Jihočeské univerzita v Českých Budějovicích
Přírodovědecká fakulta

Zpracování dat v prostředí Hadoop

Bakalářská práce

Autor: Jiří Burda

Školitel: doc. Ing. Ladislav Beránek, CSc. MBA.

České Budějovice 2018

ZMĚNA

Jihočeská univerzita v Českých Budějovicích
Přírodovědecká fakulta

ZADÁVACÍ PROTOKOL BAKALÁŘSKÉ PRÁCE

Student: .....Jiří Burda.....
(jméno, příjmení, tituly)

Obor – zaměření studia: Aplikovaná informatika.....

Katedra/ústav, kde bude práce vypracovávána: Ústav aplikované informatiky.....

Školitel: Doc. Ing. Ladislav Beránek, CSc. .....
(jméno, příjmení, tituly, u externího š. název a adresa pracoviště, telefon, fax, e-mail)

Garant z PřF:
.....
(jméno, příjmení, tituly, katedra – jen v případě externího školitele)

Školitel – specialista, konzultant: .....
(jméno, příjmení, tituly, u externího š. název a adresa pracoviště, telefon, fax, e-mail)

Téma bakalářské práce: .....

Zpracování dat v prostředí Hadoop .....

Cíle práce:

Analýza velkého množství dat se stává nezbytným předpokladem konkurenceschopnosti podniků. Jednou z platforem, kterou lze využít pro zpracování velkých objemů dat je prostředí Hadoop. Toto prostředí je spojeno s dalšími aplikacemi pro tvorbu dat a dále umožňuje využití skriptovacích jazyků jako je Python nebo jazyk R. Cílem práce je zprovoznění instance Hadoop na vybraných serverech, provedení analýzy velkých objemů dat (pro clickstream analýzu na Webu a pro analýzu sentimentu z dat z Twitteru) s využitím prostředí Hadoop případně jeho dalších návazných aplikací. Závěrem bude provedena diskuze kladů a záporů daného prostředí.

Základní doporučená literatura:

GARRY TURKINGTON. Hadoop beginner's guide: learn how to crunch big data to extract meaning from the data avalanche. 1. publ. Birmingham, UK: Packt Publishing. ISBN 978-184-9517-300.

PERERA, Srinath a Thilina GUNARATHNE. Hadoop MapReduce cookbook: recipes for analyzing large and complex datasets with Hadoop MapReduce. Birmingham: Packt Pub., 2013, iv, 284 p. Community experience distilled. ISBN 978-1-84951-728-7.

DEAN, Jared. Big data, data mining, and machine learning: value creation for business leaders and practitioners. Hoboken, New Jersey: Wiley, 2014, 1 online zdroj (289 pages). ISBN 978-1-118-92069-5.

Financování práce: .....

Vedoucí práce: .....podpis: .....

U externích vedoucích fakultní garant práce: .....podpis: .....

Garant oboru bak. studia, pokud je obor zajišťován jinou katedrou/ústavem, než ze které je školitel (nepožaduje se u oboru biologie): .....podpis: .....

Vedoucí katedry/ústavu, kde bude práce vypracovávána: .....podpis: .....

Případný souhlas vedoucího ústavu AV: .....podpis: .....

V Českých Budějovicích dne 14. 12. 2015 .....Podpis studenta: .....

Bibliografické údaje

Burda Jiří., 2018: Zpracování dat v prostředí Hadoop
Data processing in the Hadoop environment. Bc. Thesis, in Czech. - [45] p., Faculty o Science, The University of South Bohemia, České Budějovice Czech Republic.

Anotace

Tato bakalářská práce se zabývá zpracování dat v prostředí Hadoop. V teoretické části je představena architektura Apache Hadoop, distribuovaný souborový systém HDFS, paralelní zpracování dat pomocí MapReduce a dalších nástrojů Hadoop. V praktické části je popsána konfigurace Hadoop a spuštění na vybraném serveru. Následné otestování jeho funkčnosti na vzorové úloze „wordcount“. Závěrem práce budou získána data z Twitteru a provedena jejich analýza.

In English

This bachelor thesis deals with data processing in Hadoop environment. The theoretical part introduces the Apache Hadoop architecture, the distributed HDFS file system, parallel data processing with MapReduce and other Hadoop tools. The practical part describes the Hadoop configuration and run on the selected server. Subsequent testing of its functionality on the exemplary "wordcount" task. In the end, the data will be retrieved from Twitter and analyzed.

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích dne _____.

_____

Jiří Burda

Obsah:

1. Úvod a cíle práce

1.1 Úvod

V dnešní rozvinuté době, kdy lidé po celém světě používají různé sociální sítě a platformy, roste počet generovaných dat. Každou vteřinu je generováno nesmírné množství nestrukturovaných informací. Uživatelé těchto sítí diskutují nad různými problémy, píší své názory, rozebírají aktuální dění. Zpracováním těchto rozsáhlých dat je možno získat potřebné informace, které mohou být použity v různých sférách jako je podnikání, recenze nových produktů, volby atd. Tyto informace jsou ve většině případech v textové podobě. Aby bylo dosaženo co nejefektivnějšího zpracování v co nejlepší čase a s nízkými provozními náklady, je vhodné zvolit paralelní zpracování dat.

1.2 Cíle práce

Zpracování a analýza nezměrného množství dat se v dnešní době stává nutností pro konkurence schopnost podniků. Jednou z možností, jak tyto data zpracovávat, je platforma Hadoop. V této práci bude nastíněna instalace a konfigurace platformy Hadoop na vybraném serveru. Funkčnost bude ověřena testovací úlohou „WordCount“. Dále bude popsána instalace a konfigurace Apache Flume, Hive a vytvoření app na Twitteru. Závěrem práce se provede analýza dat z Twitteru.

2. Apache Hadoop

Apache Hadoop je jeden z výtvorů Apache Software Foundation. Toto výpočetní prostředí umožňuje práci s rozsáhlými daty. Je postaven nad distribuovaným souborovým systémem, který byl inspirován již existujícm souborovým systémem GFS od Googlu. Dále využívá MapReduce jako programovací paradigma. Není to sice nová koncepce, ale Hadoop je první framework, který dokázal MapReduce využít pro mnohem obsáhlejší soubor úloh [1].

2.1 Vznik a historie Apache Hadoop

Hadoop vychází z projektu Apache Nutch, jenž je open source vyhledávací webový nástroj. Po zveřejnění dokumentace ke GFS v roce 2003 a následném vytvoření MapReduce Googlem v roce 2004, začali vývojáři Nutch s vlastní verzí MapReduce a souborového systému NDFS (Nutch Distributed File System). To vše předcházelo vzniku samostatného frameworku zvaný Hadoop. K reálnému nasazení však došlo až v roce 2008 po spojení s firmou Yahoo!. Byl vytvořen cluster sestavený z 10 000 stanic. V dnešní době využívají tuto platformu společnosti, jako jsou Twitter, IBM nebo Facebook [2].

2.2 Výhody a nevýhody Apache Hadoop

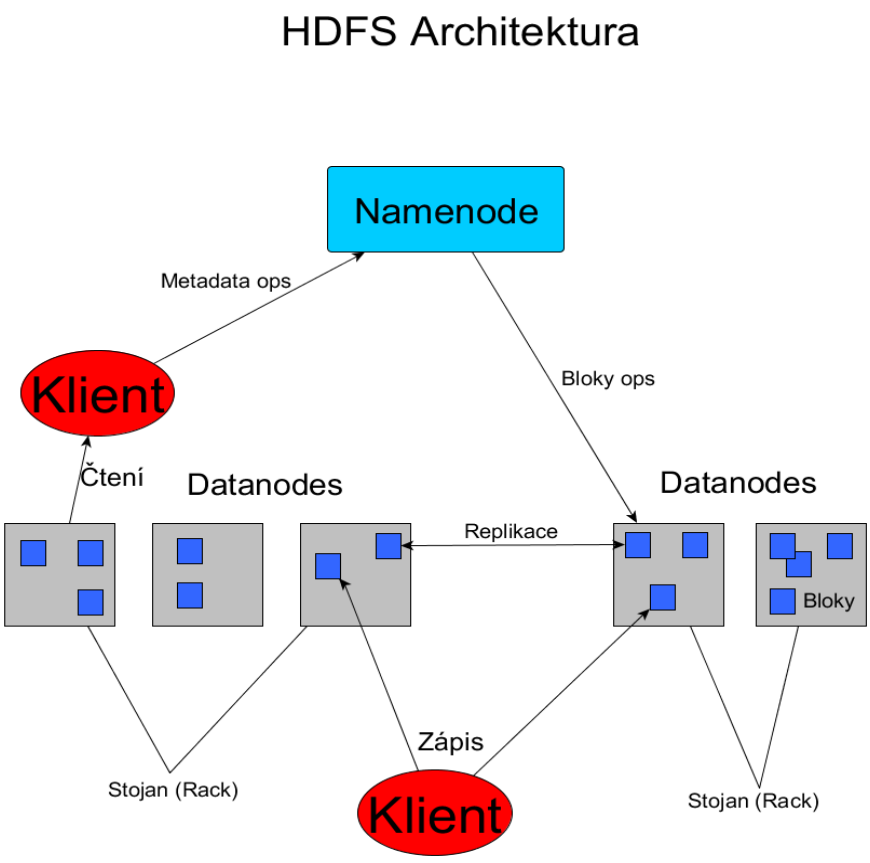
V dřívější době nebyl Hadoop zastoupen ve větších podnicích, jako je například Google, protože se nejedná o celistvou technologii, nýbrž o sjednocení více dílčích částí, které zajišťují různé účely. Udržení vzájemné kompatibility je proto velice náročné, protože oblast velkých dat se neustále vyvíjí, a to znamená i časté úpravy těchto různých částí. Tato nevýhoda byla vyřešena Hadoop distribucí. [3] Řada firem, dodávajících řešení na big data, si vytvořila svoje vlastní Hadoop distribuce s různými odlišnostmi, které zákazník potřebuje [4].

Hlavní výhodou je možnost provozování Hadoop v rámci clusteru, což je více vzájemně propojených serverů, na kterých jsou prováděny distribuované výpočty. To je hlavní rozdíl od relačních databází a datových skladů. Data jsou kopírována na různé servery, a pokud dojde k poškození některé z kopií, Hadoop zajistí přesun na jiný volný server v clusteru. Podobně se řídí i výpočty nad daty. Tyto operace jsou prováděny paralelně na více serverech. Dojde-li k chybě, tak je výpočet přesunut jinam a zopakován. Toto zajišťuje vysoký výpočetní výkon celého řešení. Na servery nepotřebujeme výkonné superpočítače, ale postačí levný

hardware. Proto také bývá cena za uložení dat výrazně nižší než u datových skladů nebo relačních databází [3].

3. Hadoop Distributed File System – HDFS

HDFS (Hadoop Distributed File Systém) - jedná se o distribuovaný souborový systém, který nám umožňuje zpracovávat gigabajty až terabajty dat. Tento systém byl navržen tak, aby bylo možno provést co nejvíce efektivní skladování a načítání dat. Jde o snahu co nejméně zapisovat do datasetu, ale o co nejvíce čtení z něj. HDFS není vhodný pro aplikace, které vyžadují okamžitý přístup k datům. Je to způsobeno vysokou propustností tohoto systému. Není zde ani podpora vícenásobného přístupu k souborům najednou a nová data, která se přidávají do souborů, se zapisují vždy nakonec [5].



Obrázek 1:

HDFS Architektura (Zdroj: [7])

3.1 Replikace

Souborový systém ukládá soubory jako posloupnosti stejně velkých bloků. A aby se vytvořila určitá odolnost vůči chybám, tak se tyto bloky replikují, tzn., ukládají se ve více kopiích. Pro každý soubor lze nastavit velikost bloku a faktor replikace (počet kopií). Implicitně však zůstává nastavený na velikost bloku 64 MB a faktor replikace = 3. HDFS se liší od ostatních distribuovaných systémů tím, že vytváří vlastní strategii, jak umístit repliky. Cluster se rozdělí na racky (jakési sady) a do nich umisťuje repliky. O vyváženost bloků se nám stará tzv. balancer.

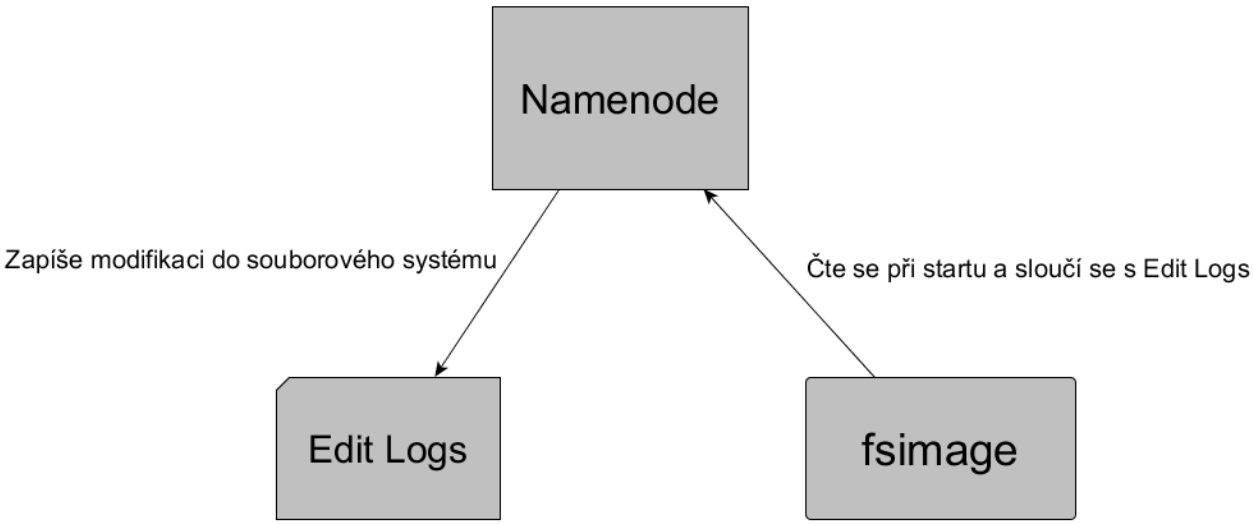
3.2 Fragmentace

HDFS blok je složen z clusterů souborového systému, na kterém běží. Nejlepším řešením je tedy zvolit velikost clusteru u souborového systému nodu takovou, která se bude co nejvíce shodovat s velikostí bloku v HDFS. Doba výpočtu záleží na velikosti bloku. Čím větší, tím bude delší doba výpočtu. Obráceně to platí u přípravy souborů. Ta je náročnější, když je využito více bloků. Nejlepším řešením je nastavení ideálního poměru délky přípravy a délky výpočtu na daném hardwaru, díky němuž se zvolí optimální velikost bloku [6].

3.3 NameNode

Jedná se o hlavní jednotku, která řídí HDFS. Udržuje informace o datech, která se ukládají do souborového systému Hadoopu. Avšak sám data neukládá. Primárně se udržují informace o stromové struktuře a v ní umístění souborů. Také uchovává metadata o právech pro užívání souborů a informace o jejich vlastnictví. NameNode vše ukládá do operační paměti RAM, protože tím se zvýší výkon NameNodu a pro bezpečnost se ještě kopírují na lokální disk. Metadata se ukládají na disk do dvou různých souborů. Prvním je *fsimage*, ve kterém se nachází obraz souborového systému při startu NameNodu. A druhý je *Edit Logs*, jenž uchovává sled změn vykonaných v souborovém systému po startu NameNode [8].

Dalším úkolem NameNodu je interakce s uživatelem. Umožňuje soubory mazat, kopírovat, přesouvat nebo jen zjišťovat jejich umístění ve stromové struktuře [9].



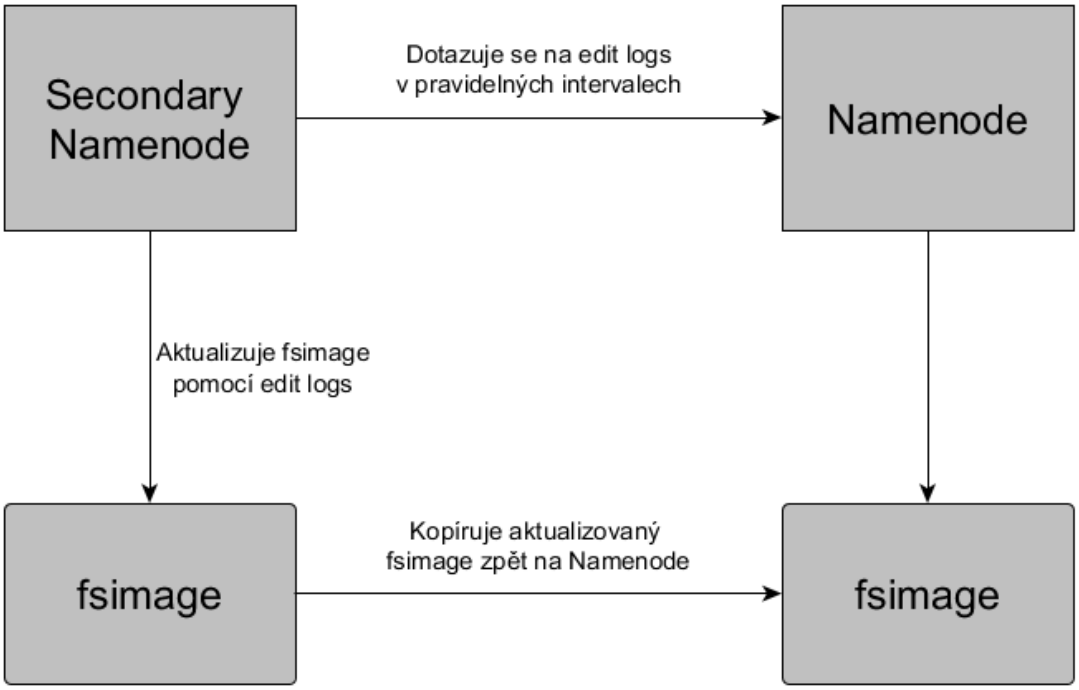
Obrázek 2:

NameNode (Zdroj: [10])

3.4 Secondary NameNode

Může se zdát, že Secondary NameNode slouží jako záložní zdroj k NameNodu, není to však pravda. Je to spíše takový „pomocník“, který usnadňuje práci NameNodu při restartování. Secondary NameNode se v pravidelných intervalech dotazuje na *Edit Logs* od NameNodu a aplikuje jej na *fsimage.* Jakmile má nový *fsimage*, zkopíruje jej zpět do NameNodu. Tento nový *fsimage* se využije při restartu NameNodu, což sníží čas potřebný ke spuštění.

Smyslem Secondary NameNode je tedy vytvářet jakýsi kontrolní bod, proto je taky znám jako „checkpoint node“. Všechna data, vytvořená po tomto kontrolním bodu, budou při restartu NameNodu ztracena [10].



Obrázek 3:

Funkce Secondary NameNode (Zdroj: [10])

3.5 DataNode

Známý jako SlaveNode je základním stavebním kamenem HDFS. DataNody zajišťují ukládání dat, která jsou v nešifrované podobě. Tato data jsou rozdělena na několik částí, a proto je nezbytné používat tzv. splittable kompresní algoritmy, jako je Parallel LZO od Cloudery. Nyní nastává problém v rychlosti, protože tyto algoritmy nejsou zrovna nejrychlejší, a to zpomaluje celý systém [11].

Aby byla zajištěna integrita čtení a zápisu, jsou ke každému bloku přiřazena metadata, která obsahují kontrolní součet. Aby NameNode měl přehled o správném fungování DataNodu a o blocích, které spravuje, tak se každé 3 vteřiny odesílá z DataNodu na NameNode report *Heartbeat*. Ten obsahuje informaci o tom, že DataNode funguje správně. Poté se ještě odesílá report *BlockReport*, který obsahuje informace o všech blocích, které spravuje.

DataNode nepotřebuje tolik výkonný hardware jako NameNode, ale zase jsou vyšší požadavky na diskový prostor, kam se ukládají data, mezi-výsledky úloh a také konečné výsledky úloh [9].

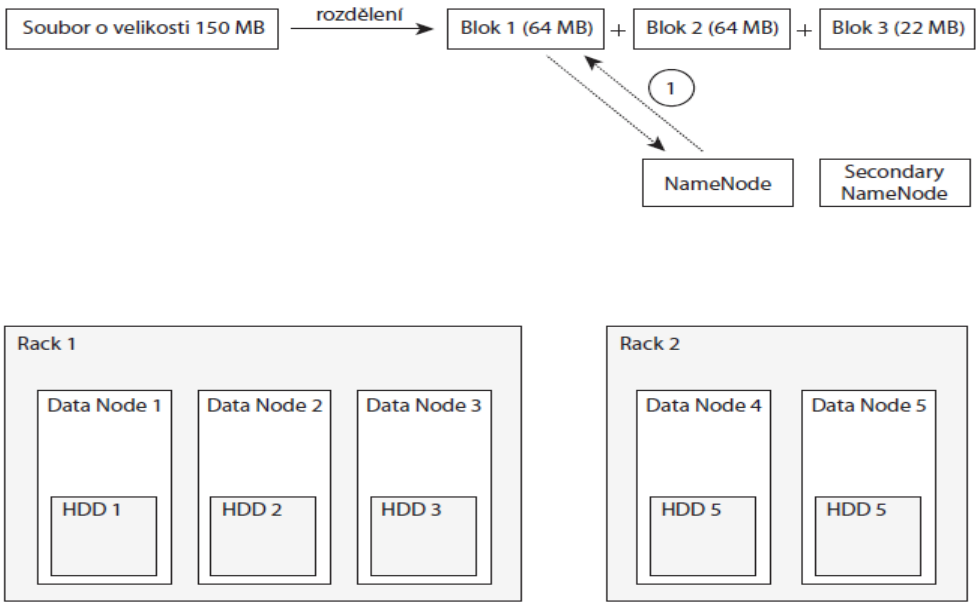
3.6 Práce s daty

3.6.1 Bloky Dat

HDFS rozděluje data do bloků. Jedná se o jednotky o velikosti 64 MB, což je maximální velikost. Proto se soubory, které se ukládají, musí rozdělit do blok velikostí a poté jsou nezávisle rozmístěny po celém clusteru [12].

3.6.2 Zápis dat

Zápis dat bude demonstrován na následujícím příkladu. Soubor o velikosti 150 MB se rozdělí podle defaultního nastavení na bloky o velikosti 64 MB, pokud není jinak nastaveno uživatelem. NameNode je informován od DataNodu pomocí reportu, které nody nejsou zaplněné a mohou se do nich uložit dané bloky. Jakmile NameNode ví, kam se můžou bloky uložit, pošle tuto informaci HDFS klientovi.

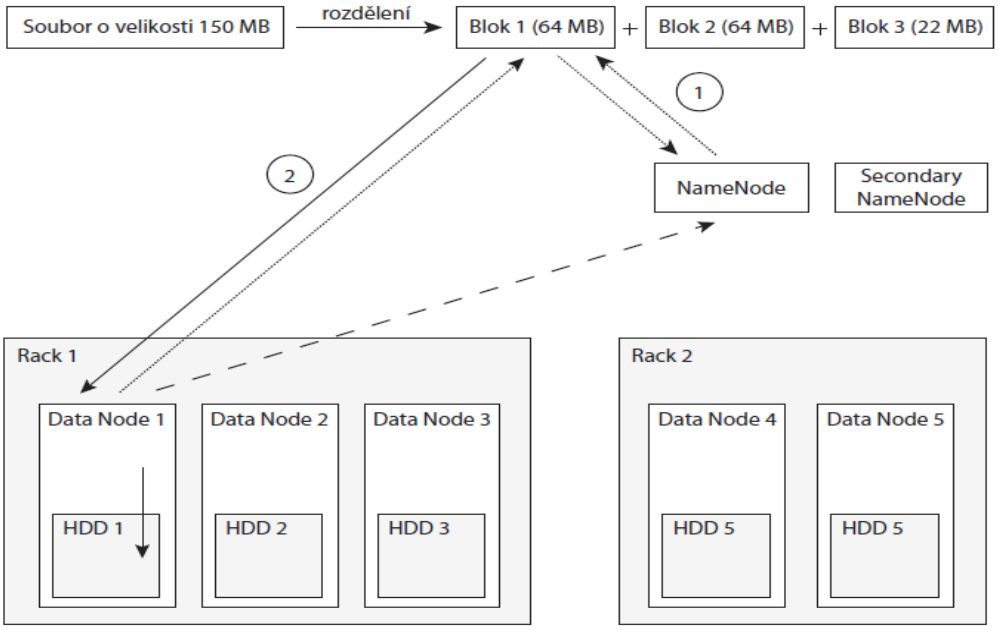


Obrázek 4

Rozdělení souborů a dotaz na zápis dat (Zdroj: [9])

Klient po obdržení této informace začne data posílat. DataNode odesílá zpět informaci o každém úspěšně zapsaném paketu. Velikost paketu se rovná 64 kB. Po uložení všech dat na

disk DataNodu se musí odeslat hlášení NameNodu, že byl uložen nový blok. Toto můžeme vidět na obrázku č. 5 [9].



Obrázek 5

Zápis dat do HDFS (Zdroj: [9])

3.6.3 Přístup k datům

Existuje mnoho různých možností, jak k HDFS přistupovat pomocí aplikací. Pro aplikace naprogramované v Javě a v C nabízí HDFS API rozhraní. Existuje také možnost přistupovat pomocí webového prohlížeče. Nejjednodušším řešením, při znalosti linuxových příkazů, je k zajištění interakce mezi uživatelem a HDFS terminálové rozhraní (FS Shell).

3.6.4 Mazání a obnova dat

V systému HDFS také existuje adresář koš. Když uživatel nebo aplikace chtějí odstranit soubor ze systému HDFS, nedojde k jeho úplnému odstranění, ale klient tento soubor přejmenuje a přesune do výše jmenovaného koše. Koš obsahuje poslední kopii souboru, než byl odstraněn. Tyto soubory se v koši uchovávají 6 hodin. Data lze rychle obnovit, pokud se pořád v koši nacházejí. Avšak po uplynutí časového limitu je NameNodem odstraněn záznam ve jmenném systému HDFS, a také jsou uvolněny bloky, ve kterých byl tento soubor umístěn [2].

4. MapReduce

Jedná se o programový model a techniku pro zpracování distribuovaných výpočtů založené na Javě. Nejvíce je využitelný pro zpracování datasetů v řádech terabytů a petabytů. MapReduce se skládá ze dvou hlavních částí, jež jsou Map a Reduce [13].

4.1 Map a Reduce

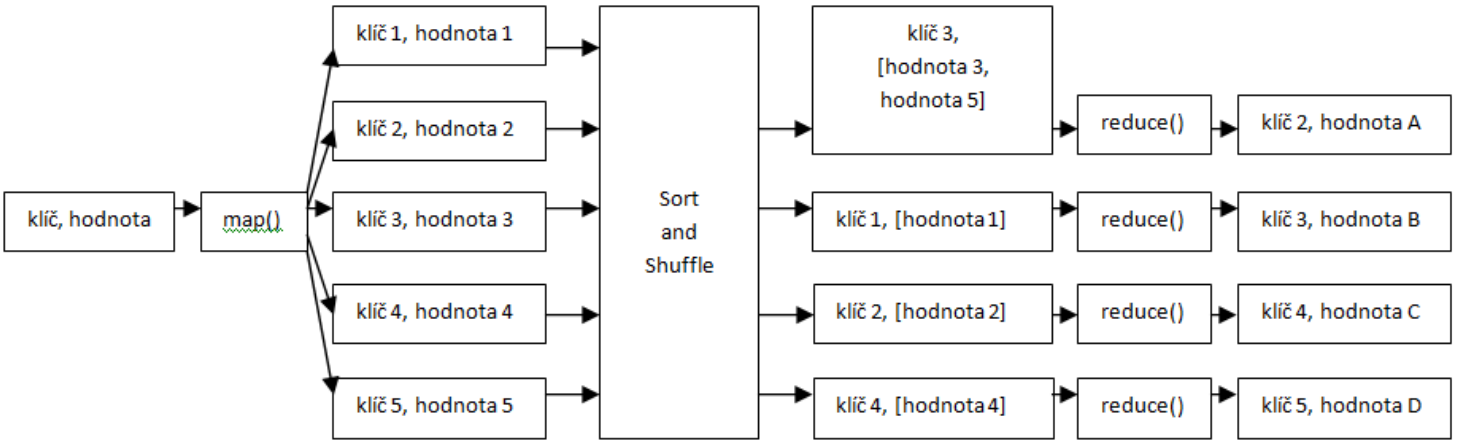
Při spuštění MapReduce je paralelně prováděno několik úloh Map a Reduce. Vstupní data jsou rozdělena na několik částí a přiřazena k výpočetnímu uzlu. Tímto vznikne několik seznamů. Poté uzly spustí paralelně funkci Map pro každý prvek ze seznamu. Výstupy jsou z uzlů shromážděny a seskupeny podle daného klíče. Mezi uzly jsou rozděleny skupiny podle hodnot klíčů a paralelně se provedou funkce Reduce pro každý seznam. Výsledky jsou posbírány a uloženy na výstup.

Map a Reduce jsou tedy definovány nad daty dvojicemi „klíč:hodnota“.
Map (k1, v1) → list (k2, v2) – vstupy (k1:v1) jsou zpracovány funkcí Map. Z výsledků je vytvořen pro každý vstup seznam výstupů (k2:v2)
Reduce (k2, list (v2)) → list (k3, v3) – výstupní hodnoty funkcí Map jsou seskupeny podle klíče. Na takovéto seznamy hodnot, pro každý jednotlivý klíč, je použita funkce Reduce, která vytvoří seznam výstupních hodnot [14].

4.2 Shuffle a Sort

Tento proces je součástí obou funkcí. Jak funkce Map, tak funkce Reduce. Princip spočívá v setřídění výstupů z mapovací funkce a přesunutí na vstup funkce redukční.

Shuffle začíná pracovat, jakmile mapovací funkce vyprodukuje první výstupy. Ty se ukládají do bufferu o velikosti 100 MB v paměti RAM. Poté, co je buffer naplněn z 80 % své kapacity, začnou se data přesouvat na disk. Před uložením na disk jsou data rozdělena podle toho, do jakého reduceru patří. Reducer provádí výpočty a redukuje. Všechny menší soubory jsou postupně spojovány, dokud nevznikne jeden velký. Poté, co se tak stane je spuštěna samotná reduce (), část definovaná uživatelem. Výsledky jsou ukládány do HDFS [5].



Obrázek 6

Fáze MapReduce se vstupy a výstupy (Zdroj: [14])

4.3 JobTracker

Veškeré výpočty provedené na Hadoop clusteru jsou řízeny JobTrackerem. JobTracker je spuštěn na MasterNodu (NameNod). Tento démon rozhoduje, jaké konkrétní soubory se budou zpracovávat a jaký nod toto zpracování vykoná. Klasicky se data zpracovávají na nodu, který je vlastní a je co nejblíže klientovi. Jde o redukci síťového provozu a o zmapování výpočetních možností TaskTrackerů. Když TaskTracker nefunguje optimálně, je restartován.

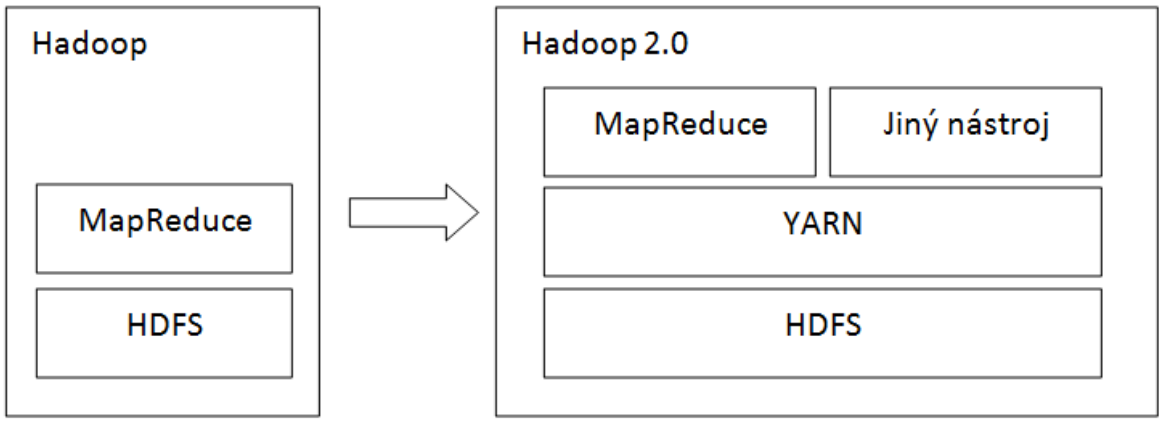
4.4 TaskTracker

Dalším démonem, který však běží na SlaveNodech (DataNodech) je TaskTracker. Ten exekuuje úlohy. JobTracker odesílá TaskTrackeru redukční nebo mapovací úlohy a TaskTracker odpovídá zpět reporty (heartbeaty), že jsou online a také upozorňuje na volné sloty pro redukční a mapovací úlohy [5].

4.5 YARN

YARN (Yet Another Resource Negotiator) je samostatná vrstva, která se objevila ve verzi Hadoopu 2.0. Jedná se o mezivrstvu HDFS a MapReduce. YARN byl vyvinut k tomu, aby se mohli používat i jiné nástroje na zpracování dat, než je MapReduce.

YARN obsahuje globální uzel, který se označuje Resource Manager. Další částí YARNu je Application Master. Ten se stará o běh aplikací, zodpovídá za komunikaci s Node Managery a kontroluje spouštění jednotlivých Tasků. Resource Manager tvoří dvě hlavní části – Plánovač a Aplikační manažer. Plánovač alokuje zdroje pro aplikace a Aplikační manažer monitoruje příjem nových MapReduce programů a jejich správné zařazení. Na všech strojích také běží, výše zmíněný Node Manager. Jedná se o agenta, který spravuje aplikační kontejnery, monitoruje stav dostupných prostředků stroje a podává informace Resource Manageru. Ten poté přerozděluje nové úkoly [15].



Obrázek 7

YARN (Zdroj: [9])

5. Další nástroje Hadoop

V Hadoopu je pro práci s daty možnost využít také celou řadu nástrojů, které mohou používat rozdílné komponenty a analytické přístupy.

5.1 Hive

Hive byl speciálně vyvinut pro účely dotazování se nad daty, která jsou uložena v distribuovaném datovém skladu v Hadoopu. Hive používá jazyk zvaný HQL (Hive Query Language). Jedná se o jazyk, který je velice podobný známému jazyku SQL. Navzdory podobnosti však HQL nepodporuje plně specifikaci SQL.

Největší výhodou u Hive je, že se data nemusí převádět na speciální formát, ale mohou zůstat ve zdrojových formátech. Dotazy HQL se překládají na MapReduce úlohy [16].

5.2 Pig

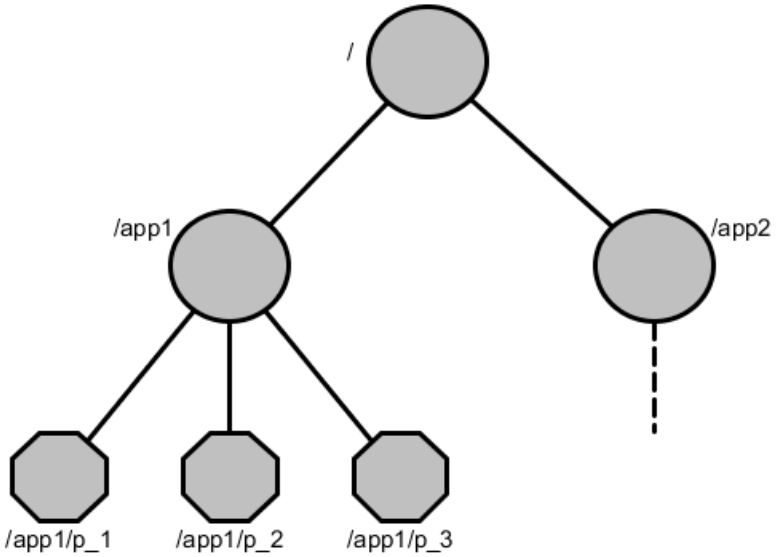
Jedná se o platformu, která provádí analýzu velkých datových souborů. Apache Pig je tvořen funkcionálním jazykem, jež popisuje datový tok a prostředí, ve kterém se spouští aplikace v něm napsané. Tento funkcionální jazyk se nazývá Pig Latin.

Mezi hlavní vlastnosti aplikací Pig patří podpora strukturální přístupnosti paralelizace, díky ní je možné zpracovávat velké soubory dat. Mezi klíčové vlastnosti Pig Latin patří snadné programování, vysoká optimalizace a rozšiřitelnost [17].

5.3 ZooKeeper

ZooKeeper má neodmyslitelnou funkci ve fungování Hadoopu. Má na starosti spolupráci všech prvků v Hadoopu. Jedná se o službu, která poskytuje distribuovanou synchronizaci dat a další skupinové služby, které jsou také synchronizované. Dále umožňuje uchovávat informace o konfiguraci a pojmenování. Také zjednodušuje správu serverů, na kterých je Hadoop spuštěn a minimalizuje riziko špatného nastavení mezi servery.

ZooKeeper využívá sdílený hierarchický jmenný prostor datových registrů, aby mohl spravovat distribuované procesy mezi sebou. Tyto registry jsou známé pod názvem *znodes*. Registry jsou vždy dostupné s velkou propustností a nízkou latencí [18].



Obrázek 8

Hierarchický jmenný prostor (Zdroj: [19])

5.4 Cassandra

Jedná se o NoSQL (klíč – hodnota) úložiště. Cassandra je vhodná u takových aplikací, které vyžadují rychlý přístup k datům do/z Hadoopu. Databáze Cassandra jsou tím nejlepším řešením, pokud je potřeba vysoká dostupnost a neomezená horizontální škálovatelnost, bez omezení výkonu.

Latence za konzistenci a konzistence za latenci. Tím se může řídit administrátor Cassandry. Může určovat individuální míru konzistence ke konkrétním skupinám záznamů. Všechny záznamy jsou uloženy v několika stejných kopiích a nacházejí se na různých serverech. Aby byla garantována určitá úroveň konzistence, musí být splněny tři volby redundance:

- Kopie dat jsou rozmístěny na určitý počet počítačů.
- Úspěšně vytvořené repliky pro dokončení zápisu jsou konfigurovatelné. Data mohou být považována za zapsaná i před dokončením stanoveného počtu replik.
- U záznamů je i čas jejich vytvoření. Nejvíce aktuální stav mají nejmladší repliky.

Také zde rozhoduje Dirichletův princip. Při celkových dvaceti replikách, sedmnácti replikách při zápisu a čtyřech replikách při čtení, je zaručeno, že se čtení alespoň v jednom případě střetne se zápisem, protože sedmnáct plus čtyři je více než dvacet. To zaručuje konzistenci.

Naopak, po změně počtu replik při zápisu na osm, se může čtení minout se zápisem a tím konzistence není zaručena, protože osm plus čtyři je méně než dvacet.

Aby se Cassandra v clusteru „neztratila“ jsou záznamy, které identifikuje klíč, převedeny na hash. Ve všech uzlech bývá určitý rozsah hashů. Tento počet hashů je přidělen deterministickým algoritmem.

Cassandra je ideální na triviální vyhledávání a ukládání dat, ale standartní SQL přístup ji převyšuje ve svoji univerzálnosti. Nejlépe je využitelná při:

- Velkém počtu paralelně zpracovávaných dotazů a dat.
- Odlišnosti hardwaru a softwaru na serverech. Databáze musí pracovat v heterogenním prostředí.
- Bez nutnosti zapamatování databázových schémat. Proto se volí úložiště typu key – value.
- Bez licenčních poplatků. Možnost úpravy zdrojových kódů.

Cassandra se nikdy nestane rovnocenným soupeřem proti klasickým SQL databázím, protože:

- Není slučitelná s SQL.
- Nepodporuje složitější vyhledávání dat a komplikovanějšího zpracování dat.
- Nepodporuje software pod platformou LAMP [20].

5.5 Jaql

Jaql je deklarativní dotazovací jazyk většinou v jazyce JSON (JavaScript Object Notation). Existuje zde i podpora aplikací programovaných v Javě, JavaScriptu, Perlu, Pythonu a dalších. Vychází také z jazyků jako jsou XQuery, SQL, Lisp a Pig.

Umožňuje práci s daty v HDFS, konkrétně spojovat, vybrat, agregovat a filtrovat. Nebo je možnost vytvoření vlastních funkcí. Mezi jeho výhody patří flexibilita a podpora zdrojů CSV, XML nebo čistě jenom obyčejných souborů [21].

5.6 Mahout

Apache Mahout je knihovna, která má na starosti škálovatelný machine learning. Toto strojové učení využívá sadu funkcí a algoritmů, které Mahout obsahuje. Jsou naprogramované pomocí MapReduce paradigmatu. Zaměření Mahoutu je primárně vedeno na odvětví kolaborativního

filtrování, klasifikaci dat a clusterování. Mahout využívá principy algoritmů a matematických funkcí z oborů lineární statistiky a algebry.

Jeho využití spočívá hlavně v:

- Na základě uživatelského chování doporučuje funkce.
- Podle shody v obsahu rozřazuje dokumenty do clusterů.
- Klasifikuje dokumenty [21].

5.7 Flume

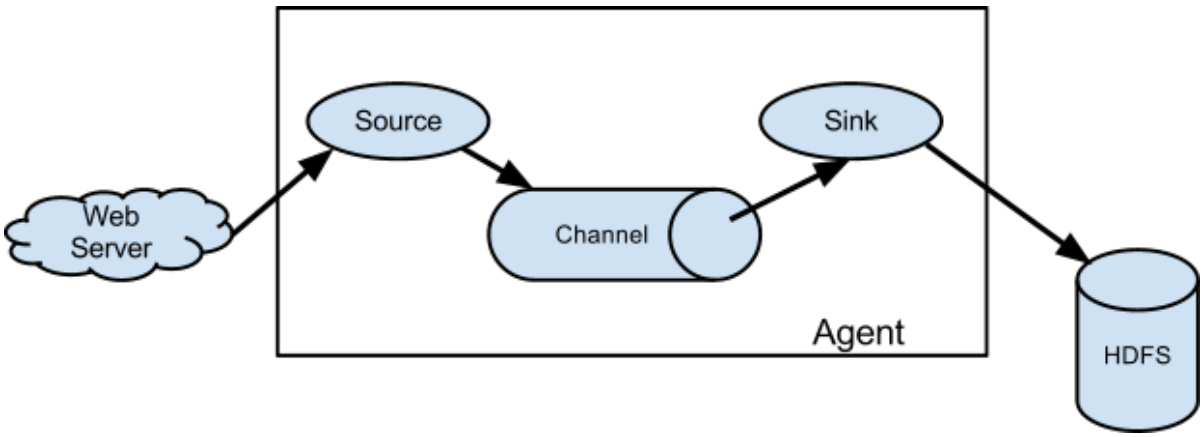
Jedná se o službu, která umožnuje sběr velkého množství dat. Zajišťuje jejich agregaci a přesun z různých zdrojů do centrálního úložiště. Flume je spolehlivý a odolný vůči chybám. Obsahuje mechanizmy, které umožňují převzetí služeb při selhání a obnovení. Flume se většinou používá k získání dat ze síťového provozu, sociálních médií či emailových zpráv.

Základní pojmy:

- Event: Jednotka dat, která je transportována z místa původu do cílového umístění.
- Flow: Sled událostí neboli tok dat.
- Client: Implementace rozhraní, která pracuje v bodě původu události a předává je agentovi Flume.
- Agent: Jedná se o nezávislý proces, který je složen z různých částí, jako jsou zdroje, kanály a tzv. sinks. Agent umožňuje ukládáni, přijímání a předávání události k dalšímu cíly.
- Source: Rozhraní, které je určeno k přijímání událostí prostřednictvím konkrétního mechanizmu.
- Channel: Přechodný systém pro ukládání událostí. Události jsou přenášeny do „channel“ pomocí „sources“, které fungují v „agent“.
- Sink: Toto rozhraní umožňuje odstranění událostí z kanálu, jejich předání dalšímu agentovi.
- Terminal sink: Nepředává událost dalšímu agentovi, ale již do konečného cíle.

5.7.1 Datový tok Flume

Datový tok začíná u klienta. Klient předá událost cíli, který působí jako zdroj v agentu. Po přijmutí události zdrojem, je přeposlána na jeden nebo více kanálů. Kanály po přijmutí události jsou zpracovány „sink“ a pokud se jedná a jednoduchý „sink“ bude událost předána dalšímu agentovi. Pokud se jedná o „terminal sink“, předá se událost do konečné destinace [23].



Obrázek 9

Datový tok – Flume (Zdroj: [23])

6. Instalace a konfigurace Apache Hadoop

6.1 Požadavky pro spuštění Hadoop

Apache Hadoop využívá operační systém typu Linux. Hadoop vyžaduje nainstalované rozhraní Java v poslední verzi. Aby se mohly spouštět všechny služby a konfigurovat, musí být vytvořena skupina a uživatel, který má k tomuto plná práva.

Pro správu stanic v clusteru se používá SSH přístup. Dále je nutno vytvořit RSA klíč s prázdným heslem. Poté se Hadoop stáhne, rozbalí a provedou se základní konfigurace změnou souborů „core-site.xml“, „mapred-site.xml“ a „hdfs-site.xml“. Nakonec už zbývá naformátovat souborový systém pomocí instance NameNode. Formátování se provádí jen před prvním spuštěním.

6.2 Režimy Apache Hadoop

Hadoop je možno spouštět a konfigurovat v několika režimech.

Jako první je standalone režim (samostatný režim). Tento režim je implicitně nastavený a nepodporuje interakci se souborovým systémem HDFS. Je využitelný pro vytváření MapReduce aplikací.

Jako další je pseudo – distribuovaný režim. Zde již je možnost pracovat s jednou stanicí, na které se provádí výpočetní funkce a zároveň slouží jako datové úložiště. S tímto režimem je možnost provádět všechny potřebné operace.

A v poslední řadě je plně distribuovaný režim. V tomto režimu je plně funkční konfigurace Hadoop clusteru, který je složen z master stanice s instancí NameNode a JobTracker, dále využívá stanici se Secondary NameNodem a podřízenou stanici s DataNode a TaskTracker [22].

6.3 Použitý hardware a software

Pro práci s Apache Hadoop jsem využil virtuální server se čtyř jádrovým procesorem a operační pamětí 16 GB RAM, který byl vytvořen na Přírodovědecká fakultě Jihočeské univerzity. Operační systém byl použit Debian.

6.4 Instalace Java Frameworku

Java je nedílnou součástí softwarové platformy Apache Hadoop, protože je v ní napsána. Je nutno mít nainstalované Java rozhraní minimálně ve verzi 1.8. Instalace jsem provedl pomocí následujících příkazů:

```
su –

echo "deb http://ppa.launchpad.net/webupd8team/java/ubuntu trusty main" | tee
    /etc/apt/sources.list.d/webupd8team-java.list

echo "deb-src http://ppa.launchpad.net/webupd8team/java/ubuntu trusty main" | tee -a
    /etc/apt/sources.list.d/webupd8team-java.list

apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys EEA14886
```

apt-get update

apt-get install oracle-java8-installer

exit

6.5 Instalace SSH serveru

Aby bylo možné spravovat jednotlivé stanice v clusteru, je nutná instalace SSH serveru. Instalace se provede následovně:

su –

apt-get install openssh-server

6.6 Vytvoření skupiny a uživatele Hadoop

Aby nedošlo ke kolizím mezi účty a jinými softwarovými aplikacemi, které běží na stejné stanici jako Hadoop, je nejlepším řešením vytvořit zvlášť vyhrazeného uživatele Hadoop. Nejdříve se vytvoří nová skupina hadoop. Dále se do této skupiny přidá nový uživatel hadoop. Je zde možné vyplnit detailnější informace o uživateli a také zabezpečit účet heslem. Vše se provede následujícími příkazy:

su –

addgroup hadoop

adduser –ingroup hadoop hadoop

6.7 SSH certifikát

Pro správné fungování clusteru v pseudo-distribuovaném režimu je potřeba nakonfigurovat SSH certifikát pro přihlášení bez hesla. Je nutné vytvořit SSH přístup pro uživatele hadoopuser k localhost.

Nejdříve se provede příkaz na přepnutí uživatele hadoop. Poté se spustí generování veřejného a privátního RSA klíče. Dále se musí potvrdit uložení klíče do složky

„/home/hadoop/.ssh/id_rsa“. Jsou zobrazeny cesty uložení veřejného a privátního klíče i s jeho alfanumerickou a obrázkovou podobou. Posledním krokem je vykonání příkazu, který přidá vygenerovaný klíč do seznamu autorizovaných klíčů na localhost a otestování SSH připojení.

```
su – hadoop
ssh-keygen –t rsa –P
cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys
ssh localhost
exit
```

6.8 Instalace Hadoop

Instalace Hadoop vyžaduje práva uživatele root. Po přepnutí na něj se zvolí složka, do které se Hadoop stáhne. Dále se provede jeho rozbalení a nastavení uživateli a skupině hadoop práva vlastníka pro veškeré soubory.

```
cd /usr/local
wget http://apache.miloslavbrada.cz/hadoop/common/hadoop-2.6.0/hadoop-2.6.0.tar.gz
sudo tar vxzf hadoop-2.6.0.tar.gz
sudo mv hadoop-2.6.0 hadoop
sudo chown -R hadoop:hadoop hadoop
```

6.9 Konfigurace Hadoop

Veškeré konfigurační změny se provádí pod uživatelem hadoopuser. Aby Hadoop mohl bezproblémově fungovat, musel jsem upravit několik souborů.

Jako první je soubor „*.bashrc“*, ve kterém se musí nastavit cesta k nainstalovanému Java Frameworku. Soubor se otevře pomocí příkazu „*nano .bashrc“.* Na konec souboru se vloží následující text.

#Hadoop variables

export JAVA_HOME=/usr/lib/jvm/java-8-oracle/ (cesta k nainstalovanému Java Frameworku)

export HADOOP_INSTALL=/usr/local/hadoop

export PATH=$PATH:$HADOOP_INSTALL/bin

export PATH=$PATH:$HADOOP_INSTALL/sbin

export HADOOP_HOME=$HADOOP_INSTALL

export HADOOP_MAPRED_HOME=$HADOOP_INSTALL

export HADOOP_COMMON_HOME=$HADOOP_INSTALL

export HADOOP_HDFS_HOME=$HADOOP_INSTALL

export HADOOP_YARN_HOME=$HADOOP_INSTALL

export HADOOP_CONF_DIR=$HADOOP_INSTALL/etc/hadoop

Poté se soubor uloží a zavře. Tento soubor se vykoná příkazem:

source ~/.bashrc

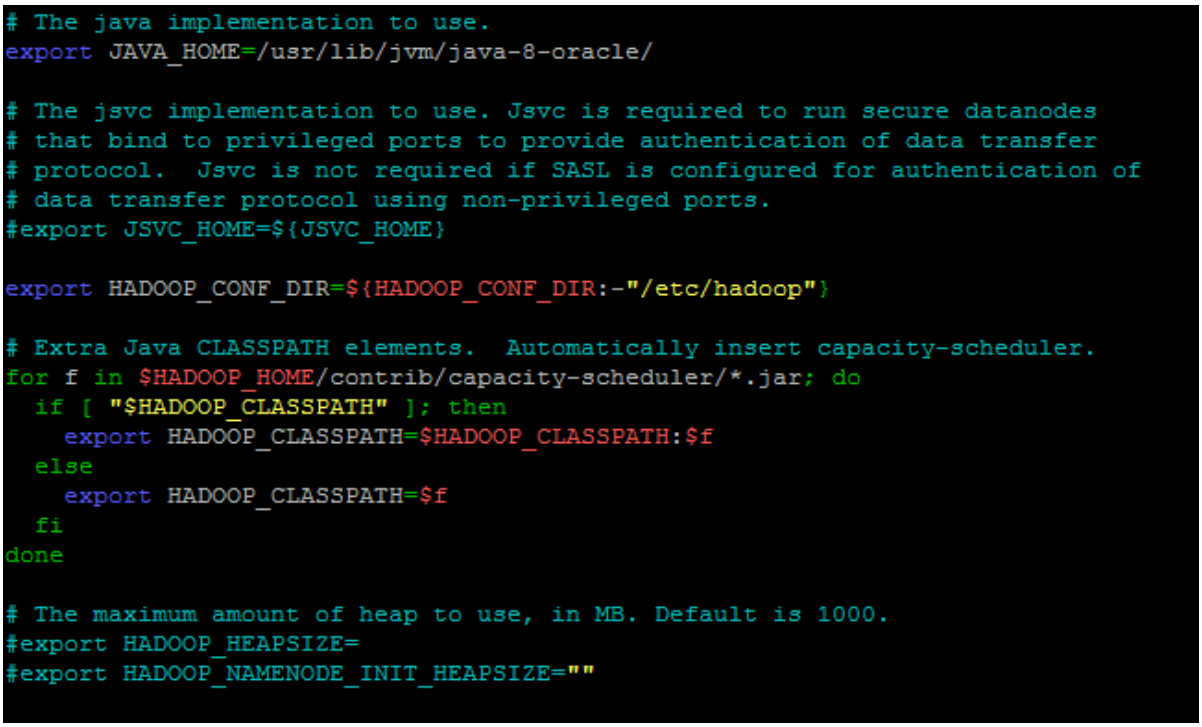
Jako další soubor bylo nutno upravit a nastavit cestu k Java Frameworku je „*hadoop-env.sh*“. Otevře se pomocí příkazu:

nano /usr/local/hadoop/etc/hadoop/hadoop-env.sh

V tomto souboru se nalezne řádek s proměnnou JAVA_HOME a upraví se:

export JAVA_HOME=/usr/lib/jvm/java-8-oracle/

Soubor se uloží a zavře.



```
# The java implementation to use.
export JAVA_HOME=/usr/lib/jvm/java-8-oracle/

# The jsvc implementation to use. Jsvc is required to run secure datanodes
# that bind to privileged ports to provide authentication of data transfer
# protocol.  Jsvc is not required if SASL is configured for authentication of
# data transfer protocol using non-privileged ports.
#export JSVC_HOME=${JSVC_HOME}

export HADOOP_CONF_DIR=${HADOOP_CONF_DIR:-"/etc/hadoop"}

# Extra Java CLASSPATH elements.  Automatically insert capacity-scheduler.
for f in $HADOOP_HOME/contrib/capacity-scheduler/*.jar; do
  if [ "$HADOOP_CLASSPATH" ]; then
    export HADOOP_CLASSPATH=$HADOOP_CLASSPATH:$f
  else
    export HADOOP_CLASSPATH=$f
  fi
done

# The maximum amount of heap to use, in MB. Default is 1000.
#export HADOOP_HEAPSIZE=
#export HADOOP_NAMENODE_INIT_HEAPSIZE=""
```

Obrázek 10

Soubor „.hadoop-env.sh“ (Zdroj: Autor)

Soubor *„core-site.xml“* obsahuje „properties“, které Hadoop využívá při startu. Slouží k přepsání defaultního nastavení startu Hadoopu.

Otevře se příkazem:

nano /usr/local/hadoop/etc/hadoop/core-site.xml

Mezi tagy <configuration></configuration> se vloží následující text:

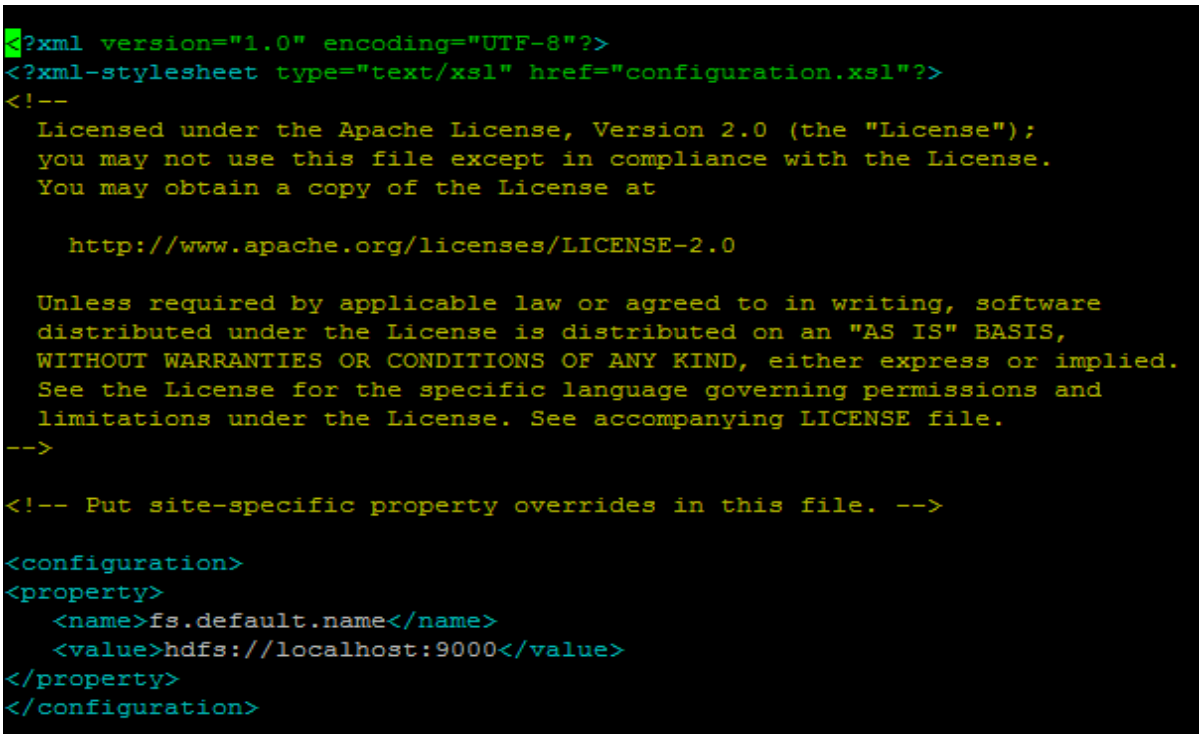
<property>

<name>fs.default.name</name>

<value>hdfs://localhost:9000</value>

</property>

Soubor se uloží a zavře.



Obrázek 11

Soubor „core-site.xml (Zdroj: Autor)

Dalším z řady konfiguračních souborů je *„yarn-site.xml“*. Tento soubor se také používá k přepsání defaultního nastavení startu Hadoopu.

Otevření souboru:

nano /usr/local/hadoop/etc/hadoop/yarn-site.xml

Poté opět mezi tagy <configuration></configuration> vložíme následující text:

<property>

<name>yarn.nodemanager.aux-services</name>

<value>mapreduce_shuffle</value>

</property>

<property>

<name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>

<value>org.apache.hadoop.mapred.ShuffleHandler</value>

</property>

Soubor se uloží a zavře.

```
<?xml version="1.0"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->
<configuration>
<property>
   <name>yarn.nodemanager.aux-services</name>
   <value>mapreduce_shuffle</value>
</property>
<property>
   <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
<value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
</configuration>
```

Obrázek 12

Soubor „yarn-site.xml“ (Zdroj: Autor)

V adresáři Hadoopu se nachází soubor s názvem *„mapred-site.xml.template“,* který se musí přejmenovat a upravit. Tento soubor definuje Framework, který bude použit pro MapReduce.

Soubor se nejdříve přejmenuje na *„mapred-site.xml“*. To se provede příkazem:

cp /usr/local/hadoop/etc/hadoop/mapred-site.xml.template
/usr/local/hadoop/etc/hadoop/mapred-site.xml

Dále se soubor otevře:

nano /usr/local/hadoop/etc/hadoop/mapred-site.xml

Opět se mezi tady <configuration></configuration> vloží následující text:

<property>

<name>mapreduce.framework.name</name>

<value>yarn</value>

</property>

Soubor se uloží a zavře.

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
<property>
   <name>mapreduce.framework.name</name>
   <value>yarn</value>
</property>
</configuration>
```

Obrázek 13

Soubor „mapred-site.xml“ (Zdroj: Autor)

Posledním souborem, který bylo nutno upravit byl soubor *„hdfs-site.xml“*. Tento soubor musí být nakonfigurován pro každého uživatele v clusteru. Jsou v něm specifikovány adresáře, které slouží jako NameNode a DataNode na konkrétním uživatelském počítači.

Než se začne tento soubor konfigurovat, je nutno vytvořit následující složky:

```
mkdir -p /home/hadoop/mydata/hdfs/namenode
mkdir -p /home/hadoop/mydata/hdfs/datanode
```

Po vytvoření složek se otevře soubor *„hdfs-site.xml“*.

```
nano /usr/local/hadoop/etc/hadoop/hdfs-site.xml
```

A mezi tagy <configuration></configuration> je vložen následující text, obsahující definovanou cestu k výše vytvořeným složkám.

```
<property>
  <name>dfs.replication</name>
  <value>1</value>
</property>
<property>
  <name>dfs.namenode.name.dir</name>
  <value>file:/ home/hadoopuser/mydata/hdfs/namenode </value>
</property>
<property>
  <name>dfs.datanode.data.dir</name>
  <value>file:/home/hadoopuser/mydata/hdfs/datanode </value>
</property>
```

Soubor se uloží a zavře.

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
<property>
   <name>dfs.replication</name>
   <value>1</value>
 </property>
 <property>
   <name>dfs.namenode.name.dir</name>
   <value>file:/home/hduser/mydata/hdfs/namenode </value>
 </property>
 <property>
   <name>dfs.datanode.data.dir</name>
   <value>file:/home/hduser/mydata/hdfs/datanode </value>
 </property>
</configuration>
```

Obrázek 14

Soubor „hdfs-site.xml“ (Zdroj: Autor)

6.10 Spuštění Hadoop

Pokud se všechny konfigurační změny provedly správně, je potřeba před prvním spuštěním naformátovat souborový systém HDFS. Tento krok smaže veškerá data z Hadoopu a je potřeba ho provést jen před prvním spuštěním. Formátování se provede následujícím příkazem.

hdfs namenode -format

Po tomto kroku je vše připraveno ke spuštění Hadoopu. Nejdříve se musí spustit NameNode a DataNode.

Provede se příkazem:

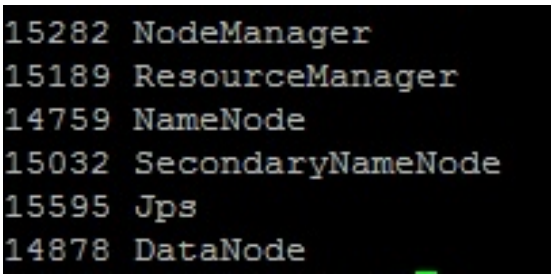
start-dfs.sh

Dále je potřeba spustit „démony“ a Resource Manager příkazem:

start-yarn.sh

Hadoop je nyní spuštěn. To je možno ověřit příkazem, který zobrazí běžící procesy:

jps



Obrázek 15

Hadoop – běžící procesy (Zdroj: Autor)

7. Otestování funkčnosti úlohou WordCount

Pro otestování funkčnosti systému Apache Hadoop byla zvolena ukázková úloha „WordCount“. Jedná se o úlohu, která zjišťuje počet výskytů slov v textovém dokumentu.

Nejdříve vytvoříme složku, ve které se bude pracovat a kompilovat. Vytvoříme ji příkazem:

mkdir /home/hadoop/priklady

Zdrojový kód napíšeme do souboru WordCount.java. Dále vytvoříme složku „wordcount_classes“, ve které se nachází cesta k „Main“ třídě v souboru „.jar“. Složku vytvoříme následujícím příkladem:

mkdir /home/hadoop/priklady/wordcount_classes

Poté je nutné vytvořit vstupní textový soubor, který bude WordCount zpracovávat. Dalším krokem je kompilace zdrojového kódu WordCount.java.

javac -classpath
/usr/local/hadoop/etc/hadoop:/usr/local/hadoop/share/hadoop/common/lib/:/usr/local/hadoop/share/hadoop/common/*:/usr/local/hadoop/share/hadoop/hdfs:/usr/local/hadoop/share/hadoop/hdfs/lib/*:/usr/local/hadoop/share/hadoop/hdfs/*:/usr/local/hadoop/share/hadoop/yarn/lib/*:/usr/local/hadoop/share/hadoop/yarn/*:/usr/local/hadoop/share/hadoop/mapreduce/lib/*:/usr/local/hadoop/share/hadoop/mapreduce/*:/usr/local/hadoop/contrib/capacity-scheduler/*.jar -d wordcount_classes WordCount.java*

Spustitelný soubor WordCount.jar se vytvoří následovně:

jar cvf /home/hadoop/priklady/WordCount.jar –C wordcount_classes/ .

Nyní se vytvoří složka input v HDFS, do které se zkopíruje textový soubor, který se bude zpracovávat. Složka output se vytvoří automaticky, při běhu programu.

Vytvoření input složky:

hadoop fs –mkdir –p /user/wordcount/input

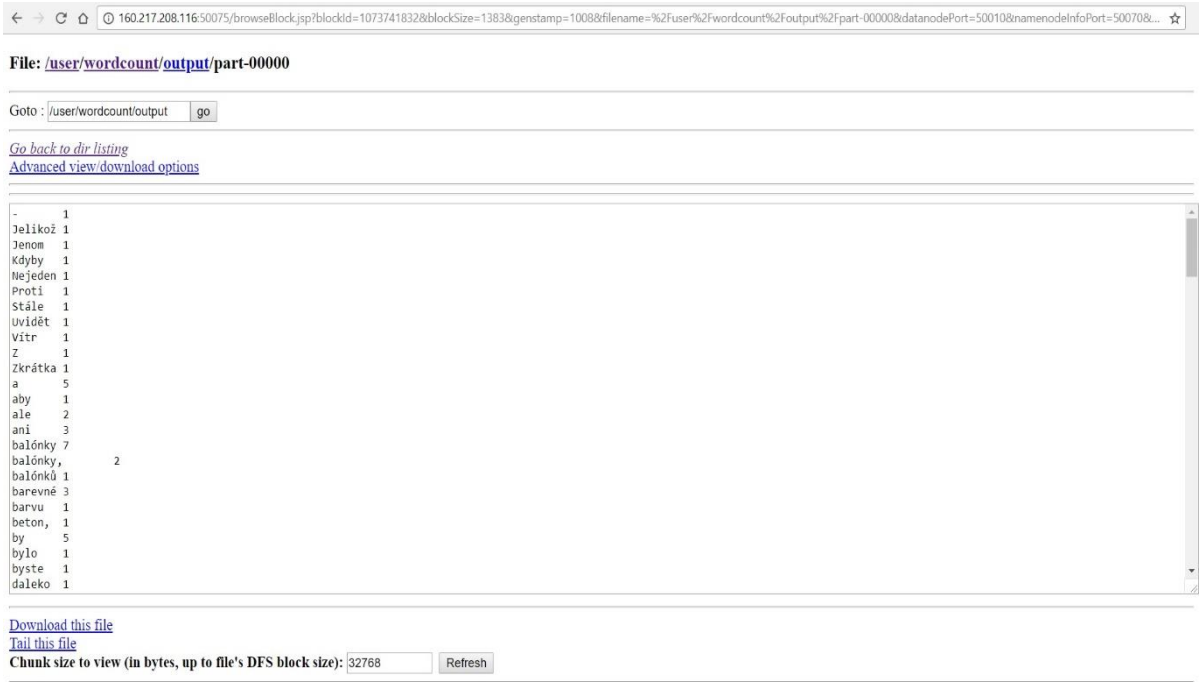
Zkopírování vstupního souboru do input složky:

hadoop fs –put /home/hadoop/priklady/text.txt /user/wordcount/input

Jako konečný krok se provede spuštění úlohy WordCount:

hadoop jar /home/hadoop/priklady/WordCount.jar org.myorg.WordCount
/user/wordcount/input /user/wordcount/output

Výsledek je možno zkontrolovat přes webové rozhraní. Ve složce output se nachází soubor „part-00000“. Po jeho otevření se zobrazí požadované výsledky.



Obrázek 16

WordCount – output (Zdroj: Autor)

8. Experimenty

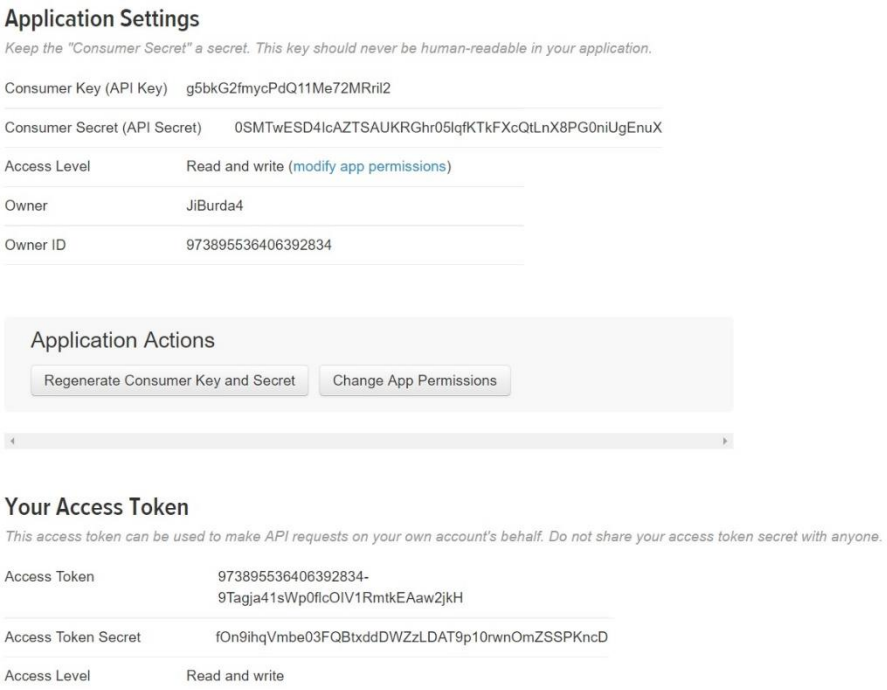
8.1 Založení účtu a Twitter API

Pro experimenty jsem využil možnosti získání dat z Twitteru. Nejdříve jsem si založil klasický uživatelský účet na Twitteru. Poté se na stránkách https://apps.twitter.com/ vytvoří nová Twitter app.

Obrázek 17

Vytvoření Twitter app (Zdroj: Autor)

Vyplní se požadované údaje a potvrdí vytvoření. Dále se v záložce „Keys and Access Tokens“ vytvoří přístupové klíče.



Obrázek 18

Twitter keys (Zdroj: Autor)

8.2 Instalace a konfigurace Apache Flume

Apachu Flume jsem poté využil pro získání dat z Twitteru. Příkaz pro stažení Apache Flume:

wget https://www.apache.org/dist/flume/stable/apach e-flume-1.6.0-bin.tar.gz

Dále se tento soubor rozbalí:

tar -xvzf apache-flume-1.6.0-bin.tar.gz

Jako další krok se provede stažení .jar souboru. Tento souboru umožňuje připojit se ke službě Twitter Streaming API a přijímat tweety v JSON formátu, které se následně ukládají do HDFS

wget http://files.cloudera.com/samples/flume-source s-1.0-SNAPSHOT.jar

Soubor se zkopíruje do složky „lib“ v domovském adresáři apache-flume.

cp flume-sources-1.0-SNAPSHOT.jar $FLUME_HOME/lib

Nyní se musí zkopírovat a přejmenovat soubor flume-env.sh.template na soubor flume-env.sh, kde na konci souboru vložíme konfigurační řádky na JAVA_HOME a FLUME_CLASSPATH

export JAVA_HOME=/usr/lib/jvm/java-8-oracle/
FLUME_CLASSPATH="/usr/local/apache-flume-1.8.0-bin/lib/flume-sources-1.0-SNAPSHOT.jar"

Jako další z konfiguračních souboru, který se musí upravit je soubor „*.bashrc“*, do něhož vložíme následující řádky:

export FLUME_HOME=/home/hadoop/work/apache-flume-1.6.0-bin
export FLUME_CONF_DIR=$FLUME_HOME/conf
export FLUME_CLASS_PATH=$FLUME_CONF_DIR
export PATH=$FLUME_HOME/bin:$PATH

Nyní přichází na řadu soubor „flume-twitter.conf“, do kterého se vloží výše vygenerované klíče z Twitter app. a zadají se keywords, podle kterých se budou požadované tweety vyhledávat.

```
TwitterAgent.sources = Twitter
TwitterAgent.channels = MemChannel
TwitterAgent.sinks = HDFS

TwitterAgent.sources.Twitter.type = com.cloudera.flume.source.TwitterSource
TwitterAgent.sources.Twitter.channels = MemChannel

TwitterAgent.sources.Twitter.consumerKey = g5bkG2fmycPdQ11Me72MRril2
TwitterAgent.sources.Twitter.consumerSecret = 0SMTwESD4IcAZTSAUKRGhr05lqfKTkFXcQtLnX8PG0niUgEnuX
TwitterAgent.sources.Twitter.accessToken = 973895536406392834-9Tagja41sWp0flcOIV1RmtkEAaw2jkH
TwitterAgent.sources.Twitter.accessTokenSecret = fOn9ihqVmbe03FQBtxddDWZzLDAT9p10rwnOmZSSPKncD

TwitterAgent.sources.Twitter.keywords = hadoop, data

TwitterAgent.sinks.HDFS.channel = MemChannel
TwitterAgent.sinks.HDFS.type = hdfs
TwitterAgent.sinks.HDFS.hdfs.path = hdfs://localhost:9000/user/twitter/
TwitterAgent.sinks.HDFS.hdfs.fileType = DataStream
TwitterAgent.sinks.HDFS.hdfs.writeFormat = Text
TwitterAgent.sinks.HDFS.hdfs.batchSize = 1000
TwitterAgent.sinks.HDFS.hdfs.rollSize = 0
TwitterAgent.sinks.HDFS.hdfs.rollCount = 10000

TwitterAgent.channels.MemChannel.type = memory
TwitterAgent.channels.MemChannel.capacity = 10000
TwitterAgent.channels.MemChannel.transactionCapacity = 100
```

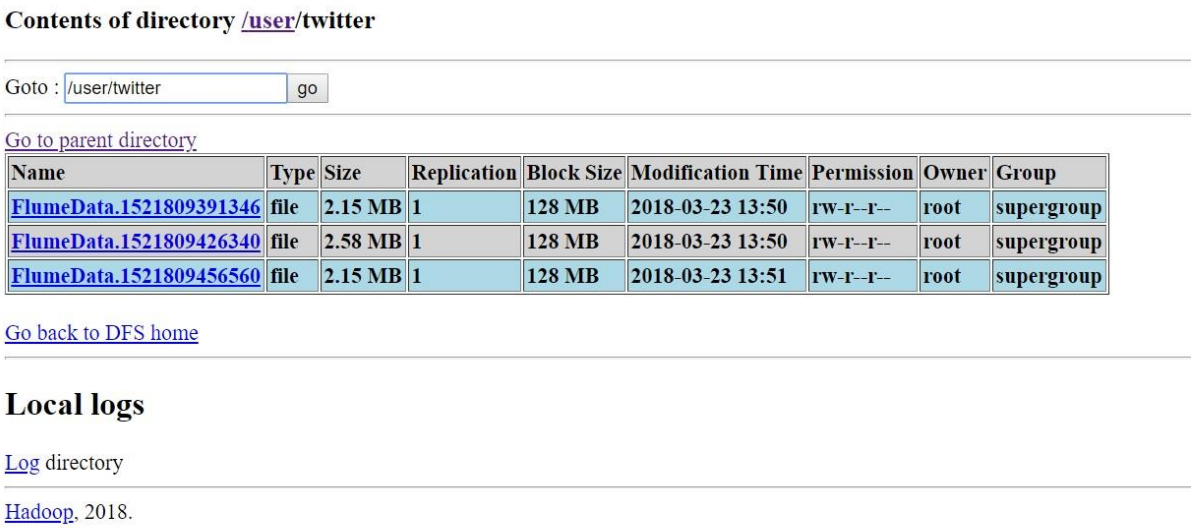
Obrázek 19

Konfigurace souboru flume-twitter.conf (Zdroj: Autor)

Posledním krokem je spuštění flume agenta:

bin/flume-ng agent -n TwitterAgent --conf ./conf/ -f conf/flume-twitter.conf -Dflume.root.logger=DEBUG,console

Výstup v JSON formátu je možno zkontrolovat přes webové rozhraní:



Contents of directory /user/twitter

Goto : /user/twitter go

Go to parent directory

Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
FlumeData.1521809391346	file	2.15 MB	1	128 MB	2018-03-23 13:50	rw-r--r--	root	supergroup
FlumeData.1521809426340	file	2.58 MB	1	128 MB	2018-03-23 13:50	rw-r--r--	root	supergroup
FlumeData.1521809456560	file	2.15 MB	1	128 MB	2018-03-23 13:51	rw-r--r--	root	supergroup

Go back to DFS home

Local logs

Log directory

Hadoop, 2018.

Obrázek 20

Twitter - output (Zdroj: Autor)

8.3 Instalace a konfigurace Apache Hive

Po úspěšném získání dat z Twitteru jsem využil Apache Hive pro jejich zpracování. Nejprve se Hive stáhne:

wget http://www.gtlib.gatech.edu/pub/apache/hive/stable/hive-1.2.2-bin.tar.gz

Dále se provede rozbalení:

tar -xzvf hive-1.2.2-bin.tar.gz

Následujícím krokem bylo nakonfigurování souboru „bashrc“. Důvodem bylo nastavení „HIVE_HOME“. Soubor se otevře:

nano ~/.bashrc

Do souboru byly přidány následující řádky:

export HIVE_HOME=/Downloads/apache-hive-1.2.2-bin

export PATH=$HIVE_HOME/bin:$PATH

Konfigurační změny se provedou příkazem:

source ~/.bashrc

Následně je nutné upravit soubor „hive-config.sh“, ve kterém se nastaví cesta k Hadoopu.

export HADOOP_HOME=/usr/local/hadoop/hadoop-2.6.5

Jelikož Hive defaultně očekává, že části dat budou od sebe řádně odděleny, je potřeba stáhnout „hive-serdes-1.0-SNAPSHOT.jar“ (Serde – serializer a deserializer), který umožní převést JSON formát do srozumitelné podoby pro Hive. Tento .jar soubor se nachází v balíku, který se stáhne pomocí příkazu:

wget https://github.com/cloudera/cdh-twitter-example.git

Soubor „hive-serdes-1.0-SNAPSHOT.jar“ se poté musí zkopírovat do složky „lib“ v Hive.

cp /Downloads/cdh-twitter-example/hive-serdes/target/hive-serdes-1.0-SNAPSHOT.jar /Downloads/apache-hive-1.2.2-bin/lib

Problémy při spouštění Hive:

Jedním z problémů, které jsem musel řešit při práci s Hive, bylo jeho spuštění. Při využití souborů .jar, které nejsou v mapperu a reduceru, není možné načíst danou třídu. Řešením je přidáním příkazu před spuštěním Hive:

export HADOOP_USER_CLASSPATH_FIRST=true

Hive se spustí jednoduchým příkazem:

Hive

8.4 Vytvoření tabulek v Hive

Na další problém jsem narazil při vytváření tabulek. Týkalo se to rezervovaných „keywords“, které jsem nemohl použít ve struktuře daných tabulek. Řešením jsem nalezl v tomto příkazu:

```
set hive.support.sql11.reserved.keywords=false;
```

Následujícím krokem bylo přidání .jar souboru „hive-serdes-1.0-SNAPSHOT.jar;“. Tento příkaz píšeme při spuštěném Hive.

```
add jar /Downloads/apache-hive-1.2.2-bin/lib/hive-serdes-1.0-SNAPSHOT.jar;
```

Jako první jsem vytvořil tabulku „dictionary“ a „time_zone“. Tabulka „dictionary“ obsahuje anglická slova a jejich hodnocení (sentiment). Tabulka „time_zone_map“ obsahuje časovou zónu uživatelovo země. Dále bylo nutno vytvořit několik externích tabulek a „views“, které umožňují přístup a transformaci nezpracovaných dat.

Vytvoření tabulky „dictionary“:

```
CREATE EXTERNAL TABLE dictionary (
 type string,
 length int,
 word string,
 pos string,
 stemmed string,
 polarity string
)
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'
STORED AS TEXTFILE
LOCATION '/user/twitter';
```

Vytvoření tabulky „time_zone“:

```
CREATE EXTERNAL TABLE time_zone (
 time_zone string,
 country string,
 notes string
)
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'
STORED AS TEXTFILE
LOCATION '/user/twitter';
```

Vytvoření externí tabulky „tweets_raw“:

```
CREATE EXTERNAL TABLE tweets_raw (
 id BIGINT,
 created_at STRING,
 source STRING,
 favorited BOOLEAN,
 retweet_count:INT,
 retweeted_status STRUCT<
 text:STRING,
 user:STRUCT<screen_name:STRING,name:STRING>>,
 entities STRUCT<
 urls:ARRAY<STRUCT<expanded_url:STRING>>,
 user_mentions:ARRAY<STRUCT<screen_name:STRING,name:STRING>>,
 hashtags:ARRAY<STRUCT<text:STRING>>>,
 user STRUCT<
 screen_name:STRING,
 name:STRING,
 friends_count:INT,
 followers_count:INT,
 statuses_count:INT,
 verified:BOOLEAN,
```

```
 utc_offset:INT,
 time_zone:STRING>,
 in_reply_to_screen_name STRING
 year int,
 month int,
 day int,
 hour int
)
ROW FORMAT SERDE 'com.cloudera.hive.serde.JSONSerDe'
LOCATION '/user/twitter';
```

Vytvoření „views“:

```
CREATE VIEW tweets_simple AS
SELECT
 id,
 cast ( from_unixtime( unix_timestamp(concat( '2014 ', substring(created_at,5,15)), 'yyyy
MMM
dd hh:mm:ss')) as timestamp) ts,
 substring(cast(cast( from_unixtime( unix_timestamp(concat( '2014 ',
substring(created_at,5,15)), 'yyyy MMM dd hh:mm:ss')) as timestamp) as string),1,10)
created_at_string,
 text,
 user.time_zone,
 source source_raw ,
 CASE WHEN array_contains(split(source,' '),'iPhone</a>') = true THEN 'iOS'
 WHEN array_contains(split(source,' '),'iPad</a>') = true THEN 'iOS'
 WHEN array_contains(split(source,' '),'iOS</a>') = true THEN 'iOS'
 WHEN source = '<a href="http://www.apple.com" rel="nofollow">iOS</a>' THEN 'iOS'
 WHEN array_contains(split(source,' '),'Windows') = true AND
array_contains(split(source,' '),'Phone</a>') = true THEN 'Windows Phone'
 WHEN array_contains(split(source,' '),'Windows</a>') = true THEN 'Windows'
 WHEN array_contains(split(source,' '),'Android</a>') = true THEN 'Android'
```

```
 WHEN array_contains(split(source,' '),'Android') = true THEN 'Android'
 WHEN array_contains(split(source,' '),'Android') = true AND
array_contains(split(source,' '),'Tablets</a>') = true THEN 'Android'
 WHEN array_contains(split(source,' '),'Mac</a>') = true THEN 'Mac'
 WHEN source = '<a href="https://dev.twitter.com/docs/tfw" rel="nofollow">Twitter for
Websites</a>' THEN 'WEB'
 WHEN source = '<a href="http://twitter.com" rel="nofollow">Twitter Web Client</a>'
THEN
'WEB'
 WHEN source = '<a href="https://about.twitter.com/products/tweetdeck"
rel="nofollow">TweetDeck</a>' THEN 'TweetDeck'
 WHEN source = '<a href="http://dlvr.it" rel="nofollow">dlvr.it</a>' THEN 'N/A'
 WHEN array_contains(split(source,' '),'BlackBerry®</a>') = true THEN 'BlackBerry'
 WHEN array_contains(split(source,' '),'BlackBerry</a>') = true THEN 'BlackBerry'
 ELSE 'UNKNOWN' END device
FROM tweets_raw;


CREATE VIEW tweets_clean AS
SELECT
 id,
ts,
 created_at_string,
 source_raw,
 device,
 text,
 m.country
FROM tweets_simple t LEFT OUTER JOIN time_zone_map m ON t.time_zone = m.time_zone;


create view l1 as select id, words from tweets_raw lateral view
explode(sentences(lower(text))) dummy as words;
```

```
create view l2 as select id, word from l1 lateral view explode( words ) dummy as word ;


create view l3 as select
 id,
 l2.word,
 case d.polarity
 when 'negative' then -1
 when 'positive' then 1
 else 0 end as polarity
from l2 left outer join dictionary d on l2.word = d.word;


create table tweets_sentiment stored as orc as select
 id,
 case
 when sum( polarity ) > 0 then 'positive'
 when sum( polarity ) < 0 then 'negative'
 else 'neutral' end as sentiment
from l3 group by id;
```

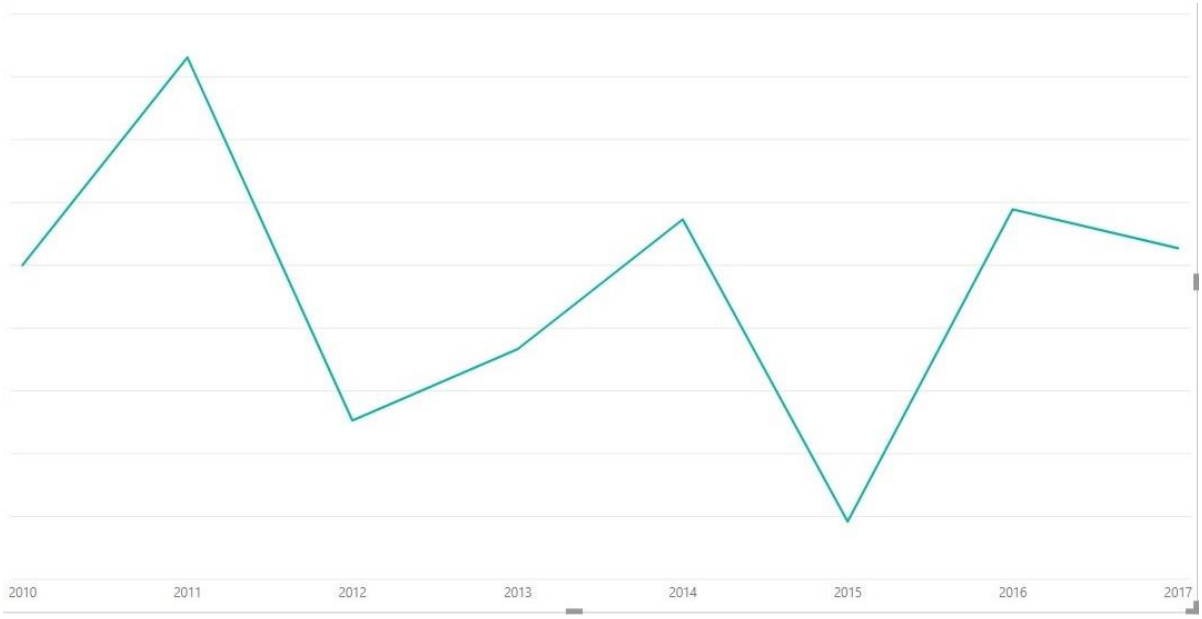
Vytvoření tabulky „tweets“, která složí vše dohromady a přečísluje sentiment:

```
CREATE TABLE tweets
STORED AS ORC
AS
SELECT
 t.*,
 case s.sentiment
 when 'positive' then 2
 when 'neutral' then 1
 when 'negative' then 0
```

```
 end as sentiment
FROM tweets_clean t LEFT OUTER JOIN tweets_sentiment s on t.id = s.id;
```

9. Analýza dat z Twitteru

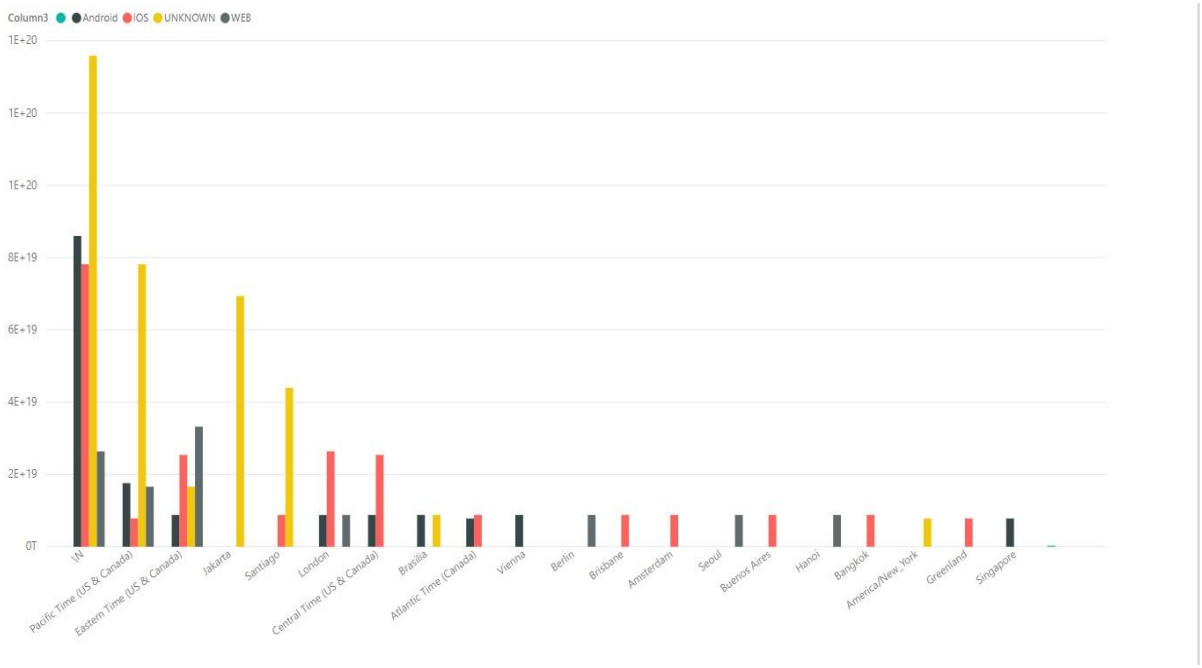
Ze získaných a zpracovaných dat z Twitteru jsem pomocí Power BI vytvořil několik grafů. Obrázek číslo 21 zobrazuje aktivitu uživatelů v období od 1.1.2010 do 31.12.2017.



Obrázek 21

Tweet activity (Zdroj: Autor)

Další graf zobrazuje používané operační systémy uživatelů v závislosti na časové zóně, ve které se nachází.



Obrázek 22

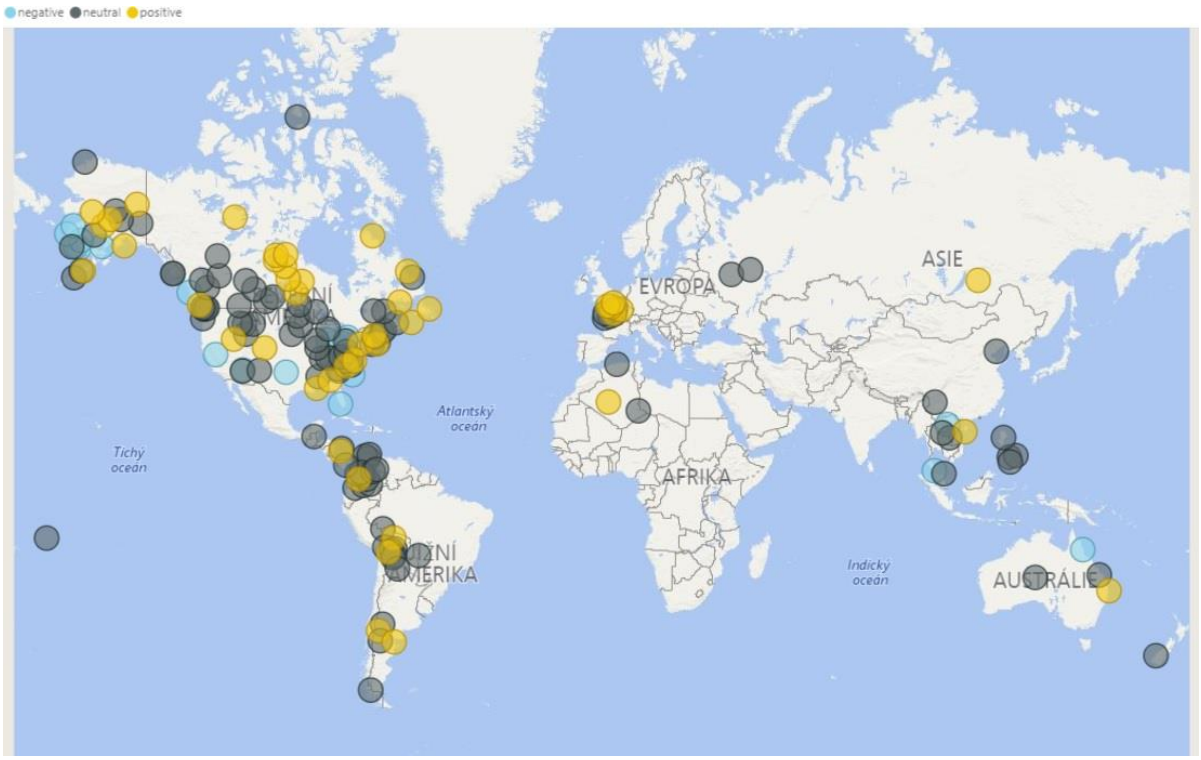
Operační systém / časové pásmo (Zdroj: Autor)

Následující graf je vytvořen na základě zeměpisné šířky a délky jednotlivých tweetů.

Obrázek 23

Tweets GEO mapping (Zdroj: Autor)

Sentiment analýza tweetů v hodnotách negative, neutral a positive.



Obrázek 24

Sentiment analýza (Zdroj: Autor)

Poslední graf znázorňuje sentiment analýzu na časové ose.

Obrázek 25

Sentiment analýza v čase (Zdroj: Autor)

10. Závěr

Jedním z hlavních cílů, které jsem si stanovil v této práci, bylo seznámení a vysvětlení fungování platformy Hadoop. V teoretické části jsem krátce popsal historii, vznik a také výhody a nevýhody. Dále jsem podrobněji vysvětlil Hadoop – HDFS a jeho práci s daty. Nesmělo také chybět nastínění Map a Reduce. Konec teoretické části jsem věnoval několika dalším nástrojům, které Hadoop využívá.

V praktické části jsem se věnoval především detailnímu popsání instalace a konfigurace Hadoop na vybraném serveru. Tato část je popsána krok za krokem s popisem a ukázkami konfiguračních souborů. Jako další krok jsem otestoval funkčnost celého systému na vzorové úloze „WordCount“. Po úspěšném vyzkoušení systému bylo možné se pustit do analýzy dat z Twitteru. Jako první krok jsem založil Twitter API, nainstaloval a nakonfiguroval nástroje Apache Flume a Hive, díky kterým jsem mohl získat data a zpracovat je. Jako výstup tohoto snažení bylo vytvořeno několik grafů pomocí softwaru Power BI.

Mezi největší problémy, které jsem musel řešit, bylo vyhledávání chybových hlášení, na které jsem během instalace a konfigurace narazil. Většina řešení na různých diskuzních fórech a stránkách, zabývajících se danou problematikou, nebyla správná a nalézt korektní příkazy bylo metodou pokus, omyl. To zabralo nejvíce času, a proto je tato práce sepsána detailně krok po kroku, aby vytvořila funkční celek.

Na závěr této práce bych chtěl poděkovat panu doc. Ing. Ladislavu Beránkovi, CSc. MBA. a panu Martinu Fialovi, za cenné rady a vedení při tvorbě této práce.

11. Seznam použité literatury

[1] Když se řekne "Hadoop". *Www.linuxexpress.cz* [online]. Autor časopis IT Systems, 2013 [cit. 2016-10-25]. Dostupné z: https://www.linuxexpres.cz/software/kdyz-se-rekne-hadoop

[2] *ŠKÁLOVATELNÉ PREDZPRACOVÁNÍ DAT PROSTREDNICTVÍM NÁSTROJE HADOOP*. Brno, 2012. Bakalářská práce.

[3] Big data a možnosti jejich využití. *Www.adastra.cz* [online]. Jakub Augustín, 2014 [cit. 2016-10-30]. Dostupné z: http://www.adastra.cz/clanky/big-data-a-moznosti-jejich-vyuziti

[4] Big data: Nové způsoby zpracování a analýzy velkých objemů dat. *Www.gcsystem.cz* [online]. GC System [cit. 2016-10-30]. Dostupné z: https://www.gcsystem.cz/cz/big-data/

[5] WHITE, T. *Hadoop, The Definitive Guide*. 1. vydání. Yahoo PRESS, 2009. 528 s.

ISBN: 978-0-596-52197-4.

[6] *Hadoop: The Definitive Guide, 2nd Edition*. 2. O'Reilly Media / Yahoo Press, 2010. ISBN 978-1-4493-8973-4.

[7] Hadoop. *Apache Hadoop* [online]. The Apache Software Foundation, 2008 [cit. 2016-10-31]. Dostupné z: http://hadoop.apache.org/docs/r1.2.1/hdfs_design.html

[8] Cod - E - mphasis. *Cod - E - mphasis* [online]. 2013 [cit. 2016-11-01]. Dostupné z: https://codemphasis.wordpress.com/

[9] *Hadoop: HDFS, MapReduce a výpočty v IBM BigInsights*. Praha, 2014. Bakalářská. Vysoká škola ekonomická. Vedoucí práce Ing. Miroslav Řezáč.

[10] Secondary Namenode - What it really do? *Madhukar's Blog* [online]. 2013 [cit. 2016-11-01]. Dostupné z: http://blog.madhukaraphatak.com/secondary-namenode---what-it-really-do/

[11] Parallel LZO: Splittable Compression for Apache Hadoop. *Cloudera Engineering blog* [online]. 2009 [cit. 2016-11-03]. Dostupné z: http://blog.cloudera.com/blog/2009/06/parallel-lzo-splittable-compression-for-hadoop/

[12] Správa souborů s Hadoop File System Commands. *DumHows* [online]. DumHows [cit. 2016-11-04]. Dostupné z: http://dumhows.com/cs/pages/1467965

[13] Hadoop - MapReduce. *Tutorialspoint* [online]. [cit. 2016-11-06]. Dostupné z: https://www.tutorialspoint.com/hadoop/hadoop_mapreduce.htm

[14] Paradigma MapReduce a Apache Hadoop. *Fakulta informačních technologií* [online]. 2015 [cit. 2016-11-06]. Dostupné z: http://www.fit.vutbr.cz/~rychly/public/docs/PDI.hadoop/PDI.hadoop.print.pdf

[15] Pavlík Martin. *Big Data Hadoop* [online]. Praha, 2014 [cit. 2016-11-06]. Dostupné z: https://docs.google.com/presentation/d/1nkxYEaFEasXDswt-TNrZd1Adt-9ITFO9dBumAHQvEmM/edit#slide=id.p9

[16] APACHE HIVE TM. *APACHE HIVE* [online]. 2014 [cit. 2016-11-14]. Dostupné z: http://hive.apache.org/

[17] Welcome to Apache Pig!. *The Apache Software Foundation* [online]. 2012 [cit. 2016-11-14]. Dostupné z: https://pig.apache.org/#Getting+Started

[18] Apaceh ZooKeeper. *The Apache Software Foundation* [online]. 2016 [cit. 2016-11-15]. Dostupné z: https://zookeeper.apache.org/

[19] *Nové trendy v Business Intelligence - Zaměření na Big Data a Hadoop*. Praha, 2014. Bakalářská. Vysoká škola ekonomická. Vedoucí práce Ing. Martin Vacek.

[20] Cassandra: Databáze pro skutečně velké projekty. *Linuxexpres* [online]. 2013 [cit. 2016-11-15]. Dostupné z: https://www.linuxexpres.cz/software/cassandra-databaze-pro-skutecne-velke-projekty

[21] Ch. Eaton, D. Deroos, T. Deutsch, G. Lapis, P. Zikopoulos. *Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data*. The McGraw-Hill Companies, 2012. ISBN 978-0-07-179053-6.

[22] *Detecting Document Similarity in Large Document Collection using MapReduce and the Hadoop Framework*. 2012. Brac University. Vedoucí práce Dr. Mumit Khan.

[23] *Apache Flume™: Flume 1.8.0 User Guide. Https://flume.apache.org [online]. Apache Software Foundation, 2017, 2009–2017 [cit. 2018-04-01]. Dostupné z: https://flume.apache.org/FlumeUserGuide.html*

12. Seznam obrázků