TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Zpracování velkých dat logistiky v automotive

Diplomová práce

Studijní program: N2612 – Elektrotechnika a informatika
Studijní obor: 1802T007 – Informační technologie

Autor práce: **Bc. Lukáš Vosecký**
Vedoucí práce: Mgr. Jiří Vraný, Ph.D.
Konzultant: Ing. Jakub Vancl (Škoda Auto a.s.)

Liberec 2020

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Zadání diplomové práce

Zpracování velkých dat logistiky v automotive

Jméno a příjmení: **Bc. Lukáš Vosecký**
Osobní číslo: M18000156
Studijní program: N2612 Elektrotechnika a informatika
Studijní obor: Informační technologie
Zadávající katedra: Ústav nových technologií a aplikované informatiky
Akademický rok: **2019/2020**

Zásady pro vypracování:

1. Seznamte se s problematikou velkých dat a platformami pro jejich zpracování jako Splunk, Cloudera Hadoop a Power BI.
2. Navrhněte řešení pro transformaci, manipulaci a přenos dat ze softwarové platformy Splunk do Cloudera Hadoop. Zaměřte se na univerzálnost a přenositelnost zpracování datového toku.
3. Implementujte prototyp navrženého řešení pomocí jazyka Python.
4. Vytvořte reporty v Power BI pro otestování funkčnosti celého datového toku a pro základní ukázkovou vizualizaci.

Rozsah grafických prací:	dle potřeby
Rozsah pracovní zprávy:	40-50 stran
Forma zpracování práce:	tištěná/elektronická
Jazyk práce:	Čeština

Seznam odborné literatury:

[1] POWELL, Brett. Mastering Microsoft Power BI: Expert techniques for effective data analytics and business intelligence. 1. Birmingham, UK: Packt Publishing, 2018. ISBN 978-1788297233.
[2] KELLEHER, John D. a Brendan TIERNEY. Data science. Cambridge, Massachusetts: The MIT Press, [2018]. ISBN 978-0262535434.
[3] VANDERPLAS, Jacob T. Python data science handbook: essential tools for working with data. 2016. ISBN 978-1491912058.

Vedoucí práce:	Mgr. Jiří Vraný, Ph.D. Ústav nových technologií a aplikované informatiky
Konzultant práce:	Ing. Jakub Vancl Škoda Auto a.s.
Datum zadání práce:	9. října 2019
Předpokládaný termín odevzdání:	18. května 2020

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

Ing. Josef Novák, Ph.D.
vedoucí ústavu

V Liberci dne 17. října 2019

Prohlášení

Prohlašuji, že svou diplomovou práci jsem vypracoval samostatně jako původní dílo s použitím uvedené literatury a na základě konzultací s vedoucím mé diplomové práce a konzultantem.

Jsem si vědom toho, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu Technické univerzity v Liberci.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti Technickou univerzitu v Liberci; v tomto případě má Technická univerzita v Liberci právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Současně čestně prohlašuji, že text elektronické podoby práce vložený do IS/STAG se shoduje s textem tištěné podoby práce.

Beru na vědomí, že má diplomová práce bude zveřejněna Technickou univerzitou v Liberci v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů.

Jsem si vědom následků, které podle zákona o vysokých školách mohou vyplývat z porušení tohoto prohlášení.

9. dubna 2020 Bc. Lukáš Vosecký

Poděkování

Chtěl bych především poděkovat vedoucímu mé diplomové práce panu Mgr. Jiřímu Vranému, Ph.D. za podporu, vstřícnost a motivaci k lepším výsledkům. Dále bych rád poděkoval Ing. Jakubovi Vanclovi, který byl mou oporou po celou dobu mé stáže ve Škoda Auto. V neposlední řadě děkuji Karlovi Tvrzníkovi, Milanovi Suchému a Jánovi Suchému za cenné rady a získané zkušenosti.

Zpracování velkých dat logistiky v automotive

Abstrakt

Tato práce řeší problém zpracování, transformace a přenosu dat z platformy Splunk do data lake Cloudera Hadoop a následně do Power BI. Cílem práce je navrhnout a implementovat univerzální a přenositelnou aplikaci v jazyce Python, která bude tento problém řešit. Na základě analýz možností komunikace výše zmíněných systémů je vytvořena univerzální aplikace, která se skládá z několika Python skriptů. Univerzálnost a přenositelnost je zajištěna tím, že se pro jiný zdroj dat ze Splunku bude měnit pouze jeden konfigurační skript a ostatní zůstanou beze změny. Navržená aplikace byla nasazena do produkce a úspěšně řeší první use case pro sklad logistiky, který je v této práci popsán.

Klíčová slova:

zpracování, transformace a přenos velkých dat, big data, Splunk, Python, Cloudera Hadoop, Power BI

Automotive Logistic Big Data Analysis

Abstract

The issue discussed in this work concerns processing, transforming and transferring of data from Splunk platform to Cloudera Hadoop data lake and then to Power BI. The main goal of this work is to design and implement a universal and transferable application in Python language which is supposed to solve this issue. The universal

application consisting of several Python scripts is based on analyses of communication capabilities between the systems mentioned above. For any other source of Splunk type data, there is only one configuration script that needs to be changed, hence the needs of universality and transferability are met. The application was put into production and is now solving first use case in a logistic warehouse which is described in this work.

Key words:

processing, transforming and transferring of big data, big data, Splunk, Python, Cloudera Hadoop, Power BI

Obsah

Seznam obrázků

Seznam zkratek

TB Terabyte

BI Business Intelligence

SW Software

SQL Structured Query Language

CAP Consistency, Availability and Partition Tolerance

ACID Atomicity, Consistency, Isolation and Durability

DML Data Manipulation Language

DDL Data Definition Language

RDBMS Relational Database Management System

DW Data Warehouse

DL Data Lake

ODBC Open Database Connectivity

SPL Search Processing Language

HTTP Hypertext Transfer Protocol

HTTPS Hypertext Transfer Protocol Secure

Úvod

Pojem big data v současné době nemá sjednocenou a ustálenou definici. Existuje jich více, jelikož každý může vnímat vlastnosti velkých dat trochu jinak. Dle mého názoru je nejlépe vystihující Gartnerova definice: Velká data jsou velkoobjemová, rychlá a/nebo různorodá informační aktiva, která vyžadují nákladově efektivní, inovativní formy zpracování informací, které umožňují lepší přehled, rozhodování a automatizaci procesů [3]. Podrobnější vysvětlení se nachází dále v kapitole 1.

Dle [1] je pojem big data v současnosti velice používaný oproti předchozím rokům. Ať už jde o automobilový nebo jiný průmysl, tento pojem se v různých odvětvích vyskytuje často. V návaznosti na to není úplně problém big data získat nebo je generovat. Problém spočívá v jejich zpracování a obecně v manipulaci s nimi.

Existuje mnoho systémů, které dokáží big data zpracovávat, ukládat a transformovat. Správný výběr systému závisí na více ukazatelích, například na cenové dostupnosti, zkušenostech se systémy, jejich společné komunikaci a podobně. V této práci se jedná o následující: Splunk pro získávání dat a jejich prvotní parsování a vizualizaci, Cloudera Hadoop pro jejich ukládání a Power BI pro koncovou datovou analýzu.

Tato práce se zabývá přenosem, zpracováním a transformací dat právě mezi těmito systémy za účelem získání dat ze Splunku do data lake Cloudera Hadoop pro jejich ukládání a následně získání z data lake do Power BI. Jak je popsáno v kapitole 3.2, je více způsobů, jak toho docílit. Cílem je zautomatizovat celý proces získávání dat ze Splunku do Cloudera Hadoop. To hlavně z důvodu zamezení možné chybovosti při ručním exportování dat (do csv souborů), ušetření práce, nákladů a času. Tato úskalí spolu s přesnějším popisem ručního získávání dat ze Splunku

jsou popsána v kapitole 3.1.

Vytvořená aplikace v jazyce Python řeší první přenos dat mezi systémem Splunk a Cloudera Hadoop. Důraz byl kladen na univerzálnost a přenositelnost aplikace. To je z toho důvodu, že přenosů dat pro různé aplikace (myšleno různá data z různých systémů) bude přibývat. V této práci byl řešen přenos dat konkrétně pro sklad logistiky.

V závěru práce je vytvořena datová analýza v Power BI pro otestování funkčnosti celého datového toku a základní ukázková vizualizace.

1 Big Data

Co vlastně termín big data znamená? Na to není bohužel v současnosti jednoduché odpovědět. V dnešní době je to velice používaný termín, který nabývá na popularitě. Současná společnost je obklopena velkým množstvím zařízení všeho možného druhu, která velká data generují nebo již zpracovávají.

Definice

Definice big data má za sebou dlouhý vývoj vzhledem k vývoji technologií a společnosti. V současnosti neexistuje jednotná ustálená definice. Mnoho vědeckých pracovníků a ředitelů společností zavedlo své definice na základě analytických přístupů nebo pro svá využití velkých dat. Například dle útvaru Leadership Council for Information Advantage je big data souhrn nekonečných datasetů (sestavená převážně z nestrukturovaných datasetů) [2]. Tato definice se zaměřuje primárně na velikost dat, což je obecný problém v těchto definicích, protože tato definice unikátně nedefinuje big data od jiného datasetu. Jak je popsáno v kapitole 1.1, charakteristik jako velikost dat je více a proto existují lepší definice, než je tato. Gartnerova definice je pravděpodobně nejlépe vystihující charakteristikou big data: Velká data jsou velkoobjemová, rychlá a/nebo různorodá informační aktiva, která vyžadují nákladově efektivní, inovativní formy zpracování informací, které umožňují lepší přehled, rozhodování a automatizaci procesů [3].

Ovšem z definice nevyplývá přesná velikost dat, jen jejich vlastnosti. Není totiž přesně definováno, jaký objem dat je považován za big data. Lze ovšem předpokládat, že by se mohlo jednat o desítky TB a více.

Business Inteligence

Stejně jako big data, BI nemá jednotnou definici. Tento termín si opět prošel velkým vývojem, ovšem první zmínka byla v roce 1958 ze společnosti IBM [6]. Jedna z mnoha definic je například tato: BI je framework sestávající z množiny konceptů, teorií a metod pro vylepšení obchodního rozhodování skrze pomocné systémy [7]. Zjednodušeně řečeno, jedná se o sadu nástrojů, které pracují s daty za účelem interpretace výsledků. Příkladem jsou nástroje Power BI, SAP BI, Splunk a další.

1.1 Historie

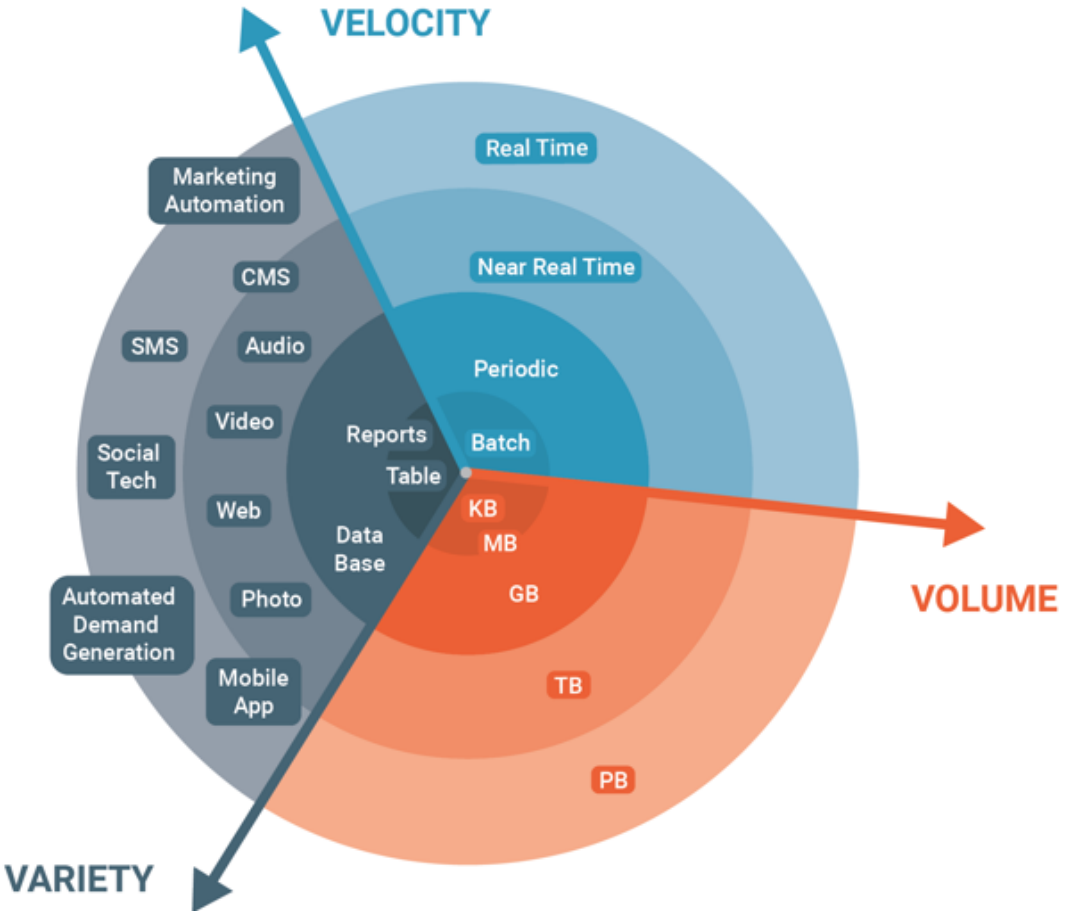
První zmínka o termínu big data pochází z roku 1980. Výzkumníci z Oxford English Discovery zjistili, že sociolog Charles Tilly byl prvním člověkem, který použil termín big data v jedné větě svého článku. Zajímavostí je, že již v roce 1944 Fremont Ryder spekuloval o tom, že v roce 2040 bude mít knihovna Yale 200 miliónů záznamů kvůli explozi informací.

Mezi lety 1997 a 2000 byl termín big data používán v různých akademických článcích. Avšak v roce 2001 přišel první zlomový bod ohledně tohoto termínu. Byla to takzvaná „3 V's of big data. High-volume, High-velocity and High-variety.“ Autorem je Doug Laney [3]. Doug Laney poukazuje na velikost dat (high-volume), rychlost, kterou jsou data generována (high-velocity) a různorodost dat (high-variety).

Modifikace modelu 3 V's je 4 V's, která přišla v roce 2011 od společnosti IBM. Volume, Velocity a Variety zůstávají, ale je k nim navíc přidáno Veracity, což značí kvalitu dat, která jsou dostupná k analýze. Pokud je low veracity, znamená to, že se v datech nachází větší procento nepotřebných nebo nezajímavých údajů. Do toho vstupují i chybějící údaje a podobně.

Je možné říci, že rok 2005 byl pro termín big data zlomový. V roce 2005 Tim O'reilly publikoval článek „What is web 2.0“, ve kterém byl použit termín big data v moderním kontextu [5]. Zároveň v tento rok byl vytvořen Framework Hadoop společností Yahoo!. Hadoop byl nasazen nad již existující model MapReduce od společnosti Google, který byl vytvořen v roce 2004. Obě tyto technologie jsou klíčové

pro práci s velkými daty.



Obrázek 1.1: 3 V's of big data. Převzato z [4].

Problém v dobách bez těchto systémů spočíval v tom, že se data dala pouze získávat, ale nebylo možné s nimi dále efektivně nakládat. Neexistovaly žádné propracované systémy pro jejich zpracování jako Hadoop a MapReduce. Tyto technologie přišly až v nadcházejících letech.

Následující roky přicházely vyhledávací systémy, NoSQL databáze a další, o kterých je napsáno v dalších kapitolách.

1.2 Současný stav

Big data jsou spojena s velkými datasety a s velikostí dat, která je nad rámec flexibility běžných relačních databází k získání, uložení, zpracování a vyhodnocení dat.

V dnešním digitálním světě jsou data generována z různých zdrojů a velkou rychlostí přemisťována z jednoho místa na druhé. Analýza velkých datasetů umožnila obrovský posun v mnoha odvětvích.

1.2.1 Překážky

Soukromí

Soukromí je jednou z největších překážek nejen v oblasti velkých dat. Naprostá většina lidí má obavy ohledně zpracování jejich osobních informací. Jak bylo již zmíněno, big data jsou všude kolem nás a zařízení, která lidé vlastní, je využívají. Příkladem by mohla být analýza zákazníků na základě nakupovacích vzorů: kde zákazníci nakupují, co nakupují a za kolik peněz. To už samozřejmě supermarkety v dnešní době využívají pomocí karet pro zákazníky.

Šum v datech

Bohužel data nejsou vždy čistá a proto je potřeba je od šumu vyčistit. Tomuto procesu čištění dat se říká data cleansing nebo data wragling. Jde o situaci, kdy v atributech některé hodnoty chybějí, nebo kdy obsahují hodnoty, které tam nepatří (některé jsou navíc) a podobně. Často se stane, že z větší množiny dat je ve výsledku znatelně menší množina. Jde v podstatě o takovou hru, kdy se hledá jen to podstatné a data se různě transformují do přehlednější podoby.

1.2.2 Datová úložiště

Pravděpodobně nejjednodušším způsobem ukládání dat v počítačích je stále uložení do souboru. K tomu není zapotřebí žádný speciální SW, pouze textový nebo binární soubor. Není potřeba žádné speciální připojení, stačí mít přístup k souboru a práva zapisovat do něj. Nevýhodou se stává nekonzistence dat souboru a jeho následné prohledávání. Při zvětšující se velikosti souboru je obtížnější jeho zpracování. V dnešní době se kvůli big data mluví o NoSQL databázích, přitom současné klasické relační databáze jsou mnohdy lepším řešením.

Relační databáze

Relační databáze mají za sebou dlouhou historii vývoje, a tím pádem stojí na pevné půdě. Jedna z jejich největších výhod je dotazovací jazyk SQL. Ten má pevný matematický základ, a proto jsou v mnoha případech relační databáze lepším řešením než NoSQL. Relační databáze je typ databáze, která ukládá a zprostředkovává přístup datovým bodům, které jsou společně propojeny. Zároveň jsou založeny na relačním modelu, což je intuitivní přímočará cesta reprezentování dat v tabulkách. [8]
Výhody:

- SQL
- Pevně daná struktura
- Rozsáhlá komunita
- Transakce
- Rychlost vyhledávání

Nevýhody:

- Vertikální škálovatelnost (vyšší cena)
- Při růstu objemu dat roste složitost udržitelnosti a flexibility

NoSQL databáze

Databáze NoSQL (Not only SQL) je databáze, která se používá pro ukládání velkého množství dat. Databáze jsou distribuovaná, nerelační, open source a horizontálně škálovaná [9]. Jsou navržena pro distribuovaná datová úložiště. V současné době se neukládají pouze data ve formě textu. Je potřeba uložit sociální vazby (grafy), geografická data, logy systémů (kdy jejich velikost často exponenciálně roste), a právě proto jsou navrženy NoSQL databáze. Pravděpodobně největší nevýhodou oproti relačním databázím je absence transakcí ve většině NoSQL databází. Zde platí takzvaný CAP teorém. CAP teorém znamená konzistenci, dostupnost a oddílovou toleranci.

- Konzistence: Data dostupná na všech strojích po aktualizacích a dalších akcích
- Dostupnost: Data musí být vždy dostupná
- Oddílná tolerance: Po dobu chybovosti stroje nebo nějaké chyby databáze musí fungovat v běžném režimu bez zastavení činnosti

Bohužel nelze splnit všechny tři možnosti, a proto se volí kombinace po dvou dle požadavků. Typicky se volí kombinace konzistence a dostupnosti.
Výhody:

- Možnost volení databáze dle datového modelu
- Nevyžadují pevné schéma databáze
- Podpora nestrukturovaných dat a nepředvídatelných dat

Nevýhody:

- Nepoužívají se JOIN operace
- Nemá deklarativní dotazovací jazyk
- Případná konzistence upřednostněna před ACID vlastnostmi

Dokumentově orientované NoSQL databáze

Dle průzkumu StackOverflow z roku 2019 je nejrozšířenější NoSQL databází MongoDB z této kategorie [10]. Na rozdíl od relačních databází, kde jsou data uložena v tabulkách, dokumentově orientované databáze mají dokumenty. Dokumenty lze zhruba přirovnat k řádkům v tabulce, ale s tím rozdílem, že dokumenty jsou mnohem více flexibilní, protože jsou bez předem daného schématu (schema-less). Dokumenty bývají standardního formátu (xml, json). Zde je opět vidět velký rozdíl od relačních databází, protože jsou řádky v tabulce často identické (myšleno v ideálním případě, kdy jsou data konzistentní a například v atributu datum se opravdu nachází datum), kdežto v případě dokumentově orientovaných databází může být struktura jednotlivých dokumentů naprosto odlišná, ale také více či méně podobná [11].

Grafové databáze

Grafové databáze jsou v podstatě založeny na teorii grafů. Grafy jsou složeny z vrcholů, hran a jejich ohodnocením (vztah). V těchto databázích vrcholy znamenají entity, ohodnocené hrany mezi vrcholy značí atributy a hrany reprezentují vztah mezi vrcholy.

V relačních databázích je obrovský problém naznačit vztahy mezi objekty, a to je právě silnou stránkou grafových databází. Pravděpodobně nejznámějším příkladem je databáze Neo4j [12].Ta umožňuje efektivní grafové zpracování v reálném čase pomocí přímého indexovaného přístupu k sousedním uzlům z daného uzlu [13].

Některé grafově orientované databáze mohou splňovat ACID. Ovšem vzniká otázka za jakou cenu. Pointa používání NoSQL databází je v jejich síle škálovatelnosti a práce s big daty, ovšem pokud jsou použity "levně", jsou NoSQL databáze použity jako alternativa k RDBMS.

Databáze typu klíč-hodnota

Jak název napovídá, data jsou uložena v páru klíč hodnota. Dalo by se říci, že tento typ databází je v podstatě rodičem všech NoSQL databází [11]. Klíč je unikátní identifikátor k daným datům. Nejpoužívanějším zástupcem tohoto typu databáze je dle dotazníku Redis [10]. Zároveň je to dle hodnocení nejvíce oblíbená databáze mezi vývojáři.

Data Warehouse

Data warehouse (DW) je velkokapacitní úložiště, které je umístěno nad databázemi. Je navrženo pro ukládání středně velkých strukturovaných dat pro časté a opakované analýzy. Časté využití DW je ke sdružování dat z více zdrojů. Některé DW jsou schopné pracovat s nestrukturovanými daty, ale není to běžné. Protože jsou data strukturovaná, schéma je determinováno ještě před tím, než mohou být data přidána do DW.

Charakteristiky:

- Data jsou typicky nahrávána z transakčních systémů a zároveň jsou očištěna
- Zachycuje data a organizuje do schémat
- Schéma je definováno ještě před tím, než se data nahrají
- Použití pro generování reportů a dashboardů

Typickými zástupci DW jsou SAP Bussiness Warehouse, Snowflake a Oracle Exadata Database Machine.

Data Lake

Data lake (DL) je centralizované úložiště navržené pro ukládání strukturovaných i nestrukturovaných dat jakéhokoliv typu a bez velikostních limitů. Typicky se data přenesou rovnou ze systémů, které je generují, přímo do DL bez jakékoliv úpravy. Každému datovému elementu DL přiřadí unikátní identifikátor, který je uložen, a následně umožňuje lehčí vyhledávání pomocí dotazů. DL je spíše považován za big data platformu pro nestrukturovaná data.

Charakteristiky:

- Data jsou typicky nestrukturovaná, ve své původní podobě
- Ideální úložiště pro hluboké analýzy nestrukturovaných dat s pomocí analytických nástrojů, strojového učení a podobně
- Schéma je definováno až po uložení dat. Díky tomu není potřebná prvotní režie a důsledkem je vyšší flexibilita

Typickými představiteli DL jsou Hadoop, Microsoft Azure Data Lake, Amazon AWS Data Lake.

1.2.3 Datová analýza

Datová analýza je proces, který přinese datům smysl [14]. Obecně se skládá z několika kroků, které na sebe navazují. Zároveň se mohou jednotlivé kroky měnit

v závislosti na tom, jestli jsou data strukturovaná nebo nestrukturovaná, pokud jsou již dopředu připravená a podobně. Pokud data nejsou nijak očištěná, to znamená, že jsou v původní podobě, je prvním krokem očištění dat. To je proces, kdy je potřeba projít velkou částí datasetu, najít chybějící a nepotřebná data, duplicitní hodnoty, anomálie a nějakým způsobem napravit tyto nedostatky. Zde je potřeba znát každý atribut a význam celého datasetu, jelikož odstranění některých atributů, byť jejich hodnoty mohou být například někde prázdné a podobně, může být kritickou chybou pro datovou analýzu. V dalším kroku probíhá analýza kvality datasetu pomocí statistických metod, jako výpočet mediánu, směrodatné odchylky a podobně. V tomto kroku se zjistí, jak je dataset rozložen a jaké má vlastnosti. Následně již může probíhat samotná datová analýza pomocí analytických nástrojů jako Splunk 2.1, Power BI 2.3 a další.

2 Nástroje pro big data

V této kapitole jsou popsány nástroje, které jsou použity v této práci. Nástrojů existuje více, ale cílem bylo propojit právě tyto konkrétní nástroje pro práci s velkými daty.

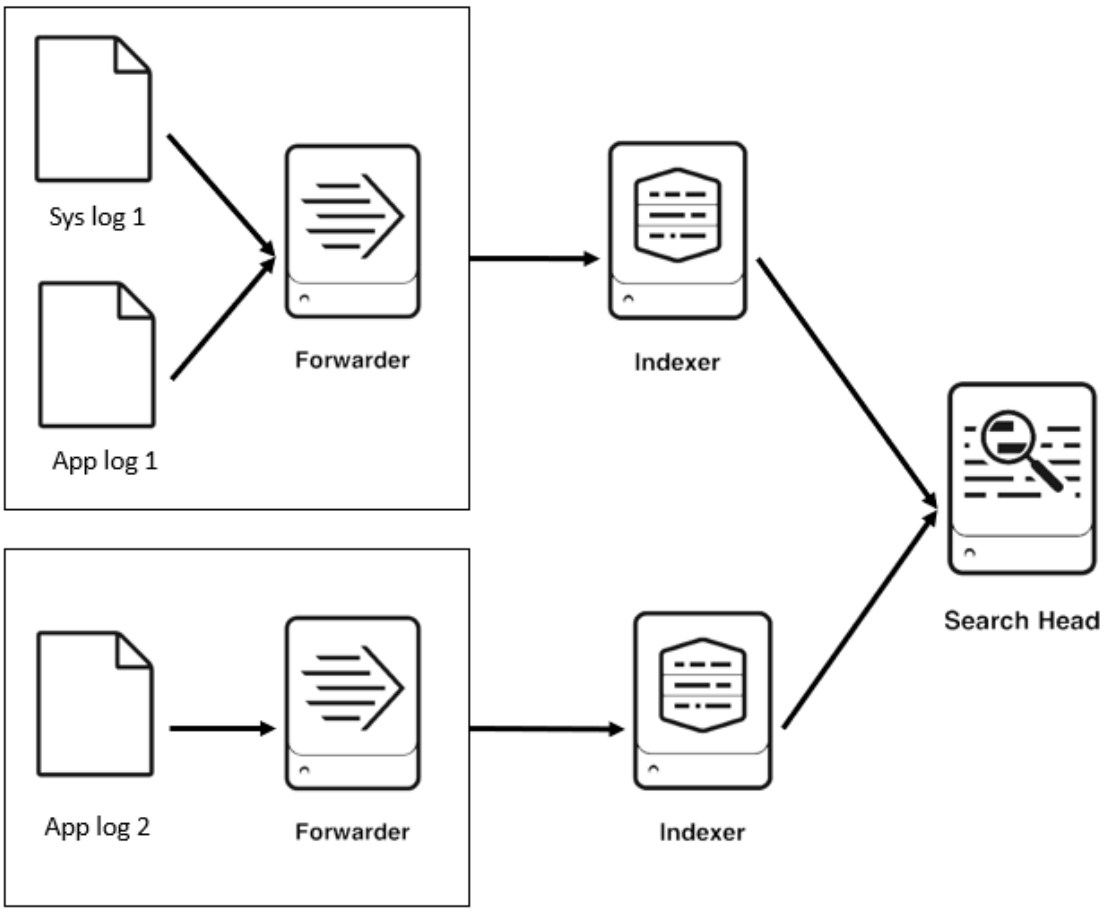
2.1 Splunk

Splunk je softwarová platforma, která slouží primárně ke zpracování a vizualizaci typicky strojových dat (například data ze senzorů, robotů, komunikačních protokolů (MQ, MQTT) a další). Splunk akceptuje téměř jakýkoliv formát dat hned po instalaci. Jinými slovy, Splunk nemá pevně definované schéma. Naopak provádí extrakci atributů v čase hledání. Mnoho datových formátů je rozpoznáno ihned (json, csv...), ty které nejsou, mohou být specifikovány v konfiguračních souborech nebo až při hledacích výrazech. Splunk je tedy nástroj pro vyhledání, analýzu a reporting velkého množství dat, typicky strojových dat v reálném čase. Tento nástroj je optimalizovaný pro rychlé indexování a načítání perzistentních nestrukturovaných dat do systému. Splunk je nástroj, který primárně používá logy. Zároveň nemusí fungovat jako nástroj pro reporting v reálném čase, i když v tom je jeho velká síla. Soubory do něj mohou být nahrány jednotlivě pro jednoúčelové datové analýzy. Během fáze indexování, kdy Splunk zpracovává příchozí data, indexer udělá velký zásah: oddělí od sebe jednotlivé události, kdy jedna událost koresponduje s jedním záznamem v souboru. Každé události přidá timestamp a některé další atributy jako například stroj, ze kterého záznam pochází. Poté jsou klíčová slova události přidána do indexového souboru pro zrychlení pozdějšího vyhledávání a samotné textové události

jsou komprimovány do souborů přímo v souborovém systému.

2.1.1 Nasazení Splunku

Nasazení spočívá v tom, že se nainstaluje forwarder na místo (typicky server), kde je uložen log soubor. Tento forwarder přeposílá data z logu do indexeru. Indexer je komponenta, která je umístěna již na Splunk serveru, a která efektivně uchovává data na nějakou definovanou dobu. Tato data jsou indexována pro rychlé vyhledávání a nad nimi je již použita finální komponenta search head, pomocí které se provádí analýza dat. Vzorový model by mohl vypadat například takto.



Obrázek 2.1: Vzorový model Splunk architektury

Na tomto modelu lze vidět, že každá aplikace má svůj vlastní forwarder. To je z toho důvodu, že instalace forwarderu je umístěna k samotným logům. Zároveň již

na forwarderu se mohou filtrovat data, která má posílat indexeru, což je pro každou aplikaci jinak. Indexer již nemusí být jiný pro různé aplikace.

2.1.2 Search Processing Language

Tento jazyk obsahuje mnoho příkazů, funkcí a argumentů, které jsou napsány tak, aby bylo použití co nejjednodušší a zároveň efektivní za účelem získání požadovaných výsledků z dat. Existují následující komponenty SPL:

Hledané výrazy

To jsou konkrétní výrazy, které jsou psány ve vyhledávacím boxu pro získání specifických záznamů z dat, která splňují daná kritéria.

```
1 index=* host="bro_http" 404 | table time src url
```

Zdrojový kód 1: Ukázka SPL na testovací dataset

Tento search job prohledá všechny indexy. Host se označuje konkrétní zdroj zařízení. Najde události, ve kterých se vyskytuje řetězec 404, a následně data přetransformuje do podoby následující tabulky.

time	src	url
2020-03-09T23:59:56.210-0500	10.11.36.4	/forgotpassword.php
2020-03-09T23:59:52.530-0500	10.11.36.22	/traq/admincp/plugins.php?newhook
2020-03-09T23:59:34.210-0500	10.11.36.10	/forgotpassword.php
2020-03-09T23:59:31.720-0500	10.11.36.48	/forgotpassword.php
2020-03-09T23:59:29.660-0500	10.11.36.48	/traq/admincp/plugins.php?hooks&plugin=1

Obrázek 2.2: Ukázka výsledku vyhledávání z testovacího datasetu

2.1.3 Použití

Splunk je výrazně používaný ve velkých společnostech, typicky tam, kde jsou výrobní linky. Forwarder je nasazen na úložných místech, kde jsou ukládány logy, ať už

z výrobních systémů nebo z jiného místa. Při nějaké změně, nebo v definovaných intervalech forwarder posílá data do Splunku, kde probíhá datová analýza a její výsledky jsou zobrazeny pomocí grafů, statistik a tabulek v reportech, dashboardech a alertech.

Reporty

Reporty jsou výsledky vyhledávacích dotazů, které mohou zobrazit statistiky a vizualizace událostí. Reporty mohou být spuštěny kdykoliv a mohou zachytit nejnovější data při každém spuštění. Zároveň mohou být sdíleny s ostatními uživateli a hlavně mohou být přidány do dashboardů.

Dashboardy

Dashboard je kolekce objektů (reportů, odkazů a podobně). Umožňují nám kombinovat více reportů dohromady, a tím ucelit příběh dat na jedno velké plátno. Dashboard se skládá z panelů, které v sobě mají grafy, statistiky a podobně, což jsou jednotlivé reporty.

Alerty

Alerty jsou akce, které se spustí při specifické události, kdy jsou splněny určité podmínky definované uživatelem. Cílem alertů je získat například logování akcí, které jsou nějakým způsobem kritické a tyto alerty odeslat pomocí e-mailu nebo na specifický endpoint.

Časovače

Časovače slouží k nastavení triggerů pro spouštění reportů automaticky bez uživatelského zásahu. Ty mohou být dle definice spouštěny v různých intervalech: měsíčně, týdně, denně nebo pro specifický časový rozsah. Tím může dojít k zlepšení výkonu (rychlosti) v dashboardech při otevření uživatelem. Časovače disponují možností automatického zasílaní reportu po skončení činnosti.

2.1.4 Další vlastnosti

Splunk disponuje mnoha addony a sadami nástrojů, které se dají přidat k základní verzi. Některé jsou samozřejmě placené. Zajímavé addony pro Splunk jsou například Splunk Analytics for Hadoop - pro ucelené vyhledávání a analyzování Hadoop dat se Splunk Enterprise. Následně různé konektory pro přípojení k databázi (ODBC, DB Connect), mobilní addon a Amazon Web Services [15]

Velmi zajímavý nástroj pro Splnuk je Splunk Machine Learning Toolkit, který disponuje knihovnami pro machine learning a Pythonem spolu s knihovnami Pandas, NumPy, SciKit, SciPy a dalšími. Tímto způsobem je možné vyřešit situaci získání dat ze Splunku pro machine learning.

2.2 Apache Hadoop

Apache Hadoop je framework, který umožňuje distribuované zpracování velkých datasetů napříč clustery s využitím jednoduchých programovacích modelů. Je navržen pro škálování od jednoho serveru až k tisícům strojů, kde každý z nich nabízí lokální komunikaci a ukládání. Abychom se nemuseli spoléhat na hardware pro doručení vysoké dostupnosti, Hadoop je navržen tak, aby detekoval a vyřešil selhání na aplikační vrstvě. Základní myšlenka je taková, že se data rozdělí a uloží napříč kolekcí strojů (cluster). Poté je na řadě práce s daty na místě, kde jsou skutečně uložena. Tedy v tomto případě už v clusteru. V této fázi je jednoduché přidávat stroje do clusteru dle růstu dat.

2.2.1 Hadoop ekosystém - základní moduly

Hadoop se skládá z mnoha modulů, některé jsou povinné a některé lze přidávat a odebírat dle potřeby řešení.

Hadoop HDFS

Hadoop HDFS je distribuovaný souborový systém, který pracuje s velkými datasety. Je to nejspodnější vrstva celého Hadoop ekosystému pro ukládání dat. Data mohou být téměř v jakékoliv formě (json, csv, txt, ...).

Soubor, nahraný do HDFS, je rozdělen do několika bloků o velikosti 64 MB (základní velikost), kde každý blok dostane své unikátní jméno. Po nahrání souboru do clusteru bude každý blok uložen do jednoho nodu v clusteru. Na každém stroji v clusteru běží takzvaný DataNode. O tom, jakým způsobem získáme z rozdělených bloků zpět původní soubor, se stará NameNode. Informace uložené v NameNode se nazývají Metadata. V rámci bezpečnosti existuje kopie NameNodu pro případ výpadku hlavního NameNodu. Další bezpečnostní prvek je takový, že Hadoop vytvoří tři kopie každého bloku souboru a náhodně je rozdělí do třech nodů.

Jeden z hlavních cílů HDFS je rychlé zotavení z hardwarových chyb. Protože jedna HDFS instance se může skládat z několika tisíc serverů, selhání některého z nich je nevyhnutelné. HDFS byl postaven tak, aby detekoval tato selhání a automaticky se z nich zotavil. Jinými slovy, HDFS a ostatní hlavní moduly Hadoopu předpokládají, že hardwarové chyby mohou nastat, a tím pádem jsou připraveny na rychlé a automatické zotavení.

Hadoop YARN

Základní myšlenkou Yarnu je rozdělení funkcionalit řízení zdrojů a plánovače úloh na rozdělené daemony. Myšlenka je taková, že existuje jeden centrální správce zdrojů a potom pro každý daemon jeden aplikační správce.

Hadoop MapReduce

MapReduce je model pro paralelní zpracování velkého množství dat. Jelikož sériové zpracování velkého souboru je pomalé, MapReduce je navržen tak, aby zpracovával data paralelně. Soubor je tedy rozdělen do bloků a každý je zároveň zpracováván.

MapReduce se rozděluje na dvě části. První je mapovací, kdy se nejdříve seskupí společné atributy s hodnotami (key, value) podle klíče. Takto seskupené části jsou následně dle úlohy poslány na redukční část, kde jsou data již seřazena a připravena k finální úpravě. Například, mějme dataset měst s obchody a jejich tržbami. V mapovací části se seskupí stejná města (key) a jejich tržby. Následně takto setříděná města jsou zvlášť poslána redukční části, kde každý „reducer“ počítá roční tržby pro jedno město.
Psaní MapReduce kódu je podporováno jazyky Python, Java, Ruby a dalšími.

Hadoop Common

Hadoop Common je kolekce běžných utilit a knihoven, které podporují ostatní moduly. Je to nezbytná část celého frameworku spolu s Yarn, MapReduce a HDFS. Je brán jako základní/klíčový modul celého frameworku, protože zprostředkovává základní služby jako například abstrakci operačního systému, na kterém je framework nasazen, a i jeho souborového systému.

2.2.2 Hadoop ekosystém - přídavné moduly

Přídavných modulů je opravdu mnoho, proto jsou zde vypsány pouze ty nejznámější, které jsou s touto prací do jisté míry spjaty.

Psaní MapReduce kódu není úplně snadné (je vyžadována znalost některého programovacího jazyku podporovaného MapReduce - Java, Python apod.). Proto vznikly nástroje jako je Impala a Hive. Namísto psaní kódu tyto nástroje umožňují využít SQL pro dotazování. Další možností je Pig, který umožňuje analyzovat data pomocí jednoduchého skriptovacího jazyku.

Impala

Apache Impala je paralelně zpracovávající SQL dotazovací nástroj pro data, která jsou uložena v clusteru běžícím na Apache Hadoop. Impala podporuje HDFS

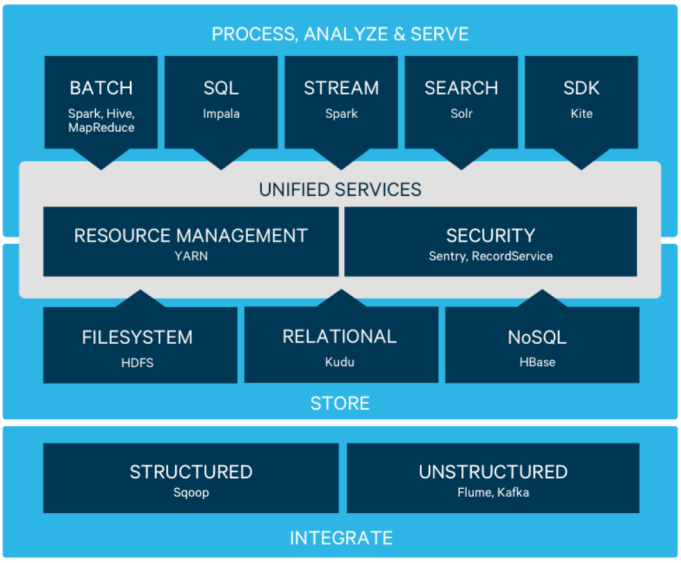
i Apache HBase, dále podporuje autentizaci pomocí Kerberos. Největší výhoda Impaly je způsob dotazování na HDFS. Impala totiž nevyužívá MapReduce, a tedy se dotazuje na přímo. Tím dojde k ušetření času pro startování MapReduce. Používá se tedy pro rychlé analýzy nebo pro velké datasety. Často je Impala používána jako nástroj pro získání dat do Power BI pomocí direct query.

Hive

Hive je pomalejší alternativa k Impala, a to z důvodu využití MapReduce. Hive interpreter přemění SQL na MapReduce kód, který je poté spuštěn na clusteru. Jinými slovy, při každém dotazu je nutné spustit MapReduce job. Což může být opravdu pomalé při velkém množství dat. Proto se Hive spíše používá při menším množství dat, nebo u aplikací, kde nezáleží na čase dokončení. Hive je optimalizovaný pro spouštění dlouhých batch-processing jobs.

Hue

Apache Hue je open source online editor, který slouží pro práci s daty uloženými v HDFS pomocí SQL. Umožňuje použít několik interpretrů (Impala, Hive, MySQL, SparkSQL a další). Zároveň umožňuje generování grafů a statistik.



Obrázek 2.3: Cloudera Hadoop ekosystém. Převzato z [16].

2.2.3 Hadoop distribuce

Propojit všechny tyto přídavné modely dohromady s hlavními částmi Hadoop ekosystému je obecně celkem náročné. Všechny moduly Hadoop ekosystému jsou open-source. Jen některé moduly spolu nejsou kompatibilní a může nastat hodně komplikací. Proto existují již různé distribuce, například CDH, která zabalí celý ekosystém s přídavnými moduly dohromady pro snadnou instalaci.

Cloudera

Společnost Cloudera vlastní Hadoop distribuci nesoucí název Cloudera distribution including Apache Hadoop (CDH). Je to open source platformní distribuce zahrnující Apache Hadoop, která je postavena tak, aby splňovala požadavky společností. Tato distribuce zároveň obsahuje mnoho dalších kritických open source projektů, které s Hadoop souvisí. Obsahuje tedy Hadoop core, Hive, HBase, Impala, Hue a mnoho dalších [17]. Zároveň obsahuje systémy, které pomáhají s integrací dat a celého systému.

mapR

Alternativní distribucí je mapR. Jedná se o více univerzální distribuci, protože není postavená čistě na HDFS. MapR má svůj vlastní souborový systém, MAPRFS. To přináší své výhody, hlavně co se týče bezpečnosti.

2.3 Power BI

Power BI je nástroj od společnosti Microsoft, který se používá pro datovou analýzu. Skládá se z mnoha konektorů, služeb a aplikací. Je možné ho použít v podobě desktopové nebo mobilní aplikace. Power BI disponuje mnoha konektory pro načtení dat, jako načtení ze souboru, z databáze nebo z cloudové či jiné datové platformy. Například pro Hadoop existuje Power BI konektor pro Impalu. Protože je objem dat často velký a každá aktualizace dat (například z databáze) trvá delší

dobu (v závislosti na objemu dat), Power BI disponuje takzvaným direct query. To umožňuje načítat data ze zdroje definovaný čas (pouze nová data).

Práce s Power BI při tvorbě reportů je celkem intuitivní, ale zároveň to neubírá na účinnosti. To stejné platí i pro načítání dat z různých zdrojů. Při práci se souborem (například csv), Power BI pozná oddělovač a podle něj rozdělí jednotlivé atributy. Pokud by ho náhodou nerozpoznal, je možné ho ručně určit.

Při práci s malým objemem dat nebude v desktopové verzi problém. Práce s větším počtem dat může být už limitující. Například při práci s několika GB dat z databáze se může zdát, že aktualizace grafů je poněkud pomalá. Je to z toho důvodu, že po vytvoření datových připojení a transformaci dat, jsou data načtena do datového modelu přímo do aplikace. Jedna z hlavních předností Power BI jsou propojené komponenty v jednom reportu. To znamená, že pokud jsou v reportu vytvořeny vizualizace (graf, tabulka) a zároveň nějaké filtrování, tak se potom přenese filtrování na každou vizualizaci. Zároveň vytvořených reportů může být více a některé (nebo všechny) komponenty a filtry mohou být použity napříč jednotlivými reporty.

3 Návrh řešení

V této kapitole je vysvětleno, jaký je současný stav získávání dat ze Splunku pro sklad logistiky. S tím souvisí popsání situací, ve kterých vzniká chybovost. Následně je popsáno, jaké jsou možnosti komunikace mezi použitými systémy. Z této analýzy jsou vybrány nejlepší způsoby, které jsou následně aplikovány.

3.1 Současný stav

Ve skladu logistiky jsou autonomní roboti, kteří vykládají boxy do regálů a také je nakládají na pás. Zároveň ukládají své stavy a chyby do souborů. Tyto soubory jsou prohledávány a jejich data jsou nahrávána pomocí Splunk forwarderu do Splunku, kde probíhá datová analýza. Problém je v tom, že přístup ke Splunku je do jisté míry omezen a práce s ním vyžaduje pokročilé znalosti. Jinými slovy, tvorba reportů ve Splunku není tak jednoduchá, jako například v Power BI. To znamená, že reporty a dashboardy ve Splunku tvoří skupina datových specialistů. To stejné platí pro jejich sebemenší změny. Dalo by se totiž říct, že Splunk není určený primárně pro business view.

Konkrétně se jedná o chyby při vykládání boxů a jejich další manipulaci. Ve Splunku jsou data očištěna a parsována do použitelné podoby pro následné vykreslení tabulek a grafů. Vždy na konci směny (tedy po 8 hodinách: 6:00, 14:00, 22:00) zaměstnanec přistoupí ke Splunku, exportuje naparsovaná a očištěná data do souboru typu csv, který stáhne a nahraje do předem definované složky s určitým názvem. V této složce se následně soubory nahrávají do Power BI. Tento proces je velice neefektivní a zdlouhavý. Při tomto procesu vzniká zároveň velká chybovost. Exportované

soubory musí být vždy na konci směny uloženy na stanovené místo s předem definovaným jménem. Často se stává, že tato kritéria nejsou dodržena a následně vznikají další problémy. Primárně z tohoto důvodu vznikla tato práce, aby byl celý tento proces efektivnější a univerzální. To znamená jakákoliv data ze Splunku uložit do data lake a následně je získat do platformy Power BI.

Kritická místa pro tento use case jsou tedy následující:

- Složitá tvorba reportů ve Splunku (je potřeba skupina datových specialistů)
- Ukládání souboru na správné místo
- Zadávání správného názvu exportovaného csv souboru
- Čas trvání ručního exportování souboru
- Čas strávený opravou po případném chybném uložení souboru

Kromě prvního bodu jsou ostatní způsobené lidskou chybou, kterou bohužel nelze vždy ovlivnit. Pakliže soubor není na správném místě se správným jménem, nelze ho automaticky nahrát do Power BI. Je tedy potom potřeba soubor najít a napravit chybu.

Ovšem obecně se jedná o to, že pokud jsou potřeba data ze Splunku získat, v současné situaci je vždy potřeba data ručně stáhnout. Jde tedy o automatizaci celého tohoto procesu získávání csv souborů pro libovolný datový zdroj ze Splnuku.

3.2 Analýza způsobu komunikace mezi systémy

Na úplném začátku je vždy potřeba zanalyzovat systémy, které spolu budou nějakým způsobem komunikovat. V této práci se jedná hlavně o systém Splunk, Cloudera Hadoop a Power BI. Existuje několik způsobů, jak přenášet data mezi těmito systémy. Vzhledem k tomu, že data jsou již odesílána pomocí Splunk forwarderu na Splunk indexy, stačí se v tomto případě zaměřit na část přenášení dat mezi Splunkem, Clouderou Hadoop a následně Power BI. Data ze Splunku do Cloudery Hadoop lze přenést více způsoby.

3.2.1 Přenos dat ze Splunku do Cloudera Hadoop

Hadoop connector

První možností je odlévat data přímo ze Splunku do data lake (Cloudera Hadoop) a z něj poté pomocí Impala connectoru do Power BI. Tato možnost je pravděpodobně nejvíce přímočará a zdá se nejjednodušší. Ovšem má to své nevýhody. Tou největší nevýhodou je zatížení Splunku při odlévání dat. Už v této situaci je relativně zatížen a při další větší zátěži by se mohl zpomalit celý jeho chod, což by mělo kritický dopad, jelikož je používán nejen pro analýzu tohoto skladu, ale pro více aplikací. Zvlášť když jsou chyby generovány v podstatě každou minutu.
Z této kritické negativní vlastnosti vyplynulo zamítnutí této metody pro tuto aplikaci. Nicméně tuto metodu je možné použít pro docílení jiné potřeby. Největší zátěž totiž nespočívá v samotném odlévání dat, ale již v parsování dat. Tedy pokud se pouze odlévají nezpracovaná raw data nebo s minimální úpravou, lze tento způsob využít jako zálohu dat, jelikož ve Splunk indexerech data zůstávají přibližně měsíc (konfigurovatelná doba).

Splunk REST API

Splunk disponuje REST API, pomocí kterého lze získávat data, zakládat alerty a podobně. Velkou výhodu tohoto API je to, že ho vystavují i Splunk frontend nody, takže zjednodušeně řečeno, připojení pomocí API nezatěžuje hlavní backendový Splunk node.
Splunk nabízí již připravené balíčky pro práci s REST API pro jazyky Python, Java a další, které velice usnadňují práci [18]. V této práci je použit balíček pro Python. Tento balíček v sobě obsahuje řešení pro autentizaci, autorizaci, získávání dat, zakládání alertů, odesílání souborů do Splunku a další. Dává tedy největší smysl použít právě toto řešení.
Balíček nabízí celkem čtyři možné metody, pomocí kterých lze získávat data:

- **Blocking search.** Tento typ vyhledávání umožňuje vytvořit search job, který běží synchronně v takzvaném blokovacím módu. To znamená, že se job vrátí

až poté, co se získají všechny výsledky. Z job objektu lze poté získat více informací. Například, jak dlouho job trval, kolik bylo vráceno eventů, jaké bylo přiřazeno job ID a další.

- **Normal search.** Normal search vytvoří klasický search job, stejně jako blocking search. Rozdíl je v tom, že normal search vrátí ihned search ID, pomocí kterého je nutné vyhledat search job a následně ho stáhnout. Ovšem opět se musí čekat, než se search job dokončí.

- **One-shot search.** Toto je ta nejjednodušší a nejpřímočařejší metoda. Jedná se o to, že se vytvoří takzvaný jednoúčelový search. Na rozdíl od ostatních metod nevytváří a nevrací search job, ale naopak se zablokuje, dokud není search dokončen a není vrácen stream obsahující eventy. To také ale znamená, že nejsou vráceny informace o searchi. Je vrácen pouze stream eventů a pokud někde nastane nějaká chyba (například v parsování dat nebo v samotném searchi), tak Splunk vrátí chybovou hlášku, která se může například zalogovat. Tato metoda je z těchto důvodů v práci použita, jelikož je ze searche získáno to nejpodstatnější - možná chybová hláška a nebo tok eventů.

- **Export search.** Export search je ta nejvíce spolehlivá metoda, kterou lze získat větší množství dat, protože se eventy vrací jako tok dat na rozdíl od ostatních metod popsaných výše, kdy je na serveru po nějakou dobu uložen search job. Takže jakékoliv limitace ze strany serveru, co se týče objemu dat, pro tuto metodu neplatí. Export search se spustí okamžitě a zároveň hned po spuštění začne přenášet data klientovi.

Tedy Splunk REST API bylo nakonec vybráno jako implementační řešení pro tuto část získávání dat z důvodu velkého množství způsobů, jak s daty pracovat a zároveň z důvodu menšího zatížení Splunku. Toto REST API bude komunikovat se Splunkem z linuxového serveru, kde budou umístěny Python skripty, generující csv soubor.

3.2.2 Přenos dat ze serveru do Cloudery Hadoop

V dalším kroku je potřeba soubor ze serveru nahrát do Cloudery Hadoop. To lze docílit pomocí UC4 jobu.

UC4 job

UC4 je software pro plánované spouštění jobů, díky kterému lze například přenášet soubory napříč architekturami a úložišti. Má mnoho konfigurovatelných parametrů. V této práci je použit právě na přenos csv souboru z linuxového serveru, kde csv vzniká, na cílový linuxový server, kde se csv ukládá do Cloudery Hadoop. Příklady konfigurovatelných parametrů jsou například smazání souboru po přenosu, možnost zaslat informace o chybném stavu a další. Tento způsob byl vybrán z důvodu již otestované funkčnosti na jiných projektech.

3.2.3 Přenos dat z Hadoop do Power BI

Opět je zde více možností, jak docílit požadovaného přenosu. Existuje totiž více connectorů a každý z nich funguje trochu jinak. Pro ukázku jsou zde uvedeny dva příklady, z toho Impala connector je použit v této práci.

ODBC

Klasický ODBC connector nabízí pouze základní jednoduchý import dat. To může být výhodné, pokud se data v databází již nemění nebo se mění jen velmi málo.

Impala connector

Impala connector je nástroj, kterým lze efektivně získávat data z Hadoop, a to hlavně z toho důvodu, že používá optimalizované dotazy pro získání dat, jelikož Impala bývá součástí Hadoop ekosystému. Zároveň nabízí takzvané DirectQuery, což je automatické stahování dat při nějaké změně dat v Hadoop. DirectQuery má tu výhodu, že následně stahuje pouze data, která jsou nová nebo změněná. Díky tomu je Power BI při aktualizaci dat rychlejší, než kdyby se data načítala pomocí klasického

Importu, kdy se načítá vše. Tento connector bude použit z důvodu automatického stahování nových dat, což je vlastnost, která je jedním z požadavků na funkcionalitu řešení.

3.2.4 Přenos logovaných eventů do Splunku

Základní myšlenkou je logovat do souboru vždy po zpracování dat, pokud přenos proběhl úspěšně. To bude platit i ve finální implementaci. Ovšem aby se nemusel implementovat nějaký mechanizmus v Pythonu pro zasílání alertů, je možné využít Splunku pro vizualizaci logů aplikace a případné vytváření alertů.

První možností je instalace forwarderu na serveru, který bude data ze souboru odesílat. Tato varianta je pro tento účel zbytečně komplikovaná, jelikož se eventy vytváří vždy na konci směny (tedy tři řádky za jeden den). Existuje tedy druhý způsob, mnohem elegantnější. Splunk disponuje HTTP Event Collectorem, pomocí kterého lze zasílat data. To znamená, že se data mohou odesílat pomocí curl funkce do Splunku pomocí HEC vždy po zalogování eventu. Tedy například minutu poté, co proběhlo stažení dat a vytvoření eventu do logu. Toto řešení nevyžaduje žádnou další instalaci komponent (jako v předešlé možnosti instalace forwarderu na server). Tedy ve Splunku stačí vytvořit token pro index, do kterého se data budou posílat.

Název indexu by měl odpovídat názvu aplikace, která řeší nějaký problém. V tomto případě to může být například dataextract. V budoucnosti budou přibývat aplikace, které toto řešení budou používat. To znamená, že v jednom indexu bude možné vidět všechny aplikace, které stahují data ze Splunku. Ty se následně budou moci různě filtrovat.

3.3 Návrh univerzální aplikace

Po získání dat pomocí REST API je potřeba data rozparsovat a zapsat do csv souboru. Všechno toto zpracování dat a ukládání do souboru probíhá na linuxovém serveru.

Idea je taková, že budou existovat celkem tři python skripty pro získání dat a je-

jich ukládání do souboru csv. Klíčovým skriptem a teoreticky jediným měnitelným by byl konfigurační skript. V tomto skriptu by byly definovány údaje pro autentizaci, samotného search jobu, parsovacích parametrů, výstupního souboru a logovacího souboru. Následně skript pro autentizaci uživatele, který by byl kompletně univerzální, jelikož přihlašovací údaje by byly definované v konfiguračním souboru. Následně hlavní skript, ve kterém probíhá získání dat ze Splunku, jejich parsování a ukládání do souboru. V tomto skriptu by se teoreticky nemuselo nic měnit, jelikož SPL dokáže dobře získat data pomocí samotného dotazu.

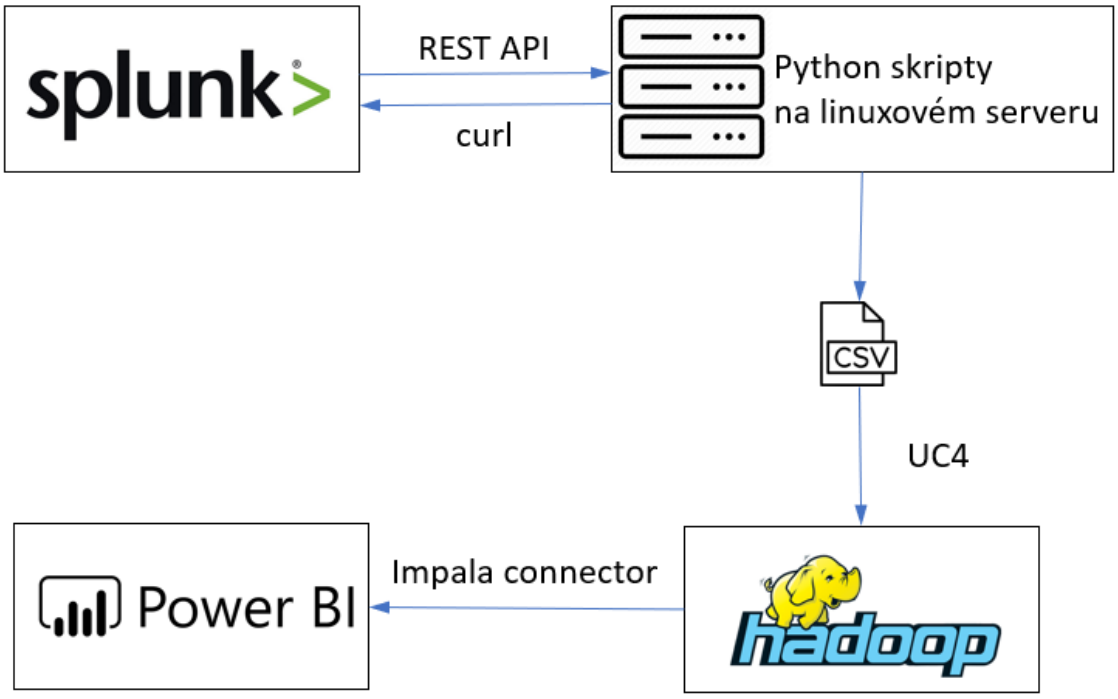
Samotné automatizace spouštění skriptů lze docílit například pomocí cron jobu. V něm se definuje přesný čas, kdy má být skript spuštěn, a tím pádem i čas získaného csv souboru.

4 Implementace řešení

V této kapitole je popsána celková implementace datového toku spolu s vytvořenou univerzální aplikací pro získávání dat ze Splunku. Zároveň s touto aplikací je popsán logovací systém samotné aplikace i její testy.

4.1 Implementace datového toku

Z analýzy použitých systémů popsaných v kapitole 3.2 vyplývá následující schéma, ve kterém je znázorněna celková implementace datového toku.



Obrázek 4.1: Schéma datového toku

Celý proces transformace dat funguje následovně. Cron job spouští hlavní skript, který je popsaný v kapitole 4.2, vždy na konci směny, tedy 6:01, 14:01 a 22:01 příkazem: python3 main.py –c error_codes_config.py. Pomocí přepínače –c se vybere požadovaný konfigurační soubor pro danou aplikaci. Tento soubor je dále popsaný v kapitole 4.2. Dále se tento soubor importuje do hlavního skriptu a následně je zavolán autentizační skript splunk_auth.py, popsaný v kapitole 4.2, s konfiguračním skriptem ve vstupním parametru. V něm se vytvoří session, se kterou se dále pracuje v hlavním skriptu, ve kterém probíhá samotné parsování dat a ukládání do csv souboru. Všechny zranitelné části kódu jsou zabalené v bloku try except, díky kterému nemůže dojít k nečekanému stavu aplikace, a se předejde případnému uložení chybných dat.

Celková doba od získání dat až po vytvoření csv souboru trvá průměrně 5,5 vteřin se směrodatnou odchylkou 1,75. Tento časový údaj byl vypočítán nad eventy za dobu jednoho měsíce od nasazení do produkce. Důvodem je celkem rozsáhlý search job query, který je ve skutečnosti složen ze dvou a je spojen pomocí operace append. Pokud by se search job rozdělil na dva, bylo by potřeba vytvořit druhý konfigurační soubor a nějakým způsobem synchronizovat zápis do jednoho csv souboru. Dále by byl potřeba další cronjob pro spouštění aplikace s jiným konfiguračním souborem. Došlo by tedy k většímu rozsahu aplikace pro tento use case, ale bylo by očekáváno, že by opravdu došlo k rychlejšímu získávání dat. Bylo totiž změřeno, že pokud se search query pustily samostatně ve Splunku, průměrná doba získání dat byla 2 vteřiny. Vzhledem k tomu, že je tento čas zanedbatelný, zůstal jako řešení jeden search job. V následující tabulce je znázorněn rozdíl zpracování dat ve Splunku a v Pythonu při použití jednoho search jobu. Tyto hodnoty byly vypočítány nad eventy za dobu jednoho měsíce od nasazení do produkce.

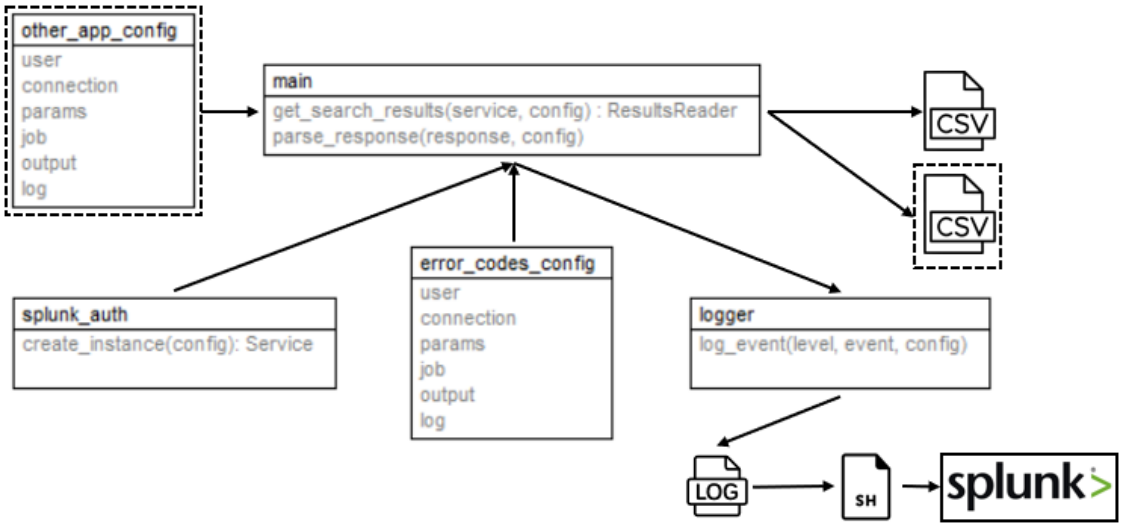
	Python	Splunk
Průměrný čas (s)	5,5	6,4
Směrodatná odchylka	1,75	1,59

Tabulka 4.1: Porovnání hodnot pro zpracování dat ve Splunku a Pythonu

Jakmile je výstupní soubor vytvořen, UC4 job ho přenese na server Cloudery Hadoop, kde probíhá nahrání do data lake a následně jsou data připravena k použití. Původní soubor je smazán, aby se soubory zbytečně nehromadily. Pokud by se náhodou stalo, že by došlo k jakémukoliv výpadku jednoho z jobů nebo serveru a podobně, je možné vždy csv soubor ručně exportovat ze Splunku a nahrát ho na sdílený disk, kde UC4 job tyto soubory v definované časy nahraje do Cloudery Hadoop. Další možností je opět i ruční import csv souborů do Cloudery Hadoop.

4.2 Vytvořená aplikace

Vytvořená aplikace pro tento use case se skládá celkem z pěti python skriptů. Jak již bylo zmíněno, celá aplikace je postavená tak, aby ji bylo možné přenášet z jednoho zdroje dat na další. To ve skutečnosti znamená, že při použití pro jiný use case je potřeba změnit pouze jeden skript, a to právě ten konfigurační.



Obrázek 4.2: Schéma aplikace

Přerušovanou čárou je znázorněno použití další aplikace. Tedy je zapotřebí pouze vytvořit nový konfigurační skript s novými parametry, a tím pádem je potom nově vzniklý csv soubor bez jakýchkoliv dalších úprav.

Konfigurační skript

V konfiguračním skriptu pro jiné datové zdroje je možné měnit následující parametry: samotný search job, výstupní proměnné ze search jobu, interval pro získání dat, výstupní soubor a popřípadě log soubor.
Výstupními proměnnými jsou myšleny parametry, které se z celého search jobu nakonec získají. Aplikace je totiž postavená tak, aby Splunk search job měl na konci formátování do tabulky. To znamená, že search job bude vypadat následovně:

```
1 index=example_search ... | table
```

Zdrojový kód 2: Požadovaný formát search jobu

Za příkaz table (formátování do tabulky) se následně automaticky vloží parametry, které jsou ručně vloženy do datové struktury list před samotným search jobem. Díky tomuto řešení je potom v hlavním skriptu možné automaticky ukládat hodnoty do csv souboru, jelikož názvy sloupců jsou totožné s těmito parametry.
Na výstupní soubor jsou kladené jisté nároky. Tím je myšleno, že soubor musí být pro integraci do Cloudery Hadoop vždy formátu csv a kódování utf-8. Pakliže by se někdy situace změnila, aplikace je na to připravená, jelikož v konfiguračním souboru je oddělený jak název a kódování souboru, tak i jeho přípona. Dalším požadavkem je ukládání timestampu za název souboru, který představuje čas vytvoření souboru. Tedy pro tento konkrétní use case: cakl_errorcodes_YYYYMMDDHHmmss.csv, kde cakl je název aplikace, errocodes je název tabulky a na konci je formát timestampu. Vkládání timestampu je opět řešeno automaticky bez ručního zásahu.

Autentizační skript

Tento skript opět nevyžaduje žádný zásah. Je volaný z hlavního skriptu a jednoduše si získá údaje z konfiguračního souboru a vytvoří si session se Splunk frontend nodem, kterou následně pošle do metody pro získání dat. Tato session má v základní konfiguraci timeout jednu hodinu, což tedy platí i v tomto případě. Samozřejmě je to konfigurovatelný atribut, který lze změnit v obecné konfigurační sekci serveru.

Nejdříve je potřeba vytvořit Splunk uživatele s právy a rolemi pro REST API a obecně REST API povolit na nějakém nodu (tyto vlastnosti nejsou výchozí). Pro vytvoření session je potřeba tedy znát jméno a heslo Splunk uživatele s rolí REST API user, host, port a volitelný parametr scheme, což je typ spojení (HTTP nebo HTTPS).

```
1  NAME = os.path.basename(__file__)
2  def create_instance(config):
3  """ Creates session
4      returns service object
5  """
6      USER = config.user['USERNAME']
7      PWD = config.user['PASSWORD']
8      HOST = config.connection['HOST']
9      PORT = config.connection['PORT']
10     try:
11       service = client.connect(
12         host=HOST,
13         port=PORT,
14         username=USER,
15         password=PWD,
16         scheme="https"
17       )
18       return service
19     except Exception as e:
20         log_event("error", NAME + " " + str(e))
21         sys.exit(1)
```

Zdrojový kód 3: Vytvoření Splunk session

Hlavní skript

Jak bylo již zmíněno, v hlavním skriptu se děje hlavní část transformace dat. Vzhledem k požadavku na univerzálnost a přenositelnost řešení je načítání konfiguračního skriptu řešeno pomocí parametru při spouštění hlavního skriptu. Funkcionalita je ukázána na následujícím bloku kódu.

```
if __name__ == '__main__':
  parser = argparse.ArgumentParser(description='Splunk ETL into hadoop')
  parser.add_argument("--c", type=str,
    help="Enter python config file with extension", required=True)
  args = parser.parse_args()
  config_name = args.c

  if os.path.isfile(config_name):
    # in case of another location of config file, or for cron usage
    config_name = config_name.split("/")[-1]
    # import of config file from args
    config_name = config_name.split(".")
    conf = importlib.import_module(config_name[0])
  else:
    log_event("error", NAME +
      " Config file from command line arguments does not exist: " +
      config_name)
    sys.exit(1)
```

Zdrojový kód 4: Načtení konfiguračního souboru

Po získání a načtení konfiguračního skriptu je vytvořena Splunk session. Pokud tento proces proběhl v pořádku, dalším krokem je odeslání pomocí REST API search job a získání výsledků. Jak bylo zmíněno v kapitole 3.2.1, pro odeslání a získání dat je použit oneshot search. Oneshot search potřebuje ve vstupních atributech samotný search job a slovník parametrů. Mezi tyto parametry patří časový interval, ve kterém se mají data stáhnout. Tento údaj je opět brán z konfiguračního souboru. Je tím myšlen čas nejstarší události. Druhý časový údaj je čas nejnovější události, tedy čas ve chvíli, kdy se operace odehrává. Následně mód search jobu, což je v případě oneshotu normal mode a nakonec limit pro počet eventů. Tento limit je nepovinný parametr a bez jeho definování je limit 100 eventů. Pro získání všech eventů je

potřeba nastavit tento parametr na 0. Celý tento proces získávání dat je časově měřen, a to díky proměnným dt_started a dt_ended. Tyto údaje jsou opět logovány.

```
kwargs = {"earliest_time": INTERVAL,
  "latest_time": "now",
  "search_mode": "normal",
  "count": 0}
# count:0 => return more than 100 events

dt_started = datetime.datetime.utcnow()
try:
  oneshotsearch_results = service.jobs.oneshot(SEARCH_QUERY, **kwargs)
except Exception as e:
  log_event("error", NAME + " "+ config.NAME + " " + str(e))
  sys.exit(1)
dt_ended = datetime.datetime.utcnow()
```

Zdrojový kód 5: Nastavení získání dat

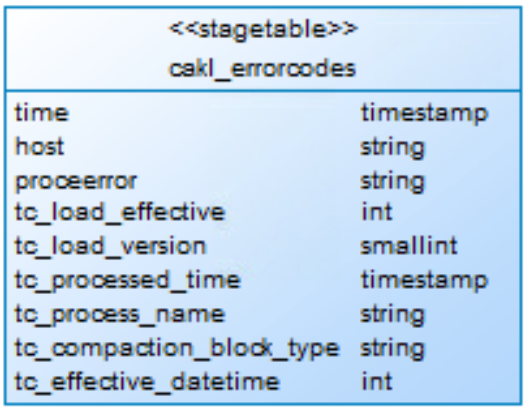
Pokud odeslání search jobu a získání dat proběhlo v pořádku (bez získané výjimky), data jsou následně parsována. Datový typ získaných dat je objekt ResultReader, ve kterém jsou data uložena ve formě slovníku. Ukládání dat do souboru tedy probíhá v cyklu, ve kterém je následně kontrolován typ dat. Pokud je formát slovník, data jsou dle parametrů z konfiguračního souboru ukládána do souboru. Pokud jsou formátu result.Message, jedná se o diagnostické zprávy. Tedy tyto zprávy se ukládají do log souboru. Jedná se totiž o zprávy ze Splunk serveru o možné chybě. Po uložení všech dat je do log souboru uložena zpráva s levelem info o úspěšném uložení dat.

Datový model

Vytváření datového modelu probíhalo v prostředí Power Designer. Datový model je potřeba pro vytvoření prostředí v data lake. V zásadě se jedná o dva json soubory, ve kterých jsou popsána metadata. Například z jakého oddělení data pocházejí, jak moc jsou citlivá, jestli se jedná o jednorázová data nebo o data přírůstková. Pokud jde o data přírůstková, je popsáno, jak často má probíhat kontrola nového souboru na úložišti, optimalizace dat v date lake a podobně. Vzhledem k jednoduchosti datového

modelu této aplikace není potřeba detailně znázorňovat jeho datový model a popisy. Jedná se v podstatě o tři sloupce. Čas vzniku chyby, host a samotný identifikátor chyby.

Datový model integrovaný do data lake tedy vypadá následovně.



Obrázek 4.3: Schéma datového modelu

Posledních šest polí se při vytváření modelu generuje automaticky. Jsou to pole, která jsou reprezentována jako metadata v tabulce. Ta ve své podstatě udávají bližší informace o datech. Tedy kdy byla konkrétní data importována do data lake, jaká je jejich verze a podobně.

Jedinou komplikací při integraci byl timestamp. Impala totiž vyžaduje timestamp v určité podobě. Tím je myšlen tento formát: YYYY-MM-DD HH:mm:ss. Ovšem timestamp získavaný ze Splunku měl podobu následující: YYYY-MM-DDTHH:mm:ss. Tedy písmeno T tam nesmělo být a muselo být nahrazeno mezerou. Dalším problémem byl název tohoto sloupce, jenž byl _time. Ten musel být změněn na time, tedy bez podtržítka. To vše se vyřešilo pomocí následujícího SQL příkazu v definici sloupce.

```
1 "transform":"CAST (from_unixtime(unix_timestamp(substr(replace(`time`,'T',' '),
2   1,19), 'yyyy-MM-dd HH:mm:ss')) AS timestamp) as time"
```

Zdrojový kód 6: Změna formátu timestampu

4.3 Logování a testování řešení

4.3.1 Logování událostí

V tomto řešení jsou logována kriticka místa do souboru. Těmi jsou: vytvoření Splunk session, načítání konfiguračního skriptu při spouštění skriptu, odesílání search jobu, získávání dat a ukládání dat do souboru. Pokud všechny tyto události proběhly v pořádku, je zalogována informace o úspěšném zpracování dat a jejich uložení do souboru.

Při vzniklé chybě

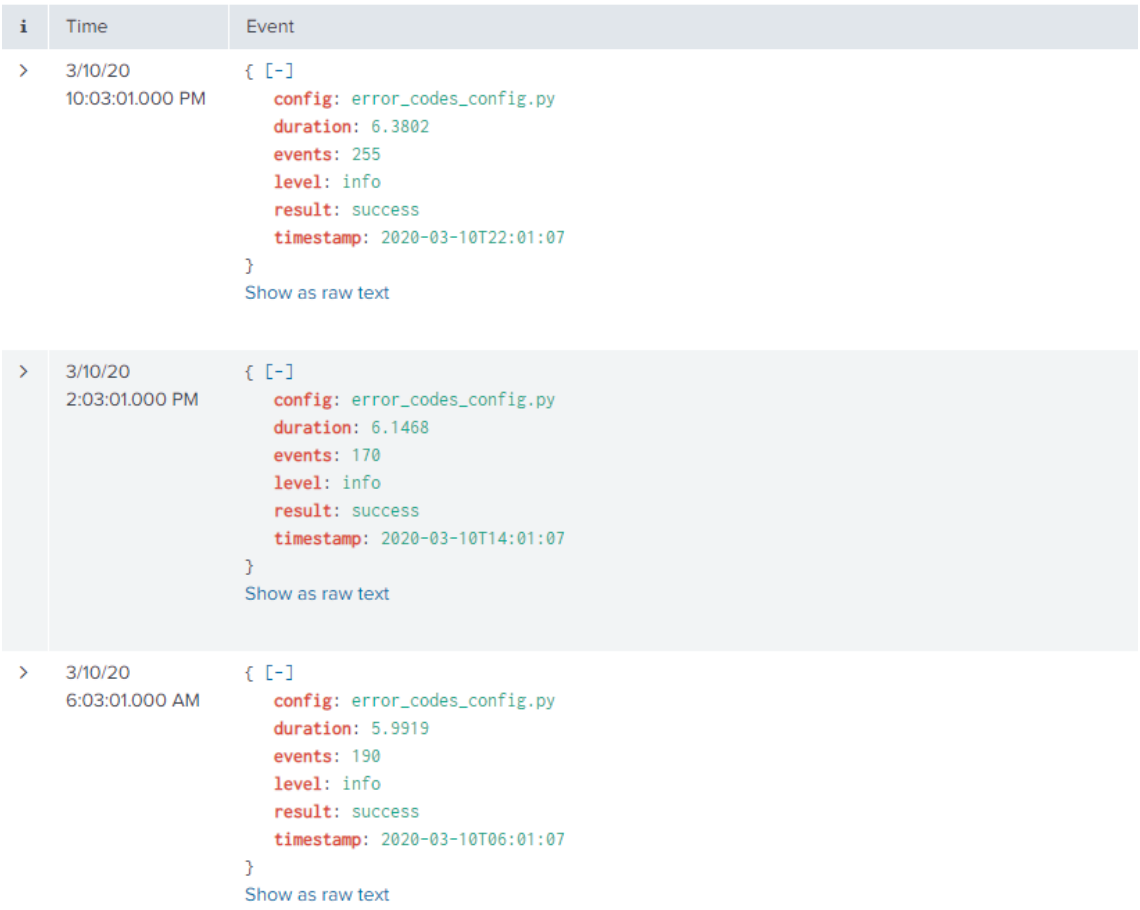
Při jakékoliv vzniklé chybě zmíněné v předešlém odstavci je do souboru vložena jedna událost. Ta má následující strukturu: čas vzniku eventu, level eventu (error), jméno skriptu (ve kterém událost nastala), jméno použitého konfiguračního skriptu a samotná výjimka.

```
2020-01-10T13:14:57 error error_codes_main.py test_config.py HTTP 400
  Bad Request --   Unknown search command 'seach'.
2020-01-10T13:34:40 error test_config.py 0.3850 FATAL: Error in 'SearchProcessor':
  Option 'field' should not be specified more than once.
```

Zdrojový kód 7: Ukázka log souboru při vzniklé chybě

Při úspěšném uložení dat

Při úspěšném uložení dat se do souboru zapíše událost ve formátu: čas vzniku eventu, level eventu (info), použitý konfigurační skript, doba trvání stahování a ukládání dat, případné diagnostické zprávy ze Splunku a nakonec počet získaných eventů ze Splunku.



Obrázek 4.4: Ukázka úspěšných eventů odeslaných do Splunku

Zasílání logů do Splunku

Aby se nemusel nastavovat mechanismus alertů na serveru při vzniklé kritické chybě, je vždy nejnovější event v log souboru zaslán do Splunku pomocí curl v bash skriptu. To se děje pomocí HTTP Event Collectoru (HEC), díky kterému je možné zasílat

data do Splunku přes HTTP nebo HTTPS protokol. HEC používá ověřování pomocí tokenu, který se generuje pro daný index. V tokenu jsou uloženy informace, které slouží jak k autentizaci, tak i k směrování a ukládání dat (na požadovaný index).

```
#!/bin/bash
#Bash script for getting last line from log file and sending it to splunk
tag=$( tail -n 1 lastevent )
curl -H "Authorization: Splunk token"
  "https://mysplunkserver.example.com:8088/services/collector/event" -d
  '{"sourcetype": "_json", "event": '"$tag"'}'
```

Zdrojový kód 8: Odesílání eventu z log souboru

Místo Splunk token je konkrétní token vygenerovaný pro index, do kterého se data budou posílat a indexovat. Pro tyto projekty získávání dat ze Splunku byl založen nový index. Jde o to, že aplikací pro získávání dat bude v budoucnu více, a tedy je dobré mít je pohromadě v jednom indexu pro kontrolu a monitoring funkčnosti. Sourcetype je pole, které určuje strukturu dat. Tou může být například json, csv a další, které Splunk zná a hned je umí naparsovat do polí. Sourcetype může být také klidně i název, který neodpovídá přímo struktuře dat, jako například cisco_log. Pokud ale struktura dat bude odpovídat json formátu, Splunk jej dokáže správně naparsovat.

4.3.2 Testování kódu

Vzhledem k této aplikaci není testování až tak rozsáhlé. Ve skutečnosti se jedná pouze o tři testovatelné metody.

První metoda, kterou lze testovat, je metoda pro vytvoření session ve splunk_auth skriptu. Ta očekává na vstupu konfigurační soubor (objekt modulu importlib), ze kterého si načte konfiguraci potřebnou pro vytvoření session. Tato metoda vrací úspěšně vytvořenou session, pokud při vytváření session nedošlo k výjimce.

V následujících dvou ukázkách kódu jsou testovány případy správného typu výstupních dat. Tedy zda je výstup správné instance a zda nejsou data typu None.

```
# Returns True if session is created successfully
def test_auth_instance(self):
  instance = create_instance(conf)
  self.assertIsInstance(instance, client.Service)

# Returns True if data are not None
def test_auth_none(self):
  instance = create_instance(conf)
  self.assertIsNotNone(instance, msg=None)
```

Zdrojový kód 9: Testování metody pro vytvoření session

Druhou metodou k otestování je metoda pro získání dat pomocí REST API. Na výstupu se očekává objekt typu ResultsReader, který by neměl být prázdný.

```
# Returns True if data are downloaded successfully
def test_results_instance_of_reader(self):
  instance = create_instance(conf)
  res = get_search_results(instance, conf)
  self.assertIsInstance(res, results.ResultsReader)

# Returns True if data are not None
def test_results_not_empty(self):
  instance = create_instance(conf)
  res = get_search_results(instance, conf)
  self.assertIsNotNone(results, msg=None)
```

Zdrojový kód 10: Testování metody pro získání správného typu dat

Poslední metoda k otestování je metoda, která již ukládá data do výstupního souboru. Zde jsou ukázány testy, které kontrolují, zda byl výstupní soubor vytvořen a zároveň jestli není prázdný.

```
def test_output_file_exists(self):
  self.assertTrue(path.exists('test.csv'))

def test_output_file_empty(self):
  self.assertNotEqual((stat("test.csv").st_size), 0)
```

Zdrojový kód 11: Testování metody pro pro vytvoření výstupního souboru

4.4 Kontrola datového toku a výsledná analýza dat

Pro kontrolu a testování datového toku byly vytvořeny reporty v Power BI. Tyto reporty jsou vizualizačně identické s reporty ve Splunku.

Jak již bylo zmíněno, data jsou stahována ze Splunku vždy na konci směny. To znamená, že data se mohla zkontrolovat kdykoliv v průběhu dne, jelikož ve Splunku i v Power BI lze filtrovat data dle časového rozpětí (tedy v době jedné směny). Jenže automatická aktualizace dat z data lake do Power BI musela být otestována pár minut po konci směny. Tedy ve skutečnosti jediný přijatelný čas byl na konci ranní směny, po 14:00.

Testování bylo zahájeno v průběhu února. Ještě předtím, než byla data zasílána z data lake do Power BI, probíhaly testy čistě pomocí generování csv souboru a prohledávání těchto dat ručně. Z počátku se vše zdálo ještě lepší, než se očekávalo, co se týče rychlosti získávání dat ze Splunku. To hlavně z toho důvodu, že search job je ve skutečnosti složen ze dvou search jobů, a ty jsou spojeny příkazem append. V obou search jobech probíhá větší počet operací nad daty (parsovaní pomocí regexů, mazání duplicitních hodnot, přejmenovávání polí apod.). Kvůli těmto operacím (a hlavně kvůli příkazu append) search job trvá ve Splunku i desítky vteřin. Kdežto při získávání dat pomocí tohoto search jobu přes REST API byla doba trvání i s uložením dat do souboru maximálně 2 vteřiny, což bylo ve skutečnosti dost pozoruhodné a bylo to bráno jako úspěch. Ovšem když byla první data zaslána do data lake a následně do Power BI, byla zjištěna chyba. Bylo získáváno více dat, než by ve skutečnosti mělo být. Po několika hodinách debugování, testování aplikace a procházení logu byla zjištěna příčina. Jednalo se o chybu v konfiguračním skriptu, konkrétně v samotném search jobu. Chyba byla v místě, kde se nastavuje cesta k log souboru pro získávání dat do Splunku. Jedná se o cestu k souboru v operačním systému Windows, a tedy jsou použita zpětná lomítka. Ve Splunku je jedním zpětným lomítkem určeno escapování, tedy byla použita dvě. Jenže v Pythonu je opět zpětné lomítko použito pro escapování, a tedy došlo k tomu, že cesta nebyla korektní. Po opravení této chyby již byla data správná. Nakonec tedy doba

trvání získávání dat i s uložením trvá přibližně 6 vteřin.

Následující dvě vizualizace dat znázorňují funkčnost datového toku a zároveň umožňují získat první náhled na data.

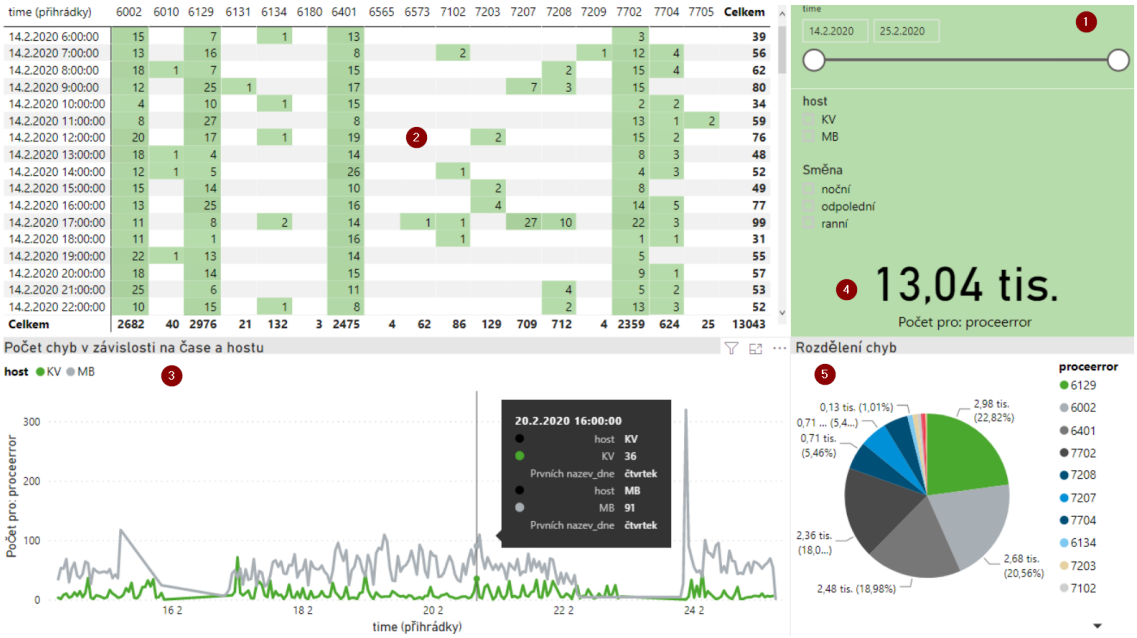
Pod oběma obrázky je v popisku uvedeno, že data byla upravena. To je z důvodu citlivosti dat, avšak tato úprava nijak vážně nezkresluje význam ani smysl vizualizace. Zjednodušeně řečeno, některé záznamy byly odebrány a jiné přidány.

V první vizualizaci jsou možnosti filtrování dat dle času, hosta a směny (sekce číslo 1). Samotná směna není součásti dat, ale díky znalosti začátku a konce směn lze přidat odpovídající hodnoty do nového sloupce.

```
1 Směna = each if [Time] >= #time(22,0,0) then "noční" else if
2   [Time] >= #time(14,0,0) then "odpolední" else if
3   [Time] >= #time(6,0,0) then "ranní") else if
4   [Time] < #time(6,0,0) then "noční" else "?"
```

Zdrojový kód 12: Přidání sloupce směny v Power BI

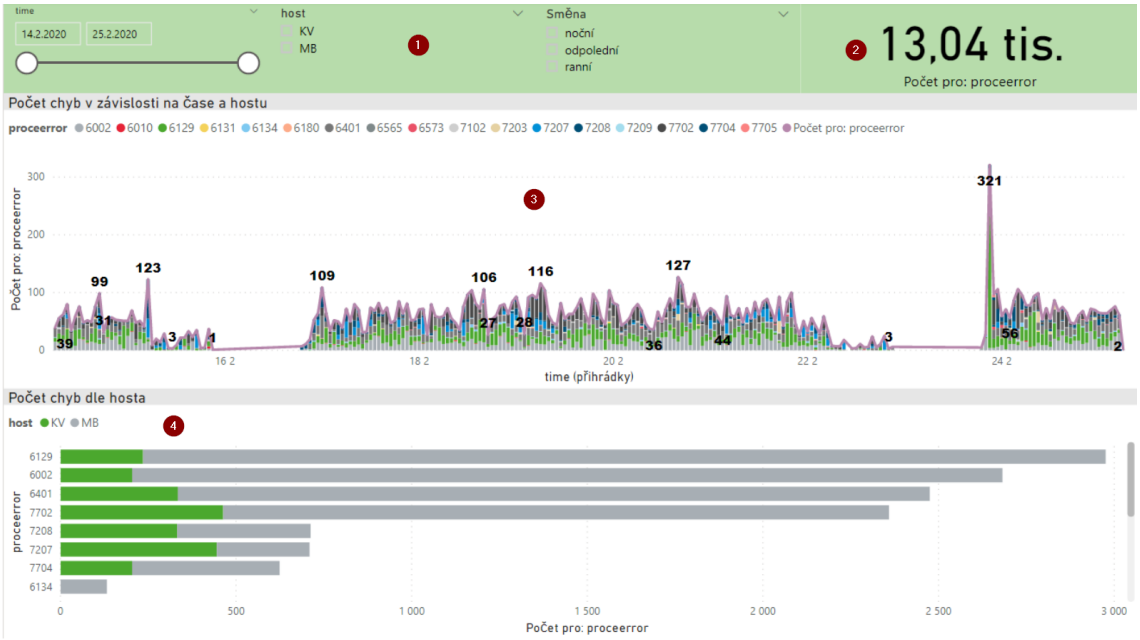
V této vizualizaci jsou dále čtyři objekty reprezentující data. V první sekci je možné filtrovat data. Sekce číslo dva reprezentuje matici počtu chyb. Řádky jsou počty chyb v daném čase a sloupce jsou samotná čísla chyb. Ve třetí sekci je graf, který znázorňuje průběh chyb v čase pro daného hosta. Sekce číslo 4 udává celkový počet chyb a sekce číslo 5 znázorňuje rozložení chyb. Je možné si povšimnout černého obdélníku, ve kterém se nachází bližší informace pro vybraný datový úsek. Zde je opět navíc pole název dne, které také nebylo součástí dat, ale bylo stejně jako sloupec pro směnu přidáno do datové sady v Power BI.



Obrázek 4.5: Ukázková vizualizace 1. Data byla upravena.

V následující vizualizaci je možné nalézt v první sekci opět možnost filtrování dat. Následně se zde nacházejí další tři sekce, které data vizualizují.

Druhá sekce, stejně jako v předešlé vizualizaci, reprezentuje celkový počet chyb. Třetí sekce znázorňuje průběh chyb v čase. Nakonec čtvrtá sekce znázorňuje počet chyb dle hosta.



Obrázek 4.6: Ukázková vizualizace 2. Data byla upravena.

5 Závěr

Cílem práce bylo seznámit se s problematikou zpracování velkých dat a nástroji pro práci s nimi, přičemž se jednalo zejména o transformaci, manipulaci a přenos těchto dat mezi systémy Splunk, Cloudera Hadoop a Power BI. Následně byla provedena analýza těchto systémů. Na základě provedené analýzy byly dále vybrány nejlepší možné technologie pro přenos dat a byla navržena aplikace, která by tyto technologie využívala. Dále byl tento návrh realizován, aplikace byla implementována v jazyce Python, čímž byl celý proces získávání dat ze Splunku do data lake zautomatizován a bylo tak zabráněno možným chybám, které vznikají při ručním exportování csv souboru. Tyto chyby jsou popsány v kapitole 3.1. Nakonec byly vytvořeny reporty v Power BI, které ověřily správnost datového toku a zároveň slouží jako ukázková vizualizace pro tento datový zdroj.

V rámci analýzy již zmíněných systémů bylo představeno několik možností, jak data přenášet mezi těmito systémy. Kompletní analýza spolu s výběrem použitých technologií je popsána v kapitole 3.2. Pro přenos dat ze Splunku do Cloudery Hadoop bylo nakonec použito REST API, kterým Splunk disponuje. V návaznosti na to byla vytvořena univerzální aplikace v jazyce Python na linuxovém serveru. Aplikace je navržena tak, aby se pro libovolný zdroj dat ze Splunku měnil pouze jeden konfigurační skript. V konfiguračním skriptu se bude pro jiný zdroj dat měnit samotný search job, získávané parametry ze search jobu, jméno výstupního souboru, interval pro získání dat a popřípadě i jiný log soubor. Všechny ostatní skripty zůstanou beze změny. Automatizované spouštění skriptu je nastaveno pomocí cron jobu.

Pro zpětnou kontrolu funkčnosti a případné alerty je vždy poslední event z log souboru odesílán zpět do Splunku pomocí bash scriptu. Pří úspěšném získání dat

se zaloguje event s časem trvání od zahájení requestu pro získání dat až po uložení dat do souboru, se jménem použitého konfiguračního souboru a počtem získaných eventů ze Splunku. Při vzniklé chybě se zaloguje jméno použitého konfiguračního souboru, jméno skriptu, ve kterém chyba vznikla, a samotná informace o chybě.

V již zmíněné analýze byl také vybrán způsob zasílání vytvořeného csv souboru na server Cloudery Hadoop pomocí UC4 jobu.

Dále byl vytvořen datový model v prostředí Power Designer, který reprezentuje samotnou datovou strukturu v Cloudera Hadoop spolu s metadaty, která slouží pro popis dat. Pomocí tohoto modelu byla vytvořena datová struktura v Cloudera Hadoop.

V poslední praktické části byla vytvořena vizualizace dat v Power BI pro kontrolu datového toku a ukázkovou vizualizaci. Ta se skládá celkem ze dvou reportů. Díky této analýze se zjistila chyba, která spočívala v tom, že bylo získáváno příliš mnoho chybových stavů jednoho druhu. To bylo způsobeno chybným formátem search jobu v konfiguračním skriptu. Konkrétně se jednalo o chybné escapování zpětných lomítek v nastavené cestě vstupního log souboru pro získání dat v SPL. Jak bylo zmíněno v kapitole 4.4, search job se skládá ve skutečnosti ze dvou search jobů spojených pomocí příkazu append a jeden z nich kvůli této chybě nebyl zpracován. Po této opravě byla data korektní, a tedy dne 9. 3. 2020 bylo celé řešení spuštěno do produkce.

Zároveň byla vytvořena interní dokumentace celého tohoto procesu získávání dat, která bude sloužit k dalším použitím pro další datové zdroje.

Toto řešení odstraňuje všechnu možnou chybovost, která vznikala při ručním exportování dat ze Splunku, což znamená zadání špatného názvu souboru a jeho uložení na nesprávné místo. S tím souvisí i čas strávený opravou těchto chyb a čas samotného exportování dat souboru. Díky tomuto řešení již není potřeba žádný lidský zásah pro export dat. Pouze se nastaví připojení k data lake, a tím jsou data automaticky načítána do Power BI. Zároveň díky univerzalitě je možné toto řešení převést na jiný zdroj dat pouze pomocí vytvoření nového konfiguračního souboru a nastavení cron jobů pro spouštění skriptů.

Literatura

[1] Sheng, Jie & Wang, Xiaojun. (2017). A Multidisciplinary Perspective of Big Data in Management Research. International Journal of Production Economics. 191. 10.1016/j.ijpe.2017.06.006.

[2] M.D. Assunca, R. N. Calheiros, S. Bianchi, M. A. S. Netto, and R. Buyya, "Big Data Computing and Clouds: Challanges, Solutions, and Future Directions,"arXiv, vol. 1, no. 1, pp. 1-39, Dec. 2019

[3] Doug Laney, "3D Data Management: Controlling Data Volume, Velocity, and Variety", Gartner, file No. 949. 6 February 2001, http://blogs.gartner.com/douglaney/files/2012/01/ad949-3D-Data-Management-Controllin-Data-Volume-Velocity-and-Variety.pdf

[4] PICKELL, Devin. What is Big Data? A Complete Guide [online]. In: . 22.08.2018 [cit. 2020-03-18]. Dostupné z: https://learn.g2.com/big-data

[5] O'REILLY, Tim, What is Web 2.0: Design Patterns and Business Models for the Next Generation of Software[online].2005, 5[cit. 2019-10-10]. Dostupné z: https://oreilly.com/pub/a/web2/archive/what-is-web-20.html

[6] Luhn, H.P.: A Business Intelligence System. IBM J. Res. Dev. 2

[7] Lim, E., Chen, H., Chen, G.: Business intelligence and analytics: research directions. ACM Trans. Manage. Inf. Syst., vol. 3, no. 4, Article 17, pp.1-10 (2013)

[8] What is Relational Database?. Oracle.com[online]. [cit. 2019-10-17]. Dostupné z: https://www.oracle.com/database/what-is-relational-database

[9] Dave, Meenu. (2012). SQL and NoSQL Databases. International Journal of Advanced Research in Computer Science and Software Engineering.

[10] Developer Survey Results 2019. Stackoverflow[online].[cit. 2019-10-11]. Dostupné z: https://insights.stackoverflow.com/survey/2019#technology-_-databases

[11] NAYAK, A., Poriya, A., Poojary, D. (2013). Type of NOSQL databases and its comparison with relational databases. International Journal of Applied Information Systems, 5(4), 16-19.

[12] WEBBER, Jim. A programmatic introduction to Neo4j. In: Proceedings of the 3rd annual conference on Systems, programming and applications: software for humanity - SPLASH '12[online]. New York, New York, USA: ACM Press, 2012, 2012, s. 2017-[cit. 2019-10-17]. DOI: 10.115/2384716.2384777. Dostupné z: http://dl.acm.org/citation.cfm?doid=2384716.2384777

[13] DAVOUDIAN, Ali, Liu CHEN a Mengchi LIU. A Survey on NoSQL Stores. ACM Computing Survey[online]. 2018, 51(2), 1-43[cit. 2019-10-17]. DOI: 10.1145/3158661. ISSN 03600300. Dostupné z: http://dl.acm.org/citation.cfm?doid=3186333.3158661

[14] P. Russom, "Big Data Analytics,"The Data Warehousing Institute, vol. 4, no. 1, pp. 1-36, 2011.

[15] Splunk Apps and Add-Ons: Enhance and Extend the Value of Splunk [online]. [cit. 2019-12-29]. Dostupné z: https://www.splunk.com/en_us/products/apps-and-add-ons.html

[16] Apache Hadoop Ecosystem [online]. [cit. 2020-03-25]. Dostupné z: https://www.cloudera.com/products/open-source/apache-hadoop.html

[17] CDH Components [online]. [cit. 2019-12-29]. Dostupné z: https://www.cloudera.com/products/open-source/apache-hadoop/key-cdh-components.html

[18] About developing apps and add-ons for Splunk Enterprise [online]. [cit. 2020-03-22]. Dostupné z: https://dev.splunk.com/enterprise/docs/welcome#Splunk-Enterprise-SDKs

A Obsah přiloženého CD

- text diplomové práce
  - diplomova_prace_2020_Lukas_Vosecky.pdf
  - zadani_diplomova_prace_2020_Lukas_Vosecky.pdf
- zdrojové kódy
  - error_codes_config.py
  - error_codes_main.py
  - splunk_auth.py
  - sendEvent.sh
  - tests.py
  - logger.py