Sem vložte zadání Vaší práce.

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Bakalářská práce

Systém pro analýzu proudu dat v reálném čase

David Viktora

Vedoucí práce: Ing. Adam Šenk

26. dubna 2016

Poděkování

Poděkování ....

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 26. dubna 2016 .....

Odkaz na tuto práci

Viktora, David. *Systém pro analýzu proudu dat v reálném čase*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.

Abstrakt

V několika větách shrňte obsah a přínos této práce v češtině. Po přečtení abstraktu by se čtenář měl mít čtenář dost informací pro rozhodnutí, zda chce Vaši práci číst.

Klíčová slova Nahraďte seznamem klíčových slov v češtině oddělených čárkou.

Abstract

Sem doplňte ekvivalent abstraktu Vaší práce v angličtině.

Keywords Nahraďte seznamem klíčových slov v angličtině oddělených čárkou.

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Kratce o jednotlivých bodech zadání a struktuře prace, motivace

Kapitola 1

Situace v oblasti zpracovávání dat

V současné době generujeme obrovská množství dat - podle některých odhadů to například v roce 2012 mohlo být až 2,5 exabajtů za den[1]. Od té doby se rychlost přibývání dat stále zvyšuje. Například nárůst objemu dat dostupných na internetu je způsoben jeho neustále větším rozšířením a rostoucí dostupností. V roce 2016 je k němu připojeno již přes 3,3 miliardy obyvatel planety[2], což je téměř polovina všech. Roste také míra využívání internetu. Oproti dřívějšku na internetu trávíme nejen díky chytrým telefonům stále více času a využíváme například sociální sítě, internetové vyhledávání a další online služby. Během toho jsou nám zobrazována personalizovaná data a cílené reklamní nabídky. Také ve firemní i státní sféře výrazně roste stupeň využívání informačních technologií a v návaznosti na to objem produkovaných dat. S přibývajícími daty nastávají problémy s jejich zpracováním. Jedním z nich je obecně schopnost zpracovat tak veliké objemy dat, druhým je schopnost jejich zpracování v dostatečně krátkém, ideálně reálném čase. Často je přitom potřeba vyřešit oba tyto problémy naráz. Pro popis těchto dat a problémů spojených s jejich zpracováním se používá relativně nový pojem Big Data.

1.1 Big Data a technologie pro práci s nimi

Právě pojem Big Data je často označován za jeden z největších buzzwords[1] současného IT světa. I přes jeho popularitu nejsou přesně vymezené jeho hranice či definované pojmy zabývající se toutu oblastí. Obecně můžeme říci, že o Big Datech hovoříme v případech, kdy je potřeba zpracovávat objemy dat v řádech gigabajtů a více. To je velice zjednodušený popis tohoto termínu, přesná definice však neexistuje a na celý problém se dá dívat různými způsoby. Rozšířené je například také tvrzení říkající, že o Big Datech mluvíme zkrátka v

[1] Slova nebo fráze, které jsou v současné době populární

těch případech, kdy klasické databázové a softwarové nástroje především kvůli objemu těchto dat selhávají[4].

Přestože jednotná definice neexistuje, ustálilo se několik problémů, kterým je při práci s Big Daty potřeba čelit. Jedná se o takzvaná 3+1V - Volume, Velocity, Variety a později přidaná vlastnost Veracity[5]. Volume popisuje objem zpracovávaných dat, Velocity pak rychlost, jakou data přibývají. Charakteristikou Variety popisujeme různorodost dat a Veracity určuje úplnost a míru důvěryhodnosti dat. Ne vždy se setkáme se všemi těmito problémy naráz, každá z nich ale přidává na složitosti zpracování těchto dat.

Právě kvůli těmto vlastnostem se při práci s velkými objemy dat klasické technologie používané v minulosti stále častěji ukazují jako nedostatečně rychlé nebo obecně neschopné tato data zpracovat. Je proto nutné sáhnout po nových technologiích určených pro práci s nimi. Bez technologií pro Big Data se v dnešní době neobejdou třeba již zmiňované internetové vyhledávače nebo sociální sítě, využití ale nacházejí i v mnoha dalších oblastech. Jejich rozvoj je také předpokladem například pro další rozšíření tzv. Internet of Things[3].

1.1.1 Strukturovaná a nestrukturovaná data

Data obecně často rozdělujeme do dvou kategorií - na data strukturovaná a nestrukturovaná. Strukturovaná data jsou obvykle uložena v klasické relační databázi, nebo obecně utříděna po řádcích a s přesně definovanými sloupci. Ostatní data, která nemají takto pevně danou strukturu, jsou data nestrukturovaná. Ještě donedávna docházelo ke zpracování téměř váhradně dat strukturovaných. Ta nestrukturovaná však často nabízejí obrovský potenciál k jejich využití. Klasickým příkladem nestrukturovaných dat je lidská řeč v psané formě - ta rozhodně obsahuje spoustu informací, ale ve formě kterou je počítačově složité analyzovat. Může se jednat například o články, konverzace nebo příspěvky na sociálních sítích. Právě příspěvkům na sociální síti Twitter se věnuje i tato práce.

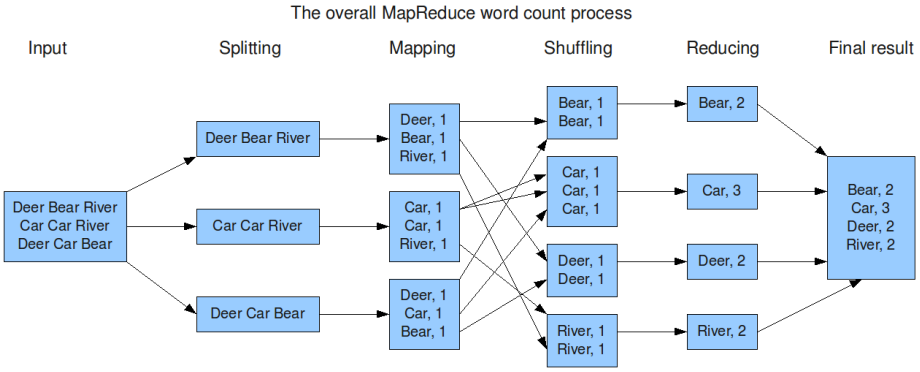
Analýzou lidské řeči se zabývá obor zvaný Natural Language Processing, obecně vytěžováním znalostí z dat pak tzv. Data Mining. Často skloňovaným pojmem je také Machine Learning, v češtině strojové učení. Pro všechny tyto obory jsou technologie pro Big Data obrovským přínosem - díky nim je možné získat opravdu cenné informace snáze a rychleji než bylo možné dříve. Například analýzou dat pohybu uživatele po webové stránce a jeho chováním můžeme odhalit nedostatky tohoto webu a jejich odstraněním zvýšit míru konverze. Hovoříme-li v kontextu sociální sítě Twitter, možnosti jsou ještě zajímavější. Na základě příspěvků a vyplněných informací jednotlivých uživatelů můžeme například odhadovat jejich volební preference nebo nabízet velice přesně cílenou reklamu. Konkrétnějším příkladem může být zajímavý projekt ze Stanfordské univerzity usilující o odhad vývoje cen akcií na základě analýzy sentimentu příspěvků z Twitteru[7]. Touto technikou se budu zabývat v následující kapitole této práce.

Technologie pro Big Data je samozřejmě možné použít i na data strukturovaná, největší využití však nabízejí při zpracování těch nestrukturovaných. Protože nestruktorovaná data přibývají výrazně rychleji[6], a protože je jejich zpracování obvykle výpočetně náročnější, jsou právě technologie pro Big Data vhodnou volbou.

1.1.2 Principy zpracování velkých dat

Jedny z prvních konkrétních technologií pro práci s Big Daty vznikaly na přelomu tisíciletí ve společnosti Google. Právě Google v roce 2004 zveřejnil článek o modelu MapReduce[8], která se stala stavebním kamenem pro většinu dalších technologií pro práci s velkými objemy dat. MapReduce vlastně popisuje dvě nezávislé funkce. První z nich je funkce Map, ve které jsou ze vstupních dat vygenerovány dvojice klíč a hodnota. Poté co je funkce Map dokončena, její výstup je použit jako vstup do funkce Reduce. Ta pak spojí vstupní data podle klíče[9].

Klíčovou vlastností MapReduce modelu je možnost paralelizace Map fáze na počítačovém clusteru. Jeden z počítačů v clusteru například přijme požadavek od uživatele. Tento počítač rozdělí vstupní data ostatním počítačům v clusteru a vyčká na provedení Map fáze těmito počítači. Výsledná data pak sám master spojí v Reduce fázi a navrátí výsledek uživateli. Distribuce co největšího množství operací a výpočtů po počítačovém clusteru je v dnešní době obecně hlavním principem fungování technologií pro zpracování velkých objemů dat. Paradoxně nemusí jít o dražší řešení než nákup jednoho supervýkoného serveru. Clustery pro práci s Big Daty jsou totiž obvykle tvořeny běžně dostupnými a relativně levnými servery.



Obrázek 1.1: Ukázka fungování MapReduce paradigmatu [10]

Především v posledních letech se objevují konkrétní komplexnější nástroje pro práci s velkými daty. Tyto nástroje obvykle obsahují i další technologie,

mimo jiné umožňující například správu a konzistenci počítačového clusteru. Tyto frameworky principielně data zpracovávají dvěma různými způsoby - jde o tzv. batchové zpracování nebo o zpracování streamové.

1.1.3 Batchové zpracování Big dat

Batchové nebo-li dávkové zpracování je vhodné především pro takové úkoly, jejichž výsledky není nutné znát ihned. Data jsou nejprve po určitou dobu shromažďována a až poté je jednorázově spuštěna úloha pro jejich zpracování. Toto zpracování obvykle zabere relativně dlouhý čas. Zmiňovaný MapReduce se využívá právě pro batchové zpracování dat.

Jedním z prvních klíčových frameworků který umožnil další vývoj v oblasti Big dat je jednoznačně open-source framework Hadoop[13]. Protože využívá MapReduce model podporuje právě batchové zpracování dat. Jeho první plná verze vyšla na konci roku 2011, ale byl využíván již přibližně od roku 2006 například ve společnosti Yahoo[11]. Je určen pro použití na počítačových clusterech složených z řádově desítek až stovek běžně dostupných serverů. Hadoop se skládá ze tří hlavních komponent - Hadoop Distributed File System, Hadoop MapReduce a Hadoop YARN. HDFS neboli Hadoop Distributed File System je distribuovaný filesystem zajišťující rozprostření dat po jednotlivých počítačích v clusteru. Klade důraz na toleranci výpadků částí clusteru. Pro zabránění ztráty dat v případě takového výpadku dochází mimo jiné k jejich replikaci na více strojů. Hadoop MapReduce je konkrétní implementace MapReduce modelu zmiňovaného dříve. Hadoop YARN pak slouží především k řízení zdrojů v clusteru, tedy například rozdělování práce jednotlivým serverům v clusteru. Nevýhodou Hadoopu je fakt, že nepodporuje proudové zpracování dat, ale pouze zpracování batchové.

V současné době jsou k dispozici i další batchově zaměřené frameworky, které často dosahují lepších výsledků než Hadoop a mimojiné obvykle umožňují vytváření úloh na vyšší úrovni. Díky ním například není nutné přímo vytvářet mapovací a reduce funkci. Nejrozšířenějím je bezesporu Apache Spark[12], kterému se budu podrobněji věnovat v následujících sekcích. Oproti Hadoopu dosahuje především díky cachování a dalším vylepšením často několikanásobně kratší doby zpracování. Krom batchového zpracování umožňuje i práci s proudy dat.

1.1.4 Streamové zpracování

V praxi často potřebujeme velké objemy dat zpracovávat v co nejkratším čase a ideálně na každý nový podnět co nejdříve zareagovat. Batchové zpracování je v takovém případě nevhodné. Řešením je již zmiňované streamové zpracování, nebo také zpracování proudů dat. Vstupem do takového programu není fixní soubor dat, ale neustálý proud dat nových. Kdykoli program obdrží nový datový objekt, zjednodušeně ho začne okamžitě zpracovávat a mezivýsledky

předávat mezi jednotlivými částmi programu nezávisle na dalších vstupech. Technologií umožňujících streamové zpracování velkých objemů dat je několik, ty nejzajímavější jsou ale Apache Spark, Apache Storm a Apache Flink. Každá z těchto technologií funguje na trochu jiném principu a je vhodná pro jiné využití.

Apache Flink je nejnovějším frameworkem, který v relativně velké míře konkuruje Sparku. Stejně jako Spark, podporuje Flink oba způsoby zpracování dat, primárně je ale zaměřený na streamy. Narozdíl od Sparku ale podporuje streamové zpracování v pravém slova smyslu a lze s ním tak dosáhnout výrazně kratší doby zpracování[14]. I právě proto se předpokládá, že v oblasti zpracování proudů dat v budoucnu předstihne Spark[15]. Jeho další výhodou je například možnost využití existujících programů pro Storm či MapReduce. Hlavní nevýhodou je momentálně fakt, že je teprve v počátcích svého vývoje a často se tak uživatel může setkat s bugy či chybějící dokumentací.

Apache Storm jako jediný ze zmiňovaných frameworků nabízí pouze streamové zpracování, i on však poskytuje pravé streamové zpracování. Jeho API je oproti Flinku více nízkoúrovňové a vývoj v něm tak může být pracnější. Obecně ale funguje na podobném principu jako právě Flink. Oproti zbývajícím dvěma technologiím Storm postrádá například tzv. Streaming Windows a další funkce[18].

Rozhodně nejrozšířenější technologií s největší komunitou a pokočilejším stupněm vývoje je Apache Spark. Tomu se podrobněji věnuji v další sekci. Pro porovnání s dalšími technologiemi ale lze říct, že v oblasti proudového zpracování se hodí především pro ty případy užití, kdy není nutné co nejrychlejší zpracování a řádově několikasekundové zpoždění nehraje roli. Je pak momentálně oproti zbylým technologiím vhodnou volbou díky svojí vyspělosti a jednoduchosti. Pokud je ale nutné opravdu real-time zpracování, je třeba volit mezi Flinkem a Stormem. Ty fungují na podobném principu, nicméně Flink má některé funkcionality, které ve Stormu chybí[18]. Flink také nabízí vysokoúrovňové API a do budoucna se jeví jako perspektivnější framework.

1.2 Apache Spark

Apache Spark je open-source framework pro distribuované zpracování dat. Vznikl na Kalifornské univerzitě v Berkeley, která ho v roce 2013 věnovala Apache Software Foundation. V rámci této nadace je Spark momentálně jedním z nejaktivněji vyvýjených projektů[19]. Poskytuje API pro jazyky Java, Scala, Python a nově také R. Jak již bylo řečeno, Apache Spark podporuje batchové i proudové zpracování dat.

Primárně je Spark zaměřený na batchové zpracování, ve kterém může Hadoop až několikanásobně překonat svou rychlostí. Toho dosahuje několika vylepšeními - především nepoužívá přímo MapReduce paradigma, které je v některých ohledech příliš svazující, ale vlastní algoritmy založené na podobných

principech ale navržené odlišným způsobem. Hadoop například výsledky většiny operací ukládá na disk, kdežto Spark často využívá cachování a operační pamět. Je pak na operačním systému, jestli se rozhodne tato data zapsat na disk[20]. Právě využívání operační paměti pak samozřejmě vede k výraznému zrychlení.

Spark narozdíl od Apache Flink a Apache Storm nenabízí plnohodnotné proudové zpracování, ale takzvané micro-batchové zpracování. Spark nereaguje na každý nový datový objekt samostatně, ale nejdříve data po zadanou dobu střádá a poté je víceméně klasicky batchově zpracuje. Tento proces je pak neustále opakován. Oproti klasickému batchovému zpracování je interval sběru dat obvykle várazně kratší. Přestože Spark při zpracování proudů dat obvykle nedosahuje takových rychlostí jako právě Flink nebo Storm, i s ním je možné dosáhnout zpracování dat v téměř reálném čase[14].

Stejně jako Hadoop využívá i Spark další technologie pro správu clusteru a distribuované ukládání dat. Jeho velikou výhodou je možnost výběrů mezi několika. Pro správu clusteru je možné využít cluster manager integrovaný přímo do Sparku, Hadoop YARN nebo Apache Mesos. Pro distribuované ukládání je pak výběr ještě rozsáhlejší, volit lze mezi již zmiňovaným HDFS, Cassandrou, OpenStack Swift, Amazon S3, Kudu nebo MapR-FS. Především při vývoji se pak hodí možnost využití pseudo-distribuovaného režimu, při kterém není žádný z těchto systémů potřeba. Spark totiž při tomto nastavení běží na jediném počítači v režimu, kdy jedno jádro odpovídá jednomu klientskému počítači v clusteru.

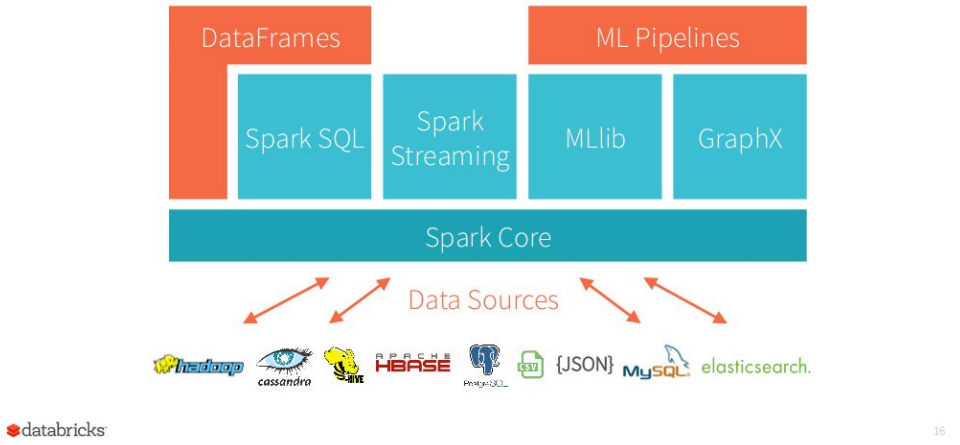
1.2.1 Komponenty frameworku

Jak znázorňuje obrázek 1.2, Apache Spark se skládá z několika hlavních komponent.

1.2.1.1 Spark core

Tou hlavní komponentou je Spark Core, který zajišťuje všechny základní funkcionality Sparku, jako jsou například plánování úloh a základní vstupní a výstupní operace. Ostatní komponenty nad ním staví. Základním pojmem je zde tzv. Resilient Distributed Dataset, nebo-li RDD. RDD je zjednodušeně řečeno téměř libovolná kolekce dat, nad kterou je možné provádět paralelní výpočty. Je navíc zajištěna jejich konzistence - při výpadku některého ze strojů mohou být ztracená data obnova pomocí informací na těch ostatních.

Kdykoli je na konkrétním RDD vykonána některá z operací umožňujících paralelní zpracování, tzv. transformace (například map, filter nebo reduce), driver nebo-li počítač řídící ostatní počítače naplánuje její zpracování na všech počítačích v clusteru. Ty poté vytvoří nová RDD s výsledky. Operace na RDD jsou ale "lazy", což znemaná, že se neprovádějí dokud na RDD neaplikujeme



Obrázek 1.2: Komponenty frameworku Spark[16]

nějakou akci. Akce je opakem transformace a jejím výsledkem není vytvoření nového RDD. Příkladem akce je vypsání obsahu RDD na standartní výstup.

1.2.1.2 Spark SQL

Další důležitou komponentou je Spark SQL, dříve nazývaný jako Apache Shark. Ten umožňuje práci s především strukturovanými daty. Zavádí také novou strukturu zvanou Data Frame (DF). Stejně jako RDD, je i Data Frame distribuovaný po clusteru. Jeho struktura je ale narozdíl od RDD pevně daná - konceptuelně odpovídá klasické tabulce v relační databázi. Vnitřně v něm dochází k optimalizacím urychlujícím operaci s ním.

Hlavní předností Spark SQL je možnost používat klasické SQL nebo HiveQL[2] dotazy, jejichž výsledky jsou uloženy právě jako Data Frame. Například komunikace s klasickou relační databází je možná pomocí JDBC/ODBC[3].

1.2.1.3 Spark Streaming

Jak již bylo řečeno, Spark neumožňuje streamové zpracování v pravém slova smyslu, ale pracuje na principu tzv. micro-batchového zpracování. To způsobuje mírné zpoždění výsledků, ty ale nepotřebujeme znát vždy opravdu ihned. Velikou výhodou je fakt, že programy určené pro streamové zpracování se ve Sparku téměř neliší od těch určených k jednorázovému spuštění.

[2] An example footnote.
[3] An example footnote.

Obrázek 1.3: Princip fungování Spark Streaming[17]

Jak ukazuje obrázek 1.4, jako zdroj dat pro Spark Streaming může posloužit například Kafka, Flume, HDFS, Amazon S3 nebo Kinesis. Přímo lze také využívat data z Twitteru bez nutnosti používání dalších knihoven. Právě Twitter poslouží jako zdroj dat pro systém na který se zaměřuje tato práce.

Obrázek 1.4: Možné zdroje dat a úložiště výsledků pro Spark Streaming[17]

I Spark Streaming zavádí novou strukturu. Nazývá se Discretized Stream nebo-li DStream. Jde o nekonečnou posloupnost nových dat reprezentovaných jednotlivými RDD. Při psaní konkrétních programů pak stačí nastavit délku intervalu, ve kterém se má opakovaně spouštět daná úloha. DStream pak v každém intervalu dané délky vrátí právě jedno RDD, které je již možné klasicky zpracovat.

1.2.1.4 MLlib

MLlib je framework určený pro strojové učení. Právě díky vlastnostem Sparku (především díky využívání operační paměti) poskytuje výrazně lepší výsledky než klasické technologie pro machine learning. Výhodou tohoto frameworku je to, že v něm jsou již naimplementovány nejpoužívanější algoritmy. Není tak nutná jejich reimplementace uživatelem.

1.2.1.5 Graphx

GraphX je ve Sparku relativně novým modulem, který je určený obecně pro práci s grafy a paralelními grafovými výpočty. Grafové výpočty získávají stále více pozornosti, jednoduchým příkladem je algoritmus PageRank, který stál u zrodu moderního vyhledávání v Google[21].

Kapitola 2

Analýza

2.1 Metody pro analýzu textu

Sentiment, ...

2.2 Požadavky na systém

Výsledný systém, jehož vytvoření je cílem této práce, se bude skládat ze tří klíčových částí - systému pro analýzu textu, databáze a webserveru zpřístupňujícího data pomocí RESTového API. Pojmu REST API se podrobněji věnuji v sekci 3.4.

První zmiňovaná část, systém pro analýzu textu, bude využívat framework Apache Spark. Jako zdroj dat bude použita sociální síť Twitter poskytující vzorek přispěvků z ní, tzv. tweetů, pomocí veřejného API. Tomu se podrobněji věnuji v kapitole 2.4. Z těchto příspěvků budou vyfiltrovány ty, které reprezentují přání jejich autorů. Kromě měření běžných statistik bude systém provádět analýzu sentimentu těchto přání. Tato analýza bude probíhat v reálném čase, respektive se zpožděním v řádu maximálně několika desítek sekund. Výsledky analýzy z této části budou persistovány ve vhodné relační databázi. Na ní bude napojen webový server zpřístupňující data pomocí RESTful API.

2.2.1 Funkční požadavky

2.2.1.1 Systém pro analýzu textu

Samotný systém pro analýzu textu musí splňovat následující požadavky:

- Získání a ukládání veškerých dat, jejichž zpřístupnění pomocí RESTového API je požadováno a jež nejsou získávána jiným způsobem

2.2.1.2 Databáze

Databáze bude poskytovat následující funkcionality:

- Agregace statistických dat dle času - u dat u kterých je to možné bude docházet k jejich seskupovaní po větších intervalech. Tím dojde ke snížení náročnosti dotazů na tato data.
- Promazávání starých a již zagregovaných dat

2.2.1.3 RESTful API

RESTful API bude poskytovat následující data:

- Jednotlivá přání a základní informace o nich, včetně jejich sentimentu a informací o jejich autorovi
- Seznam uživatelů zmíněných v konkrétním přání
- Seznam hastagů použitých v konkrétním přání
- Informace o uživatelích, kteří jsou autory některého přání nebo byli v některém zmíněni
- Seznam přání publikovaných daným uživatelem
- Seznam přání v nichž byl daný uživatel zmíněn
- Seznam přání obsahujících daný hashtag
- Statistiky nejpouívanějších hashtagů
- Statistiky nejvíce zmiňovaných uživatelů
- Statistiky o počtech již analyzovaných tweetů, anglických tweetů a z nich vyfiltrovaných přání v daném časovém intervalu
- Statistika o průměrném sentimentu v daném intervalu

Pro data pro která je to vhodné bude API umožňovat:

- Omezení počtu vrácených výsledků
- Nastavení časového intervalu
- Poskytnutí pouze některých hodnot, například pouze statistik počtu anglických tweetů namísto veškerých statistických údajů

2.2.2 Nefunkční požadavky

2.2.2.1 Kvalitativní

- **Rychlost** - analýza textu bude probíhat v reálném čase, respektivě se zpožděním v řádu maximálně několika desítek sekund
- **Škálovatelnost** - systém pro analýzu textu bude při nasazení na dostatečně výkoném počítačovém clusteru připraven pro zpracování tisíců příspěvků za sekundu

2.2.2.2 Omezující

- **Framework pro analýzu** - k analýze příspěvků z Twitteru bude použit framework Apeche Spark a jeho komponenta Spark Streaming
- **Zpřístupnění dat** - data budou zpřístupněna pomocí RESTful API. To bude odpovídat běžným praktikám používaným právě při návrhu RESTových rozhraní

2.3 Dostupné technologie

- dostupne technologie

2.4 Sociální síť Twitter

Twitter je v současnosti jednou z nejpoužívanějších sociálních sítí poskytující svým uživatelům mikroblogovací funkcionality. Umožňuje zveřejňovat 140 znaků dlouhé zprávy (tzv. tweety) a číst zprávy publikované ostatními. Účet na něm má odhadem 1,3 milardy uživatelů, denně se jich pak na Twitter přihlásí asi 100 milionů[22]. Tito uživatelé pak každý den zveřejní 500 milionů tweetů, což odpovídá přibližně 6000 tweetů každou sekundu[23].

2.4.1 Vysvětlení pojmů

Pro bližší pochopení fungování sociální sítě Twitter a analýzy příspěvků z ní pro účely této práce je potřeba vysvětlit několik základních termínů:

2.4.1.1 Tweet

Tweet je tím nejdůležitejším pojmem na Twitteru. Jak již bylo řečeno, jedná se o zprávu o maximální délce 140 znaků. Důvody pro toto omezení jsou v dnešní době víceméně historické - Twitter byl totiž už od svých počátků zamýšlen pro použití na mobilních telefonech. Protože klasické textové zprávy posílané právě pomocí mobilních telefonů jsou omezeny na 160 znaků, bylo toto omezení při vzniku Twitteru bráno v potaz. Samotný 140 znaků dlouhý

tweet by zabral většinu délky SMS zprávy, zbylých 20 znaků bylo vyhrazeno pro jméno autora zprávy[24]. Nicméně i dnes má toto omezení smysl - nutí uživatele zprávy zkracovat a zmiňovat to nejdůležitější.

Všechny tweety zveřejněné uživatelem jsou standartně viditelné pro všechny. Uživatel však má možnost nastavit si vyšší úroveň soukromí. Přímo na hlavní stránce se uživateli zobrazují příspěvky těch uživatelů, jejichž je tzv. follower.

2.4.2 Follower

Followerem nebo-li "sledujícím"jiného uživatele se člověk stane jednoduše kliknutím na tlačítko Follow na profilu vybraného uživatele. To způsobí právě zobrazování zpráv sledovaného uživatele na hlavní stránce Twitteru. Sledovaný uživatel si může zobrazit seznam uživatelů kteří ho sledují.

2.4.2.1 Retweet

Často používanou funkcionalitou Twitteru je možnost přeposlat zprávy někoho jiného nebo-li vytvořit tzv. retweet. Taková zpráva pak začíná písmeny RT označujícími právě retweet, za nimi pak následuje jméno původního autora zprávy.

2.4.2.2 Hashtag

Hashtag je řetězec začínající znakem "#". Hashtagy vkládají samotní uživatelé do jimi publikovaných tweetů. Na základě použitého hashtagu je pak možno vyhledat i ostatní zprávy, které ho obsahují. Příkladem může být hashtag #rio2016 používaný pro tweety týkající se nadcházejících Olympijských her v Brazilském Rio de Janeiru. Dostupné hashtagy však nejsou nijak omezeny, použit může být libovolný řetězec.

2.4.2.3 Mention

Stejně jako může tweet obsahovat hashtag, může v něm být zmíněný jiný uživatel Twitteru. Toto zmínění je v originále označováno jako tzv. Mention. Pro zmínění jiného uživatele je použit znak zavináče následovaný uživatelským jménem.

2.4.3 Twitter Streaming API

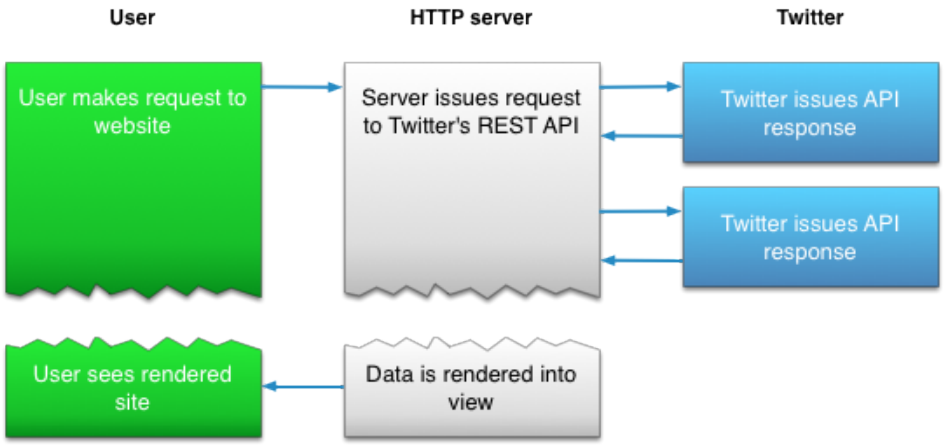
Twitter poskytuje 2 rozhraní pro přistup k jeho datům. Jde o klasické REST API s mnoha různými endpointy[4] a dále o několik streamových endpointů. Tato práce využívá stremové API, konkretně stream poskytující vzorek veřejně publikovaných tweetů[25]. Počet tweetů které toto API poskytuje není

[4]Endpoint nebo-li koncový bod je .....

přesně daný, nicméně experimentálně bylo zjištěno, že jde o přibližně 60 až 80 příspěvků za sekundu. To odpovídá asi 1% publikovaných tweetů.

Získat lze samozřejmě i přístup ke všem veřejně publikovaným příspěvkům. Nejde však již o volně dostupná API a přístup k nim je placený. Tuto službu nabízí portál Gnip.com[28]. Pro tuto práci je z pocohpitelných důvodů použito pouze veřejně dostupné API.

Pro různé případy užití se hodí využít jiné rozhraní. Klasický případ vužití RESTového API ukazuje obrázek 2.1 - uživatel navštíví webovou stránkou a chce zobrazit například údaje o konkrétním uživateli. Server tento požadavek přijme a sám odešle požadavek na API poskytované Twitterem. Ten mu odpoví, server data zpracuje a vrátí je původnímu uživateli.



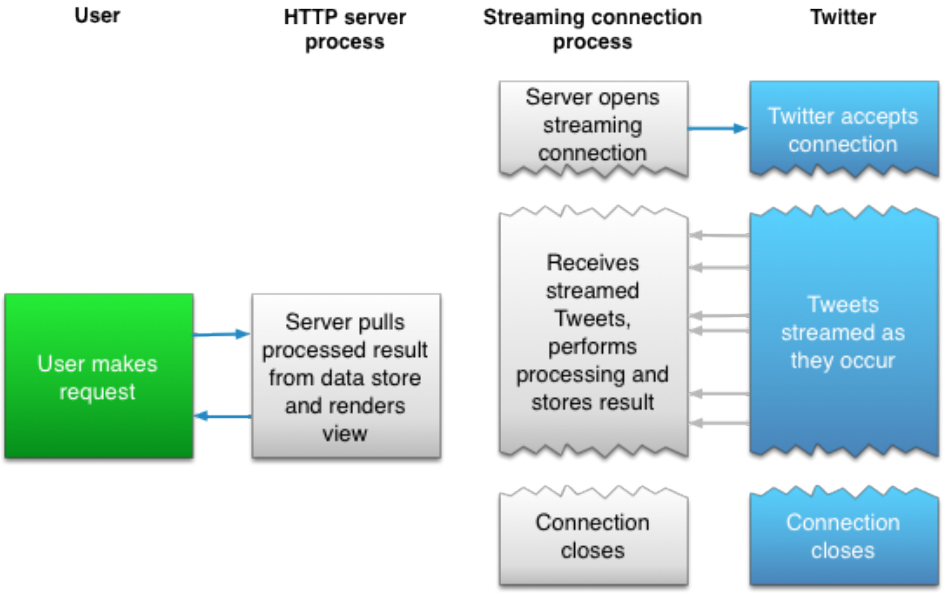
Obrázek 2.1: Schéma fungování klasického Twitter REST API[26]

V některých případech je ale vhodnější využít právě streaming API. Pokud nepotřebujeme okamžitě reagovat na jednotlivé požadavky uživatelů ale naopak chceme data zpracovávat kontinuelně, je streaming API jasnou volbou. Mimo jiné díky němu dosáhneme nižšího zatížení serveru, protože není nutné neustálne navazovat nová spojení. Tento princip je popsán na obrázku 2.2.

2.4.3.1 Twitter4j

Popisovat konkrétní parametry použitého streaming API nemá v kontextu této práce smysl. Spark streaming má totiž v sobě Twitter Streaming API zaintegrované jako jeden z možných zdrojů dat a veškerou práci s API tak řeší za uživatele. Je pouze třeba nastavit přístupové údaje k API, které je možné získat vytvořením vývojářského Twitter účtu.

Spark navíc využívá java knihovnu Twitter4j[27], která usnadňuje práci s Twitter API. Díky tomu jsou jednotlivá RDD tvořena již přímo objekty reprezentující jednotlivé tweety. Pomocí metod těchto objektů je pak možné o



Obrázek 2.2: Schéma fungování Twitter Streaming API[26]

každém tweetu snadno získat všechny dostupné informace, včetně například údajů o autorovi tweetu.

Kapitola 3

Návrh

3.1 Celkový pohled na systém

jednotlive casti, propojeni, diagram

3.2 Analýza tweetů

jak bude vypadat program ve sparku

3.3 Struktura databáze

schema, ...

3.4 Návrh API

3.4.1 REST architektura

Jedním z požadavků pro API poskytující výsledky analýzy dat je využití REST neboli Representational State Transfer architektury. Tato architektura rozhraní byla představena v roce 2000 Royem Fieldingem, spoluautorem protokolu HTTP. Proto není překvapením že REST pro přenos dat využívá právě tento protokol. Jedná se o bezestavovou architekturu, která poskytuje jednoduchý a především jednotný přístup k zdrojům. Za zdroj pak považujeme samotná data či stavy aplikace.

Každý zdroj má svoji vlastní URI[5], REST definuje čtyři základní metody pro přístup k nim. Tyto základní metody protoklu HTTP jsou označovány jako CRUD - umožňují totiž vytvoření dat (Create), jejich získání (Retrieve), aktualizování (Update) a smazání (Delete):

[5] URI

- **GET**

  Get je tou nejzákladnější metodou, která slouží k získání zdroje. Při používání webového prohlížeče se s ním setkáváme neustále - jedná se totiž o metodu sloužící jako klasický požadavek na webovou stránku. Případné parametry jsou přidány přimo do URI. Pro získání informají o uživateli Jan můžeme použít následující požadavek:

```
GET /api/user/jan
Host: www.my-web.com
```

- **POST**

  Post je metoda sloužící pro vytváření dat nových, která se používá například u klasických webových formulářů. Parametry s údaji pro nová data nejsou součástí URI ale přímo HTTP dotazu.

- **DELETE**

  Metoda Delete slouží ke smazání konkrétního zdroje definovaného pomocí URI. HTTP požadavek pro smazání uživatele Jan by se nelišil od výše uvedeného příkladu požadavku GET, pouze by bylo nutné použít metodu DELETE.

- **PUT**

  Put je metodou sloužící k aktualizaci zdroje. Jde jistým způsobem o kombinaci GET a POST - používá přímo URI již existujícího zdroje jako GET, ale v těle HTTP požadavku odesílá nová data sloužící k aktualizaci tohoto zdroje.

3.4.2 JSON

Jedním z nejčastěji používaných formátů pro přenos dat v kombinaci s RESTovým API je JSON, neboli JavaScript Object Notation. Použit bude i v rámci této práce. Skládá se z dvojic klíčů a hodnot a jeho hlavními výhodami jsou jednoduchá struktura, přehlednost a čitelnost.

Většina programovacích jazyků navíc obsahuje knhovny a funkce, které umožňují jednoduché zpracování tohoto formátu. Například v jazyce Python formát JSON v podstatě odpovídá klasickému slovníku. Následující ukázka zobrazuje zápis objektu obsahujícího pole se dvěma uživateli:

```
{
        "uzivatele": [{
                "name": "Jan Novak",
                "age": 24,
                "adresa": {
                        "mesto": "Praha 7",
```

```
                        "ulice": "Dlouha",
                        "cislo_popisne": "451"
                }
        }, {
                "name": "Karel Spacek",
                "age": 46,
                "adresa": {
                        "mesto": "Praha 4",
                        "ulice": "Siroka",
                        "cislo_popisne": "841"
                }
        }]
}
```

3.4.3 API enpointy

Kromě použití architektury REST je také nutné splnit všechny funkční požadavky definované v sekci 2.2.1. Jelikož API data pouze zpřístupňuje ale nikterak je neupravuje, veškeré API endpointy budou využívat pouze výše popsanou metodu GET. Konkrétní endpointy budou poskytovat tato rozhraní:

/wish/		
GET		Seznam přání
Bez použití následujících parametrů vrátí všechna přání za posledních 10 minut. Při použití *count* jsou ignorovány parametry *from* a *to*.		
from	timestamp	od včetně
to	timestamp	do vyjímaje
count	integer	omezení počtu vrácených přání

Příklad odpovědi:

```
{
 "wishes": [
   {
     "author": {
       "id": 2745306105,
       "profile_picture_url": "http://pbs.twimg.com/..",
       "username": "moley Jem"
     },
     "created_at": "2016-04-25 20:13:11",
     "id": 724692755319021569,
     "is_retweet": false,
     "retweet_tweet_id": 724692755319021569,
     "sentiment": 1.0,
```

```
      "tweet_text": "I just wish...that's all"
    },
    ...
    ]
}
```

/wish/{wish_id}

GET Konkrétní přání

Vrací stejné údaje jako */wish/*, ale pouze pro konkrétní přání s daným *wish_id*.

Příklad odpovědi:

```
{
  "author": {
    "id": 19748673,
    "profile_picture_url": "http://pbs.twimg.com/pro...",
    "username": "DJ Romeo Reyes"
  },
  "created_at": "2016-04-25 20:12:44",
  "id": 724692642039111680,
  "is_retweet": false,
  "retweet_tweet_id": 724692642039111680,
  "sentiment": 2.0,
  "tweet_text": "I hope the Blazers take the series to..."
}
```

/wish/{wish_id}/mentions

GET Uživatelé zmíněný v přání

Vrací seznam uživatelů zmíněných v přání s daným *wish_id*.

Příklad odpovědi:

```
{
  "mentioned_users": [
    {
      "id": 389507955,
      "profile_picture_url": "",
      "username": "EJ Gomez"
    },
    ...
  ]
}
```

/wish/{wish_id}/hashtags

GET Hashtagy použité v přání

Vrací seznam hashtagů použitch v přání s daným *wish_id*.

Příklad odpovědi:

TODO

/user/

GET Seznam uživatelů

Vrací seznam všech uživatelů, tedy autorů přání nebo uživatelů zmíněných v některém z nich.

Příklad odpovědi:

```
{
 "users": [
   {
     "id": 12,
     "profile_picture_url": "",
     "username": "Jack"
   },
   ...
   ]
}
```

/user/{user_id}

GET Konkrétní uživatel

Vrací údaje o konkrétním uživateli s daným *user_id*.

Příklad odpovědi:

```
{
 "id": 12,
 "profile_picture_url": "",
 "username": "Jack"
}
```

/user/{user_id}/wishes
GET Přání daného uživatele

Vrací seznam přání uživatele s daným *user_id*.

Příklad odpovědi: viz */wish/{wish_id}*

/user/{user_id}/mentioned_in
GET Zmínění daného uživatele

Vrací seznam přání, ve kterých byl uživatel s daným *user_id* zmíněn.

Příklad odpovědi: viz */wish/{wish_id}*

/hashtag/{hashtag}/wishes
GET Přání obsahující daný hashtag

Vrací seznam přání, ve kterých byl použit daný *hashtag*.

Příklad odpovědi: viz */wish/*

/stats/mentions
GET Seznam zmiňovaných uživatelů

Bez použití následujích parametrů vrátí seznam všech zmíněných uživatelů v posledních 10 minutách společně s počtem jejich zmínění.
Výsledky jsou řazeny sestupně dle počtu zmínění v daném intervalu.

from	timestamp	od včetně
to	timestamp	do vyjímaje
count	integer	omezení počtu vrácených uživatelů

Příklad odpovědi:

```
{
  "popular_users": [
    {
      "mention_count": 5,
      "user": {
        "id": 42562446,
        "profile_picture_url": "",
        "username": "Stephen Curry"
      }
    },
    ...
    ]
 }
```

/stats/hashtags

GET	Seznam používaných hashtagů

Bez použití následujících parametrů vrátí seznam všech hashtagů použitých v posledních 10 minutách společně s počtem jejich použití.
Výsledky jsou řazeny sestupně dle počtu použití v daném intervalu.

from	timestamp	od včetně
to	timestamp	do vyjímaje
count	integer	omezení počtu vrácených hashtagů

Příklad odpovědi:

```
{
 "popular_hashtags": [
   {
     "count": 3,
     "hashtag": "dubnation"
   },
   ...
   ]
}
```

/stats/general

GET	Seznam používaných hashtagů

Bez použití následujících parametrů vrátí obecné statistiky za posledních 10 minut.

from	timestamp	od včetně
to	timestamp	do vyjímaje
density	"3s"/"10m"/"1d"	hustota statistických dat

Příklad odpovědi:

```
{
 "stats": [
   {
     "datetime": "2016-04-25 20:43:29",
     "sentiment_average": 2.1570285426245794,
     "tweets_english": 644,
     "tweets_total": 1591,
     "wishes_total": 9
   },
   ...
   ]
}
```

Kapitola 4

Implementace

4.1 Analýza proudu dat

4.2 Databáze

4.2.1 Výběr technologií

Při výběru konkrétní databáze bylo nejdříve nutné rozhodnout, zda použít klasickou relační databázi nebo sáhnout po NoSQL databázi. NoSQL databáze, někdy také označované jako Not Only SQL, zpravidla nejsou relační a nemají pevně dané schéma. Na druhou stranu tyto databáze nabízejí větší možnosti škálování, často jsou distribuované a umožňují pojmout výrazně větší objemy dat než databáze klasické[29].

Data jsou v nich obvykle uložena ve formátu podobném JSONu, přičemž narozdíl od relačních databázích často provádíme jejich denormalizaci. To umožňuje rychlejší načtení informací z databáze, aktualizace dat je ale naopak časově náročnější. NoSQL databáze se často využívají právě v projektech pracujících s Big Daty. Jednou z nejrozšířenějších NoSQL databází je Cassandra.

Z několika důvodů jsem však pro tuto práci zvolil klasickou relační databázi. Po vyfiltrování jednotlivých přání a jejich analýze nebudu pracovat s tak velkými objemy dat, které by relační databáze nebyla schopna pojmout. Tato data navíc mezi sebou mají jasné vztahy, které je možné zanést do schématu databáze a usnadnit si tak práci s nimi. Relační databáze jsou také využívány výrazně delší dobu a jsou vývojově vyspělejší.

Z volně dostupných databázových řešení jsou nejrozšířenější především MySQL a PostgreSQL. Protože tato práce neklade na databázi žádné specifické požadavky, výběr konkrétní databáze je možné založit pouze na obecných parametrech. Protože PostgreSQL nabízí některé funkcionality které MySQL nebsahuje a protože je obecně považována za databázi na lepší úrovni[**?**], bylo PostgreSQL první volbou i pro tuto práci. Po několika neúspěšných pokusech propojit právě PostgreSQL s frameworkem Apache Spark však bylo nutné

použít MySQL.

4.2.2 Nasazení databáze

Amazon

4.2.3 Agregace dat v čase

Jedním z funkčních požadavků této práce je agregace statistických dat dle času. Té je možné dosáhnout například pomocí CRONu nebo přímo na úrovni databáze pomocí Event Scheduleru.

CRON je unixový systémový démon zajišťující provádění uživatelem definovaných příkazů v daném čase. Stačilo by tedy napsat jednoduchý script, který by se připojil k databázi a provedl sloučení dat. Tento skript by pak byl pomocí CRONu pravidelně spouštěn. Protože byl pro nasazení databáze zvolen Amazon RDS, nemáme přístup k celému systému na kterém databáze běží, ale pouze k samotné databázi. CRON proto není vhodným řešením.

Event Scheduler poskytovaný přímo jako součást MySQL je naopak ideálním řešením. Jeho využití je oproti CRONu omezenější, nicméně pro námi požadovanou agregaci dat je jeho funkcionalita postačující. Půjde navíc o čistší řešení, protože bude operace modifikující databázi řešena přímo na její úrovni.

Příkaz pro vytvoření eventu agregujího data z tabulky *stats_general_3s* do tabulky *stats_general_10m* pak vypadá takto:

```
CREATE EVENT `stats_general_3s_to_10m`
ON SCHEDULE EVERY 10 MINUTE STARTS '2016-01-01 00:01:00'
ON COMPLETION NOT PRESERVE ENABLE COMMENT
        'Aggregate general stats 3s -> 10m'
DO BEGIN
        INSERT INTO stats_general_10m (tweets_total,
                tweets_english, wishes_total,
                sentiment_average)
        SELECT SUM(tweets_total), SUM(tweets_english),
                SUM(wishes_total), AVG(sentiment_average)
        FROM stats_general_3s
        WHERE datetime > timestamp(utc_timestamp()
                - interval 10 minute);
END;
```

Tento příkaz však provádí pouze slučování dat. Pro mazání dat starých bylo potřeba vytvořit další event. Ten zajišťuje mazání dat starších než 30 minut v tabulce stats_general_3s:

```
CREATE EVENT `stats_general_3s_clean`
ON SCHEDULE EVERY 10 MINUTE STARTS '2016-01-01 00:00:00'
ON COMPLETION NOT PRESERVE ENABLE COMMENT 'Clean old data
```

```
FROM stats_general_3s' DO
        DELETE FROM stats_general_3s
        WHERE datetime < timestamp(utc_timestamp()
                - interval 1 hour) */ ;;
```

4.3 API webserver

4.4 Webová aplikace

Kapitola 5

Testování

5.1 Test API endpointů

da se web castecne povazovat jako otestovani api?

5.2 ???

Kapitola 6

Zhodnocení výsledků

6.1 Využití systému

6.2 Možná budoucí vylepšení

Závěr

Zaver

Literatura

[1] WALL, Matthew. Big Data: Are you ready for blast-off? In: BBC News [online]. 2014 [cit. 2016-04-10]. Dostupné z: http://www.bbc.com/news/business-26383058

[2] Internet Users. Internet Live Stats: Internet Usage & Social Media Statistics [online]. [cit. 2016-04-10]. Dostupné z: http://www.internetlivestats.com/internet-users/

[3] Why Big Data And The Internet of Things Are A Perfect Match. In: Datamation: IT Management, IT Salary, Cloud Computing, Open Source, Virtualization, Apps. [online]. [cit. 2016-04-10]. Dostupné z: http://www.datamation.com/applications/why-big-data-and-the-internet-of-things-are-a-perfect-match.html

[4] What is big data? Webopedia: Online Tech Dictionary for IT Professionals [online]. [cit. 2016-04-11]. Dostupné z: http://www.webopedia.com/TERM/B/big_data.html

[5] TŘÍSKA, Martin. Customer Intelligence v kontextu Big Data. Praha, 2013. Diplomová práce. České vysoké učení technické v Praze. Vedoucí práce Tomáš Bruckner.

[6] Structured vs. Unstructured Data: The Rise of Data Anarchy. In: Data Science Central [online]. [cit. 2016-04-11]. Dostupné z: http://www.datasciencecentral.com/profiles/blogs/structured-vs-unstructured-data-the-rise-of-data-anarchy

[7] MITTAL, Anshul a Arpit GOEL. Stock Prediction Using Twitter Sentiment Analysis. 2012. Dostupné také z: http://cs229.stanford.edu/proj2011/GoelMittal-StockMarketPredictionUsingTwitterSentimentAnalysis.pdf. Standford University.

[8] DEAN, Jeffrey a Sanjay GHEMAWAT. MapReduce: simplified data processing on large clusters. Google Inc., 2004.

[9] What is MapReduce. IBM [online]. [cit. 2016-04-11]. Dostupné z: https://www-01.ibm.com/software/data/infosphere/hadoop/mapreduce/

[10] MapReduce introduction. Computer Science [online]. [cit. 2016-04-16]. Dostupné z: http://www.cs.uml.edu/ jlu1/doc/source/report/MapReduce.html

[11] The history of Hadoop: From 4 nodes to the future of data. In: Gigaom: The industry leader in emerging technology research [online]. [cit. 2016-04-17]. Dostupné z: https://gigaom.com/2013/03/04/the-history-of-hadoop-from-4-nodes-to-the-future-of-data/

[12] Apache Spark: Ligthtning-Fast Cluster Computing [online]. [cit. 2016-04-17]. Dostupné z: http://spark.apache.org/

[13] Welcome to Apache Hadoop! [online]. [cit. 2016-04-17]. Dostupné z: http://hadoop.apache.org/

[14] Benchmarking Streaming Computation Engines at Yahoo!. Yahoo Engineering [online]. [cit. 2016-04-17]. Dostupné z: https://yahooeng.tumblr.com/post/135321837876/benchmarking-streaming-computation-engines-at

[15] Fast Big Data: Apache Flink vs Apache Spark for Streaming Data. Analytics, Data Mining, and Data Science [online]. [cit. 2016-04-17]. Dostupné z: http://www.kdnuggets.com/2015/11/fast-big-data-apache-flink-spark-streaming.html

[16] The 5-Minute Guide to Understanding the Significance of Apache Spark. Big Data Hadoop Blog | MapR [online]. [cit. 2016-04-17]. Dostupné z: https://www.mapr.com/blog/5-minute-guide-understanding-significance-apache-spark

[17] Spark Streaming Programming Guide. Overview - Spark 1.6.1 Documentation [online]. [cit. 2016-04-17]. Dostupné z: http://spark.apache.org/docs/latest/streaming-programming-guide.html

[18] What is/are the main difference(s) between Flink and Storm? Stack Overflow [online]. [cit. 2016-04-17]. Dostupné z: http://stackoverflow.com/questions/30699119/what-is-are-the-main-differences-between-flink-and-storm

[19] The Apache Software Foundation Announces Apache Spark as a Top-Level Project. The Apache Software Foundation [online]. [cit. 2016-04-17]. Dostupné z: https://blogs.apache.org/foundation/entry/the_apache_software_foundation_announces50

[20] What is the difference between Apache Spark and Apache Hadoop (Map-Reduce) ? Quora [online]. [cit. 2016-04-17]. Dostupné z: https://www.quora.com/What-is-the-difference-between-Apache-Spark-and-Apache-Hadoop-Map-Reduce

[21] Getting started with Apache Spark GraphX – Part 1. PhData [online]. [cit. 2016-04-17]. Dostupné z: https://phdata.io/getting-started-with-apache-spark-graphx-part-1/

[22] By The Numbers: 170+ Amazing Twitter Statistics. DMR [online]. [cit. 2016-04-19]. Dostupné z: http://expandedramblings.com/index.php/march-2013-by-the-numbers-a-few-amazing-twitter-stats/

[23] Twitter Usage Statistics. Internet Live Stats: Internet Usage & Social Media Statistics [online]. [cit. 2016-04-19]. Dostupné z: http://www.internetlivestats.com/twitter-statistics/

[24] Twitter Basics: Why 140-Characters, And How to Write More. Social Times: Covering the world of social media [online]. [cit. 2016-04-19]. Dostupné z: http://www.adweek.com/socialtimes/twitter-basics-why-140-characters-and-how-to-write-more/442608

[25] GET statuses/sample. Twitter Developers [online]. [cit. 2016-04-20]. Dostupné z: https://dev.twitter.com/streaming/reference/get/statuses/sample

[26] The Streaming APIs Overview. Twitter Developers [online]. [cit. 2016-04-20]. Dostupné z: https://dev.twitter.com/streaming/overview

[27] Twitter4J - A Java library for the Twitter API [online]. [cit. 2016-04-20]. Dostupné z: http://twitter4j.org/en/index.html

[28] Gnip: Unleash the Power of Social Media [online]. [cit. 2016-04-20]. Dostupné z: https://gnip.com/

[29] NOSQL Databases [online]. [cit. 2016-04-26]. Dostupné z: http://nosql-database.org/

Příloha **A**

Seznam použitých zkratek

Item1 foo

Item2 bar

Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `exe` ..... adresář se spustitelnou formou implementace
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
  - `thesis.ps` ..... text práce ve formátu PS