České vysoké učení technické v Praze
Fakulta Informačních technologií
Obor Softwarové inženýrství

Semestrální práce

Využití a zpracování dat o platbách na internetových obchodech

Josef Doležal
3. ročník

1 Úvod

Pro svou semestrální práci k předmětu BI-BIG jsem si vybral problematiku nakupování na internetových obechodech. Ve větších obchodech tohoto typu vznikne během dne tisíce, mnohndy až desetitisíce objednávek [1]. Z tohoto důvodu přestávají být relační databáze dostatečně výkonné a je potřeba se uchýlit k jiné technologii.

Jedním z možných řešení problému může být distribuovaná databáze, která je pro tento účel optimalizována. V svém řešení uvažuji jako distribuovanou databázi Apache Cassandra.

Data o objednávkách jsou pro obchody důležitým ukazatelem. Mohou o vypovídat o zálibách, volnočasových aktivitách ale i zdravotním stavu zákazníka [2]. Internetový obchod může na základě takových informací přizpůsobit danému zákazníkovi svou nabídku. Aby byla konečná informace o zákazníkovi relevantní, je zapotřebí mít dostatečné množství dat, která se budou analyzovat.

Analýza nad takovými daty může být pro velké množství zákazníků časově náročná. Z tohoto důvodu se nad databázemi staví vyhledávací index, který umožní po počátečním zaindexování databáze vyhledávat ve výrazně nižším čase.

2 Mé řešení

2.1 Databáze

Při vytváření semestrální práce jsem využil technologie Docker pro zprovoznění databáze na lokálním počítači. Cassandru o jednom nodu jsem zprovoznil následujícím příkazem[3]:

```
docker run --detach --name cassone poklet/cassandra
```

V tuto chvíli běží v Docker kontejneru Cassandra, napojit na ní je možné pomocí cqlsh příkazu. Pro použití tohoto příkazu jsem vytvořil nový kontejner:

```
docker run -it -v "$PWD:/var/data_src" --rm --net
    ↪ container:cassone poklet/cassandra cqlsh
```

Nyní je možné využívat plnohodnotné rozhraní příkazové řádky cqlsh pro komunikaci s Cassandrou. Do kontejneru je namapovaný aktuální adresář z lokálního počítače, v kontejneru je umístěn ve složce /var/data_src.

2.2 Zdroj dat

Ve své práci jsem využil dostupná data z webu BigDataBench[4]. Tyto data reprezentují objednávky v internetovém obchodu. Původní data nebyla vhodná pro vožení do Cassandry kvůli špatnému formátu. Data jsem očistil pomocí textového editoru o hlavičku - v příloze je možné zdrojové soubory v adresáři src.

Ze stažených datasetů jsem náhodně vybral 5000 záznamů, které jsem následně připravil pro vložení do cassandry:

```
gshuf src/OS_ORDER.txt | head -n5100 | awk -F" " '{printf
    ↪ ("%s,%s,%s,%s,%s,%s\n", $1, $2, $3, $4, $5, $6)}' >
    ↪  src/OS_ORDER.csv
```

```
gshuf src/OS_ORDER_ITEMS.txt | head -n5100 | awk -F" " '{
    ↪ printf("%s,%s,%s,%s,%s,%s,%s\n", $1, $2, $3, $4, $5
    ↪ , $6, $7)}' > src/OS_ORDER_ITEMS.csv
```

Uvedené příkazy jsou přiložené v souboru prepare_data.sh. Výstupem jsou CSV soubory připravené na vložení do databáze.

Zvolený zdroj obsahoval pouze dva datasety, třetí jsem proto doplnil pomocí generátoru v jazyce swift.

2.3 Vložení dat

Pro vložení dat do Cassandry použijeme cqlsh z Docker kontejneru.

Jako první krok je nutné vytvoři keyspace a vybrat ho k používání:

```
// Drop old keyspace if exists
DROP KEYSPACE IF EXISTS dolezjo3;

// For class purpose, I have decided to use replication
    ↪ factor with value 1,
// in production strategy, I would use bigger value
CREATE KEYSPACE dolezjo3 WITH replication = {'class':'
    ↪ SimpleStrategy','replication_factor':1};

// Set created keyspace as current keyspace
USE dolezjo3;
```

Nyní máme vytvořený keyspace a je potřebné vytvořit tabulky, do kterých následně vložíme data.

```
// Create keyspace structure

CREATE TABLE orders (
  id int PRIMARY KEY,
  code text,
  buyer_id text,
  created_at text,
  payed_at text,
  buyer_ip text
 );
```

```
CREATE TABLE order_items (
  id int PRIMARY KEY,
  order_id int,
  goods_id int,
  goods_number double,
  shop_price double,
  goods_price double,
  goods_amount double
);

CREATE TABLE order_returns (
  id int PRIMARY KEY,
  return_date text,
  order_id int
);
```

V tuto chvíli je připraven keyspace a je možné vložit data:

```
// Insert data

COPY orders (id, code, buyer_id, created_at, payed_at,
    ↪ buyer_ip) FROM '/var/data_src/src/OS_ORDER.csv';
COPY order_items (id, order_id, goods_id, goods_number,
    ↪ shop_price, goods_price, goods_amount) FROM '/var/
    ↪ data_src/src/OS_ORDER_ITEMS.csv';
COPY order_returns (id, return_date, order_id) FROM '/var
    ↪ /data_src/src/OS_ORDER_RETURNS.csv';
```

Keyspace je připraven pro další použití.

3 Práce s daty

3.1 Dotazy

3.2 Agregace

Nad vložnými daty lze nyní spustit agregace. Pomocí technologie Apache Hive spustíme operaci, která provede potřebnou sekvenci map-reduce kroků a a výsledek uloží do nově vytvořené tabulky.

Prvním selectem je vyhledání uživatelů podle geolokace. Uživatele z dané lokace (zde simulováno IP adresou) lze pak oslovit v velmi dobře cílenými obchodními sděleními. Takový use case může být například oslovení lidí z povodňových oblastí se slevou na pojištění.

```
# Select all users from affected area (By IP)
CREATE TABLE affected_users AS SELECT buyer_id, buyer_ip
    ↪ FROM orders where buyer_ip LIKE '218.*';
```

```
CREATE TABLE latest_bought_products AS SELECT orders.
    ↪ buyer_id, order_items.goods_id FROM orders JOIN
    ↪ order_items ON (orders.id = order_items.order_id)
    ↪ WHERE orders.id > 50000;
```

Druhý agregující dotaz vybírá produkty zakoupené v poslední době (omezením id) a uživatele, kteří produkt zakoupili. Lze tak sestavit mapu produktů, které jsou oblíbené v určité období nebo mapu aktivních uživatelů.

```
# Select all users from affected area (By IP)
CREATE TABLE affected_users AS SELECT buyer_id, buyer_ip
    ↪ FROM orders where buyer_ip LIKE '218.*';

CREATE TABLE latest_bought_products AS SELECT orders.
    ↪ buyer_id, order_items.goods_id FROM orders JOIN
    ↪ order_items ON (orders.id = order_items.order_id)
    ↪ WHERE orders.id > 50000;
```

4 Využití dat

V úvodu práce jsem zmínil obecné využití dat. Využití dat z mé práce je možné v širokém měřítku. Obchod může analyzovaná data použít pro konkrétní cílení nabídek na zákazníka.

Oproti doporučovacím systémům mají big data výhodu v tom, že další nabízené zboží se nezakládá jen na aktuální objednávce (co si např. zakoupili ostatní zákazníci s vybraným zbožím), ale i na historii nákupů. Zákazník tedy vidí více relevantních nabídek a je náchylnější ke koupi.

Další možností je cílení pomocí geolokace. Z aktuální IP zákazníka je možné přibližně zjistit kde se nachází a nabízet zboží, které je v daném místě oblíbené nebo by mohlo být v budoucnu žádané (např. nabízení připojištění lidem ze záplavových oblastí).

5 Ukázka dat

V práci využívám následující tabulku, která reprezentuje jednu objednávku v systému. U objednávky se eviduje její ID, číslo, ID zákazníka, datum vytvoření a zaplacení, ip zákazníka. Předpokládá se, že výstup agregací bude zpracovávat externí systém, tedy ID zákazníka je dostačující údaj.

id int created_at text	code text payed_at text	buyer_id text buyer_ip
19667 2016-12-03	101208019976 2016-12-05	22681 123.127.124.2

Druhou důležitou tabulkou jsou produkty v objednávce. Podle produktů je možné analyzovat a predikovat chování zákazníka. Neméně důležitým údajem jsou cena a množství. Tabulka obsahuje tyto sloupce: ID položky, ID objednávky, ID zboží, počet položek, obchodní cena, skutečná cena, cena objednávky.

id int	order_id int	goods_id int	goods_number double	shop_price double	goods_price double	goods
425	292	1010060	999.00	10.400000	10.40	10389

V běžném provozu internetových obchodů se stává, že zákazníci zboží vrací. Tento fakt zachycuje třetí tabulka, skládající se z id, data vrácení a identifikátoru objednávky:

id int	return_date date	order_id int
425	2016-12-06	1010

6 Popis technologií

Ve své semestrální práci předpokládám využití technologií Apache Cassandra a Apache Solr. Obě technologie jsou vedené jako open source a je tedy možné je využívat be nutnosti placení licence.

Apache Cassandra je distribuovaná databáze navržená tak, aby zvládala obsluhovat velké množství dat. Tyto data umí Cassandra distribuovat na mnoho zařízení najednou, čímž zajišťuje vysokou dostupnost, rychlost načítání dat a v neposlední řadě mnoho prostoru pro data. Jedná se NoSQL databázi, tedy databázi kde data nejsou uložená v běžné tabulkové podobě ale v podobě "klíč - hodnota".

Apache Solr je index nad databázím, v mém případě na Cassandrou. Solr umožňuje nad databází vyhledávat pomocí složitějších parametrů (např fulltext) a toto vyhledávání je díky indexaci velmi rychlé. Solr poskytuje REST API na tvorbu dotazů, díky tomu je možné vytvářet vlastní klientské aplikace.

V práci předpokládám, že by systém byl napojený na další zdroje, které by dodaly informace, chybějící v tuto chvíli v databázi. Takovým zdrojem může být například systém, který překládá IP adresu zákazníka na geolokaci.

Reference

[1] Rekordní rok pro Alzu. Loni zvýšila tržby o čtvrtinu na 14,5 miliardy korun. *Hospodářské noviny.* [online]. [cit. 2016-12-13]. Dostupné z: `http://byznys.ihned.cz/c1-65137080-alza-cz-zvysila-trzby-o-ctvrtinu-na-rekordnich-14-5-miliardy-vyridila-pet-mi`

[2] How Target Figured Out A Teen Girl Was Pregnant Before Her Father Did. *Forbes.* [online]. [cit. 2016-12-13]. Dostupné z `http://www.forbes.com/sites/kashmirhill/2012/02/16/`

how-target-figured-out-a-teen-girl-was-pregnant-before-her-father-did/#de48d2d34c62

[3] Cassandra on Docker. *GitHub*. [online]. [cit. 2016-12-13]. Dostupné z https://github.com/pokle/cassandra

[4] Downloads. *BigDataBench*. [online]. [cit. 2016-12-13]. Dostupné z http://prof.ict.ac.cn/BigDataBench/old/3.0/download.html