KIV/OPSWI
Porovnání rychlosti dotazů v MongoDB a Elasticsearch

Veronika Kutková

29. května, 2017

Obsah

1 Úvod

Cílem tohoto projektu bylo porovnat rychlost prohledávání dat uložených v NoSQL databázi (MongoDB) a Elasticsearch. Pro testování byla zvolena data reprezentující množinu balíků Java tříd a jejich dostupné veřejné metody.

2 MongoDB

MongoDB patří mezi NoSQL databáze, data jsou v MongoDB ukládána ve formátu JSON. Pro práci s MongoDB je zapotřebí si stáhnout MongoDB server z adresy `https://www.mongodb.com/download-center#community`.

Ke spuštění serveru použijeme příkaz `mongod -dbpath CUSTOM_DB_PATH --httpinterface --rest`, k importu dat na server příkaz `mongoimport --jsonArray -d <DATABASE> -c <COLLECTION> --file data.json`

Jedním z existujících GUI k MongoDB je nástroj Robomongo.

3 Elasticsearch

Elasticsearch je škálovatelné úložiště, které lze fulltextově prohledávat. Využívá open-source knihovnu Apache Lucene, představující vyhledávací engine. Elasticsearch je dostupný zdarma pod licencí Apache. Ukládaná data jsou reprezentována ve formátu JSON a komunikace se samostatně běžícím serverem probíhá prostřednictvím REST API.

3.1 Princip

Elasticsearch funguje na principu invertovaných indexů, o jejichž implementaci se stará Apache Lucene. Při vkládání záznamů (tzv. *dokumentů*) do Elasticsearch projdou veškerá data analýzou, při které jsou řetězce textu rozděleny na jednotlivá slova, všechna písmena jsou převedena na malá a z textu je odstraněna interpunkce. Teprve tyto výsledné tokeny (*terms, tokens*) jsou indexovány.

Při použití invertovaných indexů je každý token namapován na kolekci dokumentů, v nichž se vyskytuje (na rozdíl od tradičního mapování ve stylu „dokument → tokeny“). Jelikož jsou tokeny abecedně řazeny a dotazovaný výraz je při dotazování analyzován stejným způsobem jako při vytváření indexu, prohledávání dokumentů je díky invertovaným indexům opravdu rychlé – stačí se podívat, ve kterých dokumentech je (resp. není) vyhledávaný token obsažen.

Pokud vyhledáváme podle více tokenů zároveň, Elasticsearch postupuje tím způsobem, že vyhledá výskyty všech zadaných tokenů a následně

provede jejich průnik (v případě podmínky AND), resp. jejich sjednocení (podmínka OR).

Rychlost, s jakou dokážeme tokeny prohledávat, se odvíjí od způsobu, jakým jsou tokeny indexovány. Vzhledem k abecednímu řazení tokenů je snadné vyhledávat podle jejich počátečních písmen, pokud však např. víme, že budeme chtít vyhledávat spíše podle konců tokenů, vytvořeníme index pro revertovaný výraz (např. *index* → *xedni*) a vyhledáváme opět podle počátků tokenů.

Podobně lze postupovat i při vyhledávání podřetězců, kdy je před indexováním navíc zapotřebí jednotlivé tokeny rozdělit na tzv. *n-gramy* a ty teprve indexovat.

3.2 Instalace a spuštění

Elasticsearch je ke stažení na adrese `https://www.elastic.co/downloads/elasticsearch`.

Pro nastartování Elasticsearch stačí spustit skript `elasticsearch` (v případě OS Windows `elasticsearch.bat`), umístěný v adresáři `bin` staženého archivu. Server Elasticsearch poté běží lokálně na portu 9200. Pro vyzkoušení dotazů lze využít např. Chrome rozšíření Postman.

4 Popis experimentu

Experiment porovnání rychlosti dotazů v MongoDB a Elasticsearch proběhl nad daty představujícími balíky tříd knihoven získaných z lokálního maven repository.

4.1 Vstupní data

Každý balík, jednoznačně definovaný svým názvem a přesným označením verze, sdružuje vždy několik tříd, obsahujících různé množství veřejných metod.

Vstupní data byla získána z lokálního maven repository (pomocí bash skriptu) následujícím způsobem:

```
Input: repo: local maven repository
Output: paths.txt: .jar paths
Output: methods.txt: public methods of each .jar
for each jar in repo do
    paths.txt ← jar.path
end
for each path in paths.txt do
    version ← path.version
    go to path
    create/empty methods.txt file
    for each .jar in path do
        jar.txt ← classes names
        unzip into temp folder
        for each line in jar.txt do
            methods.txt ← "PATH: "+ line
            methods.txt ← "VERSION: "+ version
            methods.txt ← all public methods
        end
    end
    delete temp folder
    go back to repository folder
end
```

Algorithm 1: Získání vstupních dat.

Pro získání názvů tříd obsažených v *.jar* archivu slouží příkaz `jar tf <filename> | grep '.class$'`. Získání názvů veřejných metod je možné použitím příkazu `javap -public <path>`.

Dále je potřeba projít všechny vytvořené textové soubory *methods.txt* (v adresářích ze souboru *paths.txt*) a naparsovat je do formátu JSON.

1. Obsah souboru rozdělíme podle řetězce „PATH: “, tím dostaneme jednotlivé třídy archivu.

2. Z textu za řetězcem „PATH: “ oddělíme označení třídy (část za posledním lomítkem). Ve zbylém textu nahradíme lomítka tečkami a uložíme jako název balíku (atribut *package*).

3. Další řádka, začínající řetězcem „VERSION: “, představuje označení verze balíku. Tento údaj uložíme do atributu *version*.

4. Následující řádky již představují jednotlivé veřejné metody, jejichž názvy uložíme vždy do atributu *header* v kolekci *methods*.

Tímto způsobem bylo zpracováno 438 *.jar* archivů a celkem získáno 4204 balíků. Ukázka struktury výsledných vstupních dat je vidět na následujícím příkladu:

```
{
   "package":"org.apache.commons.lang",
   "version":"2.5",
   "classes":[
      {
         "name":"ArrayUtils.class",
         "methods":[
            {
               "header":"public static java.util.Map toMap(java.lang.Object[]);"
            }
         ]
      },
      {
         "name":"BitField.class",
         "methods":[
            {
                "header" :"public int getValue(int);"
            },
            {
                "header" :"public short getShortValue(short);"
            }
         ]
      }
   ]
}
```

Příklad 1: Ukázka struktury vstupních dat

4.2 Dotazy

Nad těmito vstupními daty bylo v rámci experimentu opakovaně provedeno několik dotazů (viz dále), a to jak v MongoDB, tak i v Elasticsearch a byly změřeny časy potřebné pro jejich provedení.

query1 hledá podle názvu metody (výskyt jen jednou),

query2 *query1* doplněné o název třídy,

query3 *query2* doplněné o název balíku,

query4 hledá podle názvu metody (výskyt celkem v 1545 dokumentech),

query5 *query4* doplněné o název třídy,

query6 *query5* doplněné o název balíku,

query7 hledá balík podle 4 názvů tříd a celkem 14 názvů metod v nich obsažených (právě jeden výskyt),

query8 obdoba *query7*, hledá balík podle 8 názvů tříd a celkem 64 názvů metod v nich obsažených,

query9 obdoba *query8*, obsahuje však název neexistující třídy, tj. nevrací výsledek.

Aby bylo možné v MongoDB vyhledávat i jen podle částečné shody názvu metody (např. bez seznamu parametrů), je nutné vynutit text index pomocí `db.<collection>.ensureIndex({"classes.methods.header":"text"})`. V dotazu voláme `db.<collection>.find({$text:{$search:"method"}})`.

Při vkládání do Elasticsearch prochází veškerý text analýzou, při které je převeden na tokeny (tokenizován), indexace probíhá až pro výsledná slova (tokeny). Při fulltextovém dotazování je nad výrazem dotazu provedena stejná analýza, a proto lze pro vyhodnocení dotazu použít indexované hodnoty.

4.3 Výsledky

Každý z dotazů byl proveden celkem stokrát. V následující tabulce (tabulka 4.3) jsou zachyceny průměrné časy běhu jednotlivých dotazů a jejich odchylky (v ms).

	MongoDB		**Elasticsearch**	
	průměr	**odchylka**	**průměr**	**odchylka**
query1	0,000000	0,000000	1,700001	1,072391
query2	0,000000	0,000000	1,630001	0,482826
query3	0,000000	0,000000	1,670001	0,510979
query4	11,690000	2,444156	2,479999	0,741345
query5	33,100000	4,341659	1,320000	0,466469
query6	34,500000	4,038564	1,400001	0,489934
query7	721,700000	78,106914	2,189999	0,716878
query8	1178,120000	61,026106	11,610000	2,190402
query9	1185,820000	57,490935	3,140001	1,191812

Tabulka 1: Časy potřebné k provedení jednotlivých dotazů [ms].

Zatímco u jednoduchých dotazů, které vracejí právě jeden dokument (*query1–query3*), vyhodnotí MongoDB dotaz okamžitě a vykazuje tak vyšší rychlost než Elasticsearch, s rostoucí složitostí dotazů se situace obrací.

V případě dotazů *query4–query6* se průměrný čas zpracování v MongoDB pohybuje v řádech desítek milisekund, zatímco Elasticsearch zůstává v řádech jednotek milisekund.

Kombinací více názvů tříd a metod složitosti dotazů výrazně rostou a rozdíly v rychlostech vyhodnocení dotazů se prohlubují. I v případě nejsložitějšího z testovaných dotazů (*query8*) zvládá Elasticsearch vyhodnotit výsledek za necelých 12 ms, provedení stejného dotazu v MongoDB však trvá 1178 ms, tedy přibližně stokrát déle. I pokud takto složitý dotaz nic nevrací (případ *query9*), vyhledávání v MongoDB trvá přibližně stejně (téměř 1186 ms). Elasticsearch si se stejným dotazem poradí za pouhé 3 ms.

Vysoká rychlost vyhledávání pomocí Elasticsearch je zapříčiněna použi-

tím tzv. invertovaných indexů, kdy ke každému slovu uchováváme kolekci dokumentů, ve kterých je slovo obsaženo (místo toho, abychom ukládali kolekci obsažených slov pro každý dokument).

4.4 Struktura projektu a postup experimentu

Výsledný projekt má následující strukturu:

- Adresář ***scripts*** obsahuje skripty v bashi pro zpracování dat v lokálním maven repository.
- Adresář ***data*** obsahuje Python skripty pro naparsování dat do formátu JSON.
- Adresář ***queries*** obsahuje testované dotazy ve verzi pro MongoDB i Elasticsearch.
- Skript ***mongodb.py***, resp. ***elastic.py*** slouží k opakovanému spuštění definovaných dotazů a měření doby trvání.

Pro vyzkoušení experimentu ve vlastním prostředí je potřeba provést tyto kroky:

1. Pro vytvoření souboru *paths.txt* spusťte ve vašem lokálním maven repository skript *paths.sh* příkazem `./paths.sh`.
2. Pro vytvoření *methods.txt* souborů spusťte skript *methods.sh* (ve stejném adresáři).
3. Načtení a naparsování dat do souboru *data.json* proveďte spuštěním *parser.py* příkazem `python parser.py`[1]. Na začátku tohoto souboru definujte cestu ke svému lokálnímu maven repository.
4. Importujte data do MongoDB a Elasticsearch (databázi nazvěte *opswi*, pro název kolekce použijte *data*).
5. Vynuťte v MongoDB text index příkazem `db.data.ensureIndex({ "classes.methods.header":"text"})` (viz kapitola 4.2).
6. Poté je již možné spustit provádění dotazů v MongoDB a v ElastiSearch (`python mongodb.py`, resp. `python elastic.py`).
   Předpoklad: MongoDB běží na portu 27017, Elasticsearch na portu 9200, název databáze je *opswi* a název kolekce *data*.

[1] Je nutné mít nainstalovaný Python. Při implementaci byla použita verze 2.7.13.

5 Závěr

Vyzkoušený přístup lze aplikovat při zkoumání závislostí Java knihoven. Implementaci experimentu jsem realizovala v programovacím jazyce Python za využití knihoven *pymongo* a *elasticsearch*. Provedený experiment prokázal, že s rostoucí složitostí dotazů je rozhodně výhodnější použít Elasticsearch než MongoDB.