. V případě `min(y.PropOne)` v paralelních řešeních dochází k mechanismu `CompareExchange` a u `avg(y.PropOne)` dojde ke dvou atomickým přičtením.

- **G_Q1** testuje single group Group by. Vše je agregováno pouze do jedné skupiny.
- **G_Q2** testuje vytváření skupin podle `ID` vrcholů y.
- **G_Q3** přidává k **G_Q2** overhead za ukládání a zpracovávání (hash + compare) další proměnné.
- **G_Q4** testuje vytváření skupin náhodně vygenerovaných hodnot Int32 (viz 4.2).
- **G_Q5** testuje situaci, kdy při paralelním zpracování žádné jiné vlákno během vyhledávaní nenajde stejnou hodnotu.
- **G_Q7** testuje situaci, kdy dojde k rozprostření několika stejných hodnot v grafu. Dodává větší šanci, že nějaké vlákno během paralelního zpracování dostane stejnou skupinu jako vlákno jiné.

4.3 Metodika

Pro provedení experimentu jsme připravili jednoduchý benchmark, který je součástí příloh zdrojových kódů A.1. Paralelizování řešení jsme otestovali při zatížení všech dostupných jader procesoru (argument `ThreadCount = 8`). Při spuštění programu dojde k navýšení priority procesu, aby docházelo k méně častému vykonávání ostatních procesů na pozadí během testování. Pro `ThreadCount = 1` navíc dochází k navýšení priority hlavního vlákna. To není možné u paralelního testování, protože vlákna běží v nativním `ThreadPool`, který neumožňuje navyšování priority vláken.

Následuje ukázka hlavní smyčky benchmarku:

```
...
 WarmUp(...);
...
double[] times = new double[repetitions];
for (int i = 0; i < repetitions; i++)
{
    CleanGC();

    var q = Query.Create(..., false);

    timer.Restart();

    q.Compute();

    timer.Stop();
    times[i] = timer.ElapsedMilliseconds;

    ...
}
```

Hlavní smyčka benchmarku se skládá z 5-ti opakování warm up fáze následovanou 15-ti opakováními měřené části.

V konstruktoru `Query.Create(..., false)` argument `false` způsobuje, že vykonávaný dotaz neprovede `select` část dotazu, která není cílem testování.

Před měřením dochází vždy k úklidu haldy.

```
static void CleanGC()
{
    GC.Collect();
    GC.WaitForPendingFinalizers();
    GC.Collect();
}
```

K měření uplynulé doby jsme použili nativní třídu C# `Stopwatch`, protože náš hardware a operační systém podporuje high-resolution performance counter. Pro interpretaci výsledků jsme zvolili medián naměřených hodnot, který je doprovázen minimem a maximem.

4.3.1 Argumenty pro jednotlivé grafy

Pro měření argumenty `FixedArraySize` a `VerticesPerThread` jsme volili následovně:

	FixedArraySize	**VerticesPerThread**
Amazon0601	4194304	512
WebBerkStan	4194304	512
As-Skitter	8388608	1024

Tabulka 4.7: Výber argumentů konstruktoru dotazu pro grafy

Dané argumenty se nám nejvíce osvědčili v průběhu vývoje dotazovacího enginu. Vyhledávání vzoru na nich docilovalo nejrychlejších výsledků.

4.3.2 Hardwarová specifikace

Všechny testy proběhly na notebooku Lenovo ThinkPad E14 Gen. 2 verze 20T6000MCK s operačním systémem Windows 10 x64.

- 8 jádrový procesor AMD Ryzen 7 4700U (2GHz, TB 4.1GHz)
- 24GB RAM DDR4 s 3200 MHz

Každému testování předcházel studený reboot systému a odpojení od internetu. V průběhu testování neběžel žádný klientský proces kromě benchmarku a nativních systémových procesů. Rovněž, použitý notebook byl napájen po celou dobu testování.

4.4 Výsledky a diskuze

Závěr

Tady ma byt text

Seznam použité literatury

Anděl, J. (2007). *Základy matematické statistiky.* Druhé opravené vydání. Matfyzpress, Praha. ISBN 80-7378-001-1.

Leskovec, J. a Krevl, A. (2014). SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`.

Seznam obrázků

Seznam tabulek

Seznam použitých zkratek

A. Přílohy

A.1 Zdrojové kódy

Přílohou této bakalářské práce jsou zdrojové kódy dotazovacího enginu, benchmarku a použité knihovny HPCsharp. Vše zmíněné je přiloženo v rámci jednoho projektu Visual Studia, kromě souborů Gitu. Dále, mimo projekt jsou přiloženy zdrojové kódy programů na generování vstupních grafů pro experiment. Jedná se o soubory GraphDataBuilder.cs a PropertyGenerator.cs.

A.2 Online Git repozitář

V době vydání tohoto textu probíhal vývoj dotazovacího enginu na GitHubu.

`https://github.com/goramartin/QueryEngine`

A.3 Použité grafy při experimentu

Grafy použité při experimentu jsou vloženy do odpovídajících složek dle názvu grafu. Složky obsahují originálni grafy před transformací a datové soubory po transformaci.

A.4 druha priloha

Priloha po prvni strance priloh