VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ZPRACOVÁNÍ VELKÝCH DAT V OBLASTI PRŮMYSLU 4.0

BIG DATA PROCESSING IN INDUSTRY 4.0

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **DÁVID STREDÁNSKY**
AUTHOR

VEDOUCÍ PRÁCE **doc. RNDr. PAVEL SMRŽ, Ph.D.**
SUPERVISOR

BRNO 2019

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2018/2019

Zadání bakalářské práce

Student: **Stredánsky Dávid**
Program: Informační technologie
Název: **Zpracování velkých dat v oblasti Průmyslu 4.0**
Big Data Processing in Industry 4.0
Kategorie: Umělá inteligence

Zadání:

1. Seznamte se s metodami zpracování velkých dat, jejich filtrováním, vyhodnocováním, zpracováním a vizualizací.
2. S využitím existujících implementací navrhněte a realizujte systém, který bude analyzovat informace z různých čidel, dostupných v moderních průmyslových provozech, a na základě výsledků navrhovat optimalizace specifických procesů.
3. Vyhodnoťte chování systému na dodaných historických datech a předpoklady pro integraci výsledků v reálném provozu.
4. Vytvořte stručný plakát prezentující práci, její cíle a výsledky

Literatura:

- dle domluvy s vedoucím

Pro udělení zápočtu za první semestr je požadováno:

- funkční prototyp řešení

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Smrž Pavel, doc. RNDr., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 15. května 2019
Datum schválení: 1. listopadu 2018

Abstrakt

Hlavným cieľom tejto bakalárskej práce je vytvorenie aplikácie pre spracovanie priemyselných veľkých dát. Výsledná aplikácia pracuje s vibračnými dátami nameranými na ložiskách. Aplikácia je navrhnutá podľa lambda architektúry pre spracovanie veľkých dát. Umožňuje sledovanie prijímania dát zo senzorov v reálnom čase a periodické spracovanie zhlukov dát. Pri spracovaní v zhlukoch analyzuje známymi metódami v oblasti sledovania kondície ložísk, ako napríklad sledovanie efektívnej hodnoty, odchýlky alebo šikmosti. Otestované je aplikovanie metódy pre predikciu dát Prophet. Výsledná webová aplikácia je napísaná v jazyku Python s využitím frameworku Dash a výsledky zapisuje do MySQL databázy.

Abstract

Main goal of this thesis is to create application for industrial big data processing. Final application uses bearing vibration data. The application's design is inspired by Lambda architecture for big data processing. The application monitors data recieved from sensors in real time and enables periodic batch processing. Known methods from bearing condition monitoring, such as root mean square, deviation or skewness extraction are used in batch processing. Data prediction method Prophet is tested out in this thesis. Final web application is written in the Python language with the use of Dash framework and results are stored in MySQL database.

Kľúčové slová

Priemyselné veľké dáta, Priemysel 4.0, IIoT, Spracovanie dát, Dash

Keywords

Industrial big data, Industry 4.0, IIoT, Data processing, Dash

Citácia

STREDÁNSKY, Dávid. *Zpracování velkých dat v oblasti Průmyslu 4.0.* Brno, 2019. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce doc. RNDr. Pavel Smrž, Ph.D.

Zpracování velkých dat v oblasti Průmyslu 4.0

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána doc. RNDr. Pavla Smrža, Ph.D . Ďalšie informácie mi poskytla firma, s ktorou som spolupracoval. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Dávid Stredánsky
15. mája 2019

Poďakovanie

Chcel by som poďakovať vedúcemu mojej práce, doc. RNDr. Pavlovi Smržovi, Ph.D za odborné vedenie, ochotu a pomoc pri tvorbe tejto práce. Taktiež by som chcel poďakovať firme, s ktorou som spolupracoval, za poskytnutie materiálov a prostriedkov pre vznik tejto práce

Obsah

Kapitola 1

Úvod

Problematika veľkých dát je aktuálnou témou posledných rokov vo väčšine rozvíjajúcich sa oblastí. Výnimkou nie je ani oblasť priemyslu 4.0. V rámci štvrtej priemyslovej revolúcie sú veľké dáta nevyhnutnou súčasťou, potrebnou napríklad pre vznik autonómnych tovární. Priemyselné továrne generujú každý deň množstvá dát pripravených na spracovanie a využitie pre zdokonalenie výroby. Vhodné spracovanie dát nameraných na napríklad výrobných strojoch môže zabezpečiť efektívnu údržbu, ktorá obmedzí výskyt zlyhaní strojov a neplánovaných odstávok.

V rámci tejto práce prebehla spolupráca s firmou, ktorá sa pohybuje v oblasti priemyslu 4.0 a prediktívnej analýzy a údržby. Od partnerskej firmy boli poskytnuté dáta namerané na výrobných strojoch, presnejší popis poskytnutých dát sa nachádza v kapitole 4.

Cieľom tejto práce je vytvorenie aplikácie pre spracovanie, analyzovanie a monitorovanie nameraných dát na ložiskách osadených na rôznych strojoch, ktorú by mohla partnerská firma využívať. V prvej časti je rozobratá problematika industriálnych veľkých dát, možnosti ich spracovania a metódy na získanie podstatných informácií. V ďalšej časti je opísaný návrh výslednej aplikácie a následne aj jej finálna implementácia.

V kapitole 2 je opísaná štvrtá priemyselná revolúcia, dáta ktoré z nej plynú a možnosti ich využívania. V tej istej kapitole sú opísané príklady využitia priemyselných dát v praxi a softvérové existujúce riešenia pre spravovanie týchto dát. Porovnanie metód spracovania veľkých priemyselných dát a zároveň aj pohľad na možnosti analýzy vibračných dát z ložísk sa nachádza v kapitole 3. Pre účely mojej práce sú dôležité dáta, ktoré sú detailne opísané v kapitole 4. V tejto kapitole je opísaná aj partnerská firma a jej spolupráca na práci. Aplikácia, ktorá je výsledkom mojej bakalárskej práce, je navrhnutá v kapitole 5, jej finálna implementácia je prezentovaná v 6. Vyhodnotenie a opis testovania aplikácie sa nachádza v kapitole 7

Kapitola 2

Využitie priemyselných veľkých dát

Kapitola opisuje problematiku štvrtej priemyselnej revolúcie, priemyselného internetu vecí a priemyselných veľkých dát, ktoré sú generované v takomto prostredí. Ďalej opisuje možnosti využívania generovaných dát a uvedené sú aj návrhy alebo existujúce riešenia využití.

2.1 Priemysel 4.0

Aktuálna vývojová fáza, v ktorej sa nachádza svetový priemysel, sa označuje pojmom priemysel 4.0, alebo štvrtá priemyselná revolúcia. Bez narastajúcej výpočtovej sily, možností prepojenia medzi strojmi, ľuďmi a službami využívanými v priemysle, tak ako aj zaznamenávaním týchto komunikácií, by štvrtá priemyselná revolúcia neexistovala. Práve monitorovaním komunikácie medzi strojmi navzájom, a s ľuďmi generuje priemyselné veľké dáta.

Priemyselný internet vecí

Vzájomne prepojená sieť objektov, ktoré vytvárajú inteligentné prostredie, je charakterizovaná termínom internet vecí (angl. Internet of Things). V rámci priemyslu 4.0 sa takáto sieť nazýva priemyselný internet vecí a označuje sa skratkou *IIoT* (angl. Industrial Internet of Things)[22]. IIoT zahŕňa využívanie senzorov, riadiacich systémov, komunikáciu medzi strojmi, analýzu dát a bezpečnostných mechanizmov [27]. Podľa predpovedi [23] spoločnosti Gartner bude do roku 2020 takmer 20 miliárd zariadení pripojených do internetu vecí a väčšina z nich bude pochádzať práve z priemyselného sektoru.

Kyber-fyzické systémy

Významná podmnožina priemyselného internetu vecí sú takzvané kyber-fyzické systémy (angl. Cyber-physical systems, CPS). Sú to technológie, ktoré riadia systémy slúžiace na prepojenie medzi ich fyzickými prvkami a výpočtovými schopnosťami [17]. Zameriavajú sa na transformáciu surových dát na informácie, na základe ktorých je možné správne rozhodovať a riadiť procesy. Lepšie rozhodovanie o procesoch vo fabrikách vedie ku nárastu produktivity a poklesu celkových nákladov. Prepojenie fyzickej a virtuálnej vrstvy je znázornené takzvanou ”5C“ architektúrou, ktorá popisuje päť vrstiev kyber-fyzických systémov: prepojenie, konverzia, kyber-úroveň, kognitívna vrstva, konfiguračná vrstva (*Connection, Conversion, Cyber, Cognition, Configuration*)

Priemyselné veľké dáta

Jedným z produktov CPS, IIoT a ďalších sietí v oblasti moderného priemyslu je množstvo nahromadených dát. Tieto dáta sa jednotne označujú ako priemyselné veľké dáta. S veľkými dátami majú spoločné najmä veľké vygenerované množstvo (*volume*), vysokú pestrosť dát (*variety*) a rýchlosť generovania (*velocity*) [25]. Priemyselné dáta sú ale väčšinou štruktúrovanejšie, s väčšou koreláciou, usporiadanejšie v čase a vhodnejšie pre analytické výpočty [26]. Príčinou je zväčša fakt, že sú vytvárané strojmi, ktorých procesy sú viac automatizované a presnejšie ako ľudské.

Priemyselné veľké dáta, napriek tomu že sú generované strojmi, nesú negatívne charakteristiky, ktoré sú z anglického názvu definované v [26] ako "3B":

- Prostredie / Pozadie (*Background*)
  - Pri zbieraní dát v priemyselnom prostredí, narozdiel od klasických veľkých dát, nestačí numericky zaznamenávať zachytené dáta. V kontexte priemyselných dát sa často zameriava aj na fyzikálne vlastnosti a celkové pozadie vzniku dát. Je preto treba väčšiu odbornú znalosť priemyselného prostredia.
- Poškodenie (*Broken*)
  - Zhromaždené dáta pre vývoj analytického systému musia byť z viacerých kondícií a stavov zariadení. Takéto presné členenie nie je z generovaných dát bez dodatočnej analýzy jasný. Práve preto sa využíva predspracovanie, ktoré zaručí úplnosť a synchronizáciu dát.
- Nízka kvalita (*Bad-Quality*)
  - Zaznamenané dáta môžu mať často nedostatočnú kvalitu a keďže majú mať jasný fyzikálny význam, nemôžu byť uložené ako vhodné pre ďalšie spracovanie. Dôvodmi môžu byť viaceré zdroje a komunikačný šum. Na rozdiel od veľkých dát mimo priemyslu, veľké množstvo dát nepreváži to, že kvalita môže byť nižšia.

Techniky a metódy pre spracovanie a analýzu tradičných veľkých dát sú preto často nedostatočné pre priemyselné veľké dáta. Vyžaduje sa väčšia znalosť priemyselného kontextu, závislostí a fyzikálnych spojitostí.

2.2 Príklady využitia dát

Vysoká cena priemyselných zariadení vedie spoločnosti ku snahe čo najlepšie zaistiť bezpečnosť, bezporuchový beh a celkovú efektivitu zariadení. Pre tieto účely existujú viaceré systémy, vybrané z nich sú opísané v tejto podkapitole.

Monitorovanie prevádzky a chybová diagnostika

V priemyselnej výrobe sa kladú požiadavky na maximálne vyťaženie strojov a vyhnutie sa neočakávaným odstávkam fabrík, ktoré spôsobia poruchy na strojoch. Riešením bolo (niekde ešte stále je) preventívna výmena rôznych častí stroja na základe vedomostí údržbového manažéra a informácií od dodávateľov predtým než sa stihnú pokaziť [16]. Technológie priemyslu 4.0 a dáta, ktoré prináša, umožnili takéto riešenie nahradiť. Monitorovanie výrobných zariadení rôznymi senzormi zabezpečilo podklady pre detekciu a následnú diagnostiku

vzniknutých porúch. Vďaka týmto mechanizmom sa nemusia pravidelne meniť bezchybné časti z dôvodu prevencie, čo vedie k šetreniu financií. Článok [29] z roku 2017 opisuje potrebnú zmenu zamerania vývoja v odbore priemyselného procesného monitorovania z detekcie chyby práve na chybovú diagnostiku. Zistenie chyby zabezpečí okamžitú pozornosť pre hľadanie riešenia vo fabrike. Bez diagnostiky sa však nezistí príčina chyby, čo môže viesť k neefektívnej a drahej oprave stroja. Chybová diagnostika umožní lepšie ponaučenie sa z chyby, čo vo výsledku vedie k šetreniu finančných a ľudských zdrojov firiem.

Prediktívna údržba

Kým monitorovanie a chybová diagnostika riešia odpovede na otázky čo sa stalo a prečo sa to stalo, nezodpovedajú otázku čo sa môže stať v najbližšej dobe. Vyhnúť sa poruche úplne je pre výrobu ideálnejšie z finančnej aj časovej stránky. Prediktívna údržba je teda zameraná, tak ako aj chybová diagnostika, na analýzu nazbieraných dát a vyhodnotenie aktuálnej kondície, v ktorej sa stroj nachádza. Pridáva k tomu ale aj analytickú predpoveď kondície stroja v určitom časovom rozsahu.

Integrácia umelej inteligencie

Veľkou motiváciou spoločností pre zber dát je ich nevyhnutnosť v úlohe umelej inteligencie a strojového učenia. Umelá inteligencia môže byť implementovaná nad existujúcimi produktami alebo službami v priemysle a zvýšiť tak ich efektivitu, spoľahlivosť, bezpečnosť a životnosť [31].

V Priemysle 4.0 možno aplikačné oblasti umelej inteligencie rozčleniť do troch hlavných funkcií – porozumenie a automatizácia postupov a procesov, automatické rozpoznávanie vzorov ako v prípade rozpoznávania reči a tváre a na záver automatické spracovanie dát. V prostredí Priemyslu 4.0 je automatizácia rutinných úloh jedna zo zásadných aplikácií, v ktorej môžu byť inteligentné programy značným prínosom.[5]

V rozbore využívania veľkých dát v priemysle existujú ďalšie kategórie, o ktorých by sa mohlo diskutovať. V rozsahu mojej práce ich však preberať nebudem, pretože nesúvisia s výsledným produktom. Časť dát v priemysle vzniká napríklad zo sledovania komunikácie medzi ľuďmi a strojmi, na základe by sa dala sledovať a zlepšovať efektivita jednotlivých zamestnancov, či tímov ľudí.

2.3 Existujúce riešenia

Prehľad existujúcich riešení je v tejto práci rozdelený na výber spoločností v priemyselnom sektore, ktoré zakomponovali využívanie veľkých dát do svojej firemnej stratégie a na výber softvérových riešení venujúcim sa veľkým dátam, aplikovateľným pre priemysel.

2.3.1 Priemyselné spoločnosti využívajúce veľké dáta

V priemyselnej oblasti sa v dnešnej dobe väčšina spoločností zaoberá využívaním veľkých dát, automatizácie alebo umelej inteligencie. Presný popis riešení, ktoré sú reálne implementované a zdokumentované, často podlieha utajeniu a preto tieto informácie nie sú väčšinou verejne dostupné. Vybrané sú dve spoločnosti z automobilového priemyslu a ich prístup k veľkým dátam.

Audi

Audi začala so zberom veľkých dát v ich fabrikách v roku 2015. Dáta ukladá do distribuovaného súborového systému *Hadoop*. Pri výrobe jedného auta sa zaznamená viac ako 25000 signálových údajov, ktoré sú neskôr analyzované. Pre spracovanie zhlukov a aj prúdov dát Audi využíva platformu *Apache Kafka*. Zber veľkých dát spoločnosť využíva pre zefektívnenie výroby, prediktívnu údržbu strojov a celkový prehľad o výrobe, ktorý poskytuje priestor dátovým analytikom snažiacim sa o neustálu inováciu spoločnosti a výroby [15].

Škoda Auto

V **Škoda Auto** zaznel pojem Priemysel 4.0 v roku 2015. V rovnakom roku ako sa tento pojem predstavil prvýkrát v Českej republike na Medzinárodnom strojárenskom veľtrhu. Hneď ako sa tento pojem objavil, popredný strojársensky podnik začal prejavovať záujem dozvedieť sa o Priemysle 4.0 viac. Prvý krok bol začať zbierať potrebné údaje z rôznych typov procesov a postupov, ktoré sa pri výrobe automobilov odohrávajú. Po nazbieraní dostatočne veľkému množstva dát sa začala práca na nástrojoch, ktoré to dokážu transformovať na efektívne a kľúčové informácie [20].

Na tieto dáta sa aplikujú metódy strojového učenia. Tie sa využívajú napríklad na monitorovanie a analyzovanie postupu výrobku počas výroby. V *Škoda Auto* je snaha vytvoriť automatické diagnostické aplikácie. Tie budú môcť upozorňovať na možnú poruchu ako aj na správu a sledovanie diania na linke v reálnom čase.

2.3.2 Softvérové riešenia pre priemyselné dáta

Pre komerčnú časť softvérových riešení je prezentovaná spoločnosť *Striim* a z hľadiska vývoja aplikácií je predstavená *The Apache Software Foundation*, ktorá ponúka viaceré prostredia pre prácu s veľkými dátami, využiteľnými aj v priemyselnom prostredí.

Striim

Striim analyzuje a zobrazuje údaje z firiem a IoT senzorov v reálnom čase. Ponúka jednotnú, end-to-end softvérovú platformu, ktorá vykonáva prediktívnu analýzu dát v reálnom čase a je schopná integrácie so strojovým učením, vhodnou pre zisťovanie časovo citlivých údajov z výrobných hál [3]. *Striim* je spoločnosť na svetovej úrovni, ktorej jedným z hlavných produktov je multifunkčná platforma pre spracovávanie dát. Pre verejnosť je dostupná iba obmedzená dočasná demo verzia platformy.

Apache Spark

Opensource **Apache Spark** je viacúčelovo zameraný framework pre distribuované analytické spracovanie veľkých dát v zoskupeniach výpočtových zariadení (*clusters*). Poskytuje prostredie pre programovanie aplikácií, ktoré pracujú s dátami uloženými v rámci distribuovaného súborového systému (najčastejšie *Hadoop Distributed File System (HDFS)*). Aplikácie vyvinuté v prostredí Apache Spark sú implicitne vhodné pre paralelné spracovanie dát.

Spark obsahuje rozšírenie **Spark Streaming**, ktoré umožňuje škálovateľné, vysoko výkonné a odolné voči chybám spracovanie prúdov dát v reálnom čase. Dáta môžu byť prijímané z rôznych zdrojov a môžu byť spracované pomocou komplexných algoritmov ako napríklad mapovanie, agregácia alebo spájanie. Spracované dáta môžu byť vkladané ďalej

do súborových systémov (*HDFS*) alebo databáz [2]. Prehľadná schéma Spark Streaming funkčnosti je zobrazená na obrázku 2.1.

Obr. 2.1: Apache Spark Streaming schéma možných vstupov a výstupov [2].

Apache Hadoop

Apache Hadoop predstavuje zbierku softvérových riešení pre distribuované systémy, veľké dáta a výpočty nad nimi. Apache Hadoop je zároveň framework, ktorý umožňuje distribuované spracovanie veľkých objemov dát, pri čom využíva rozdelenie výpočtovej sily a úložiska pre súbory medzi zariadenia (*cluster*) v zoskupeniach.

Hadoop Data File System (HDFS) je distribuovaný súborový systém navrhnutý primárne na hardvér. Je súčasťou balíčka firmy *The Apache Software Foundation*, teda je kompatibilný a preferovaný súborový systém pre aplikácie vyvíjané niektorým z *Apache* prostredí a nástrojov [21].

Kapitola 3

Metódy spracovania dát

Prístup k spracovaniu priemyselných veľkých dát sa dá rozdeliť do dvoch kategórií. Prvá sa zameriava na čas spracovania. Zahŕňa spracovanie dát v reálnom čase (*real-time*) a spracovanie v zhlukoch (*batch processing*). V druhej kategórií sú prístupy ku spracovaniu dát vzhľadom na rôzne analytické a fyzikálne prístupy, ktoré berú ohľad na to, že sa zaoberáme priemyselnými dátami, narozdiel od prvej kategórie, ktorá je aplikovateľná aj na klasické veľké dáta.

3.1 Spracovanie v zhlukoch a spracovanie v reálnom čase

Spracovanie v zhlukoch

Spracovanie v zhlukoch je efektívne pri veľkom množstve dát, ktoré vzniklo zaznamenávaním rôznych transakcií za určité dlhšie časové obdobie. Dáta sú zozbierané, vložené do systému, spracované a po analýze vzniknú výsledky z nazbieraných dát. Takéto spracovanie na zhromaždených dátach je vhodné, keď [30]:

- funkčnosti systému neprekáža jednorázové vyťaženie pre účely výpočtov.
- upozornenia na základe výsledkov výpočtov nie sú potrebné v reálnom čase.
- spracovanie dát zahŕňa agregačné funkcie ako napríklad zisťovanie priemeru, maxima alebo minima.

Prvé dva body nie sú väčšinou prípadom v dennom chode priemyselných spoločností a fabrík. Výsledky zo senzorov na zariadeniach, ktoré napríklad zisťujú kondíciu zariadenia a avizujú prípadnú poruchu sú potrebné v reálnom alebo takmer reálnom čase. Pri spracovaní vibračných dát sa však využíva agregácia dát za účelom získať z dát dôležité informácie.

Samostatné programy sú potrebné pre vstup, výstup a spracovanie dát. Výpočty trvajú rádovo niekoľko hodín a rátajú sa väčšinou v časoch, kedy systém zvyčajne verejnosť nevyužíva (v noci, nad ránom). Bežným príkladom môže byť spracovanie údajov o transakciách zákazníkov veľkej finančnej firmy na konci mesiaca a na základe výsledkov prehodnotenie stratégie do nasledujúceho obdobia a prispôsobenie sa potrieb zákazníkom.

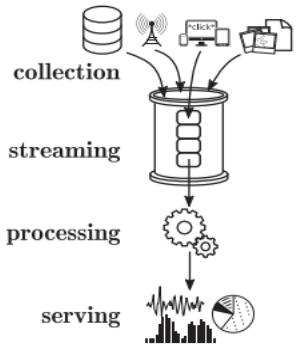
Pre priemyselných dátach, ktoré sú zamerané na zlepšenie denného behu prevádzky a minimalizáciu škôd a nákladov, je spracovanie veľkých zhlukov dát neefektívne riešenie, aj keď napríklad forma agregácie sa do funkčného spracovania priemyselných dát zakomponovať musí.

Spracovanie v reálnom čase

Na rozdiel od tradičného spracovávania v systémoch, ktoré periodicky spracovávajú veľké množstvá dát, spracovanie v reálnom čase sa vyhýba odkladaniu dát a spracuje ich ihneď po sprístupnení, čím sa minimalizuje čas, ktorý strávi nameraná jednotka dát v reťazovom procese spracovania [33]. Takéto spracovanie sa označuje aj ako prúdové spracovanie (z angl. *streaming analytics*). Značí, že generované dáta musia prejsť pred zobrazením výsledkov užívateľom reťazovým spracovaním. Spracovanie v reálnom čase je vhodné keď:

- odozva na výsledky analýz dát má byť minimálna, napríklad pre potreby okamžitých bezpečnostných upozornení
- sa pracuje s časovými radami a zisťovaním časových vzorov
- majú dáta príliš veľký objem a nie je možné ich ukladať
- sa využíva v prostredí, kde sa javí ako prirodzené. Nárazovo spracovávať vibrácie z priemyselných zariadení raz mesačne nie je najvhodnejšie riešenie predchádzaniu porúch.

Typické úrovne reťazového spracovania popisuje obrázok 3.1, kde je ilustrovaný prechod od zberu dát (senzory), cez kontinuálny presun a spracovanie až po zobrazenie užívateľom.

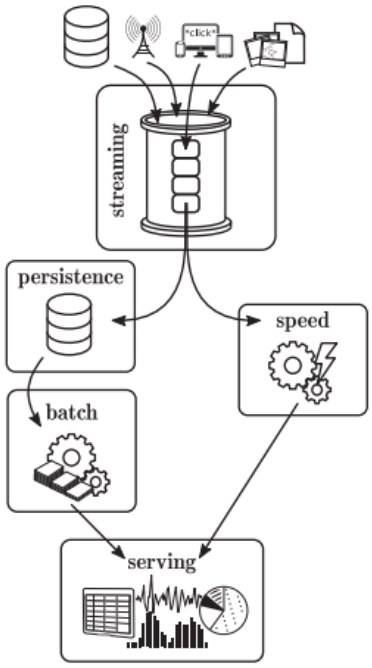


Obr. 3.1: Abstraktné zobrazenie reťazového spracovania [33].

Pre priemyselné veľké dáta je teda priaznivejšia architektúra, ktorá sa blíži spracovaniu v reálnom čase. Potreba využívania, sledovania a spracovávania priemyselných dát v reálnom čase bola zdôraznená aj v práci [32].

Efektívne softvérové riešenie spracovávania veľkých dát by malo spojiť vhodné elementy zo spracovávania v zhlukoch, ako aj v reálnom čase. Výsledkom takejto snahy je systémový návrh s názvom **lambda architektúra**. Lambda architektúra narástla na popularite, pretože spája pomalé dávkovo-orientované spracovanie s komponentom spracujúcim dáta v reálnom čase [33]. Rieši tým problém veľkosti aj rýchlosti generovania dát. Tri úrovne lambda architektúry sú zobrazené v obrázku 3.2: Dáta sú uložené v trvalom úložisku (*persistance*), z ktorého sú spracované úrovňou na spracovanie zhlukov (*batch*) v pravidelných intervaloch, zatiaľ čo rýchlostná vrstva (*speed*) spracováva surové dáta, ktoré neboli vložené na trvalé úložisko. Obslužná vrstva (*serving*) zobrazuje výstupy oboch predošlých vrstiev.

Výhody nadobudnuté spojením spracovania v reálnom čase a v zhlukoch sú zúčtované vo väčšej zložitosti pri vývoji softvéru, nasadení do prevádzky a údržbe.



Obr. 3.2: Lambda architektúra pre spracovanie veľkých dát [33].

Jedným z predstaviteľov využívania lambda architektúry je aj *Apache Spark* (podkapitola 2.3.2), ktorý umožňuje implementáciu paralelného prúdového spracovania a spracovania v zhlukoch.

Existujú aj ďalšie architektúry a prístupy ku riešeniu problémov spojených so spracovaním v reálnom čase oproti spracovaniu v zhlukoch, ktorým som sa ale v bakalárskej práci nevenoval.

3.2 Metódy spracovania vibračných dát

Keďže priemyselné veľké dáta spočívajú z veľkej časti z meraní fyzikálnych veličín, je vhodné rozobrať metódy, ktoré sa využívajú. Vzhľadom na to že v rámci tejto práce sa pracuje s meraním vibrácií na rôznych zariadeniach, rozobraté sú práve metódy spracovania vibračných dát.

Pri sledovaní ložísk sa využíva sledovanie vlastností a prvkov, ktoré sa v čase menia. Štatistické funkcie v časovej doméne ako priemer, efektívna hodnota (*angl. Root Mean Square, RMS*), štandardná odchýlka a výkyv boli v minulosti primárne využívané na rozdelenie dvoch vibračných signálov. Pokročilejšie sledovanie vlastností ako šikmosť alebo špicatosť môžu byť využité pri signály, ktorý nie je stacionárny [18]. Zmeny týchto vlastností v čase často indikujú zhoršovanie stavu ložiska. Prehľad sledovaných vlastností využívaných v sledovaní stavu ložísk sa nachádza v tabuľke 3.1

Metódami samotného merania a zberu dát sa v tejto práci nezaoberám, prezentované sú ale metódy využívané pri spracovaní. Zhromaždené dáta, nad ktorými je možné aplikovať menované metódy sú detailne opísané v kapitole 4.

Názov vlastnosti	Krátka definícia	Vzorec
Efektívna hodnota	Efektívna hodnota sa postupne zvyšuje so vznikajúcou chybou ložiska. Nieje však schopná poskytnúť informácie pri samotnom začiatku poruchy, ale až pri stálom zhoršovaní stavu.	$\mathrm{EH} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} \mathrm{x}_i^2}$
Odchýlka	Odchýlka meria rozptyl signálu okolo referenčnej strednej hodnoty.	$\mathrm{Odch} = \frac{\sum_{i=1}^{N}(\mathrm{x}_i - \mathrm{m})^2}{(\mathrm{N}-1)\sigma^2}$
Šikmosť	Šikmosť kvantifikuje asymetrické správanie vibračného signálu cez jeho funkciu hustoty pravdepodobnosti (HP).	$\mathrm{Sk} = \frac{\sum_{i=1}^{N}(\mathrm{x}_i - \mathrm{m})^3}{(\mathrm{N}-1)\sigma^3}$
Špicatosť	Špicatosť kvantifikuje maximálnu hodnotu hustoty pravdepodobnosti. Pre valivé ložiská je hodnota špicatosti pre normálový stav uznaný ako hodnota 3.	$\mathrm{Sp} = \frac{\sum_{i=1}^{N}(\mathrm{x}_i - \mathrm{m})^4}{(\mathrm{N}-1)\sigma^4}$

Tabuľka 3.1: Krátky prehľad štatistických vlastností v čase aplikovaných v kondičnom monitorovaní typických valivých ložísk [18].

Kapitola 4

Zhromaždené dáta

V kapitole sú opísané dáta využité na prezentáciu funkčnosti aplikácie navrhnutej v kapitole 5, ich štruktúra a spôsob akým boli namerané. Ďalej je opísaná alternatíva k daným dátam, a to verejné datasety z voľne dostupných zdrojov.

4.1 Dáta od partnerskej firmy

Firma, s ktorou som spolupracoval na mojej práci, sa zameriava na prediktívnu údržbu tvarovacích a obrábacích strojov. Sleduje ložiská, alebo iné časti (upínací nástroj, spojka, rám) strojov, nad nameranými dátami prevádza vlastné analýzy a vyhodnocuje aktuálnu kondíciu pozorovanej časti. Taktiež na základe rastúcich negatívnych trendov kondície predpovedá výskyt poruchy. Na riziko následne upozorní klientov.

Pre potreby mojej práce sme po konzultácií zvolili využiť namerané surové dáta zo štyroch strojov, na ktorých sa sledujú kondície ložísk. Na vybraných meraniach v rozsahu rádovo niekoľko týždňov až mesiacov je viditeľné zhoršovanie kondície ložísk a následne ich výmena a znova bezporuchový priebeh využívania strojov. Taktiež využívam už analyzované, spracované a agregované dáta z tých istých meraní. Využité sú pre porovnanie s ďalšími možnosťami spracovania.

4.1.1 Prehľad poskytnutých dát

Mená zariadení sú kvôli požiadavkám firmy zredukované na písmená A až D. Jednotlivé názvy sú teda zložené z anglického prekladu slova stroj a písmena A,B,C alebo D (*Device_A..D*).

Pri každom zariadení je nameraný rôzny počet meraní v rôznych časoch. Parametre merania v rámci jedného súboru sú vzorkovacia frekvencia a celkový počet záznamov. Prehľad údajov o meraniach na zariadeniach je zobrazený v tabuľke 4.1.

Na každom stroji je umiestnených viacero akcelerometrov, ktoré merajú vibrácie na rôznych ložiskách, poprípade rôznych častiach jedného ložiska. Počet senzorov pre zariadenie sa líši a je zaznamenaný v tabuľke 4.1.

4.1.2 Štruktúra TDMS súborov

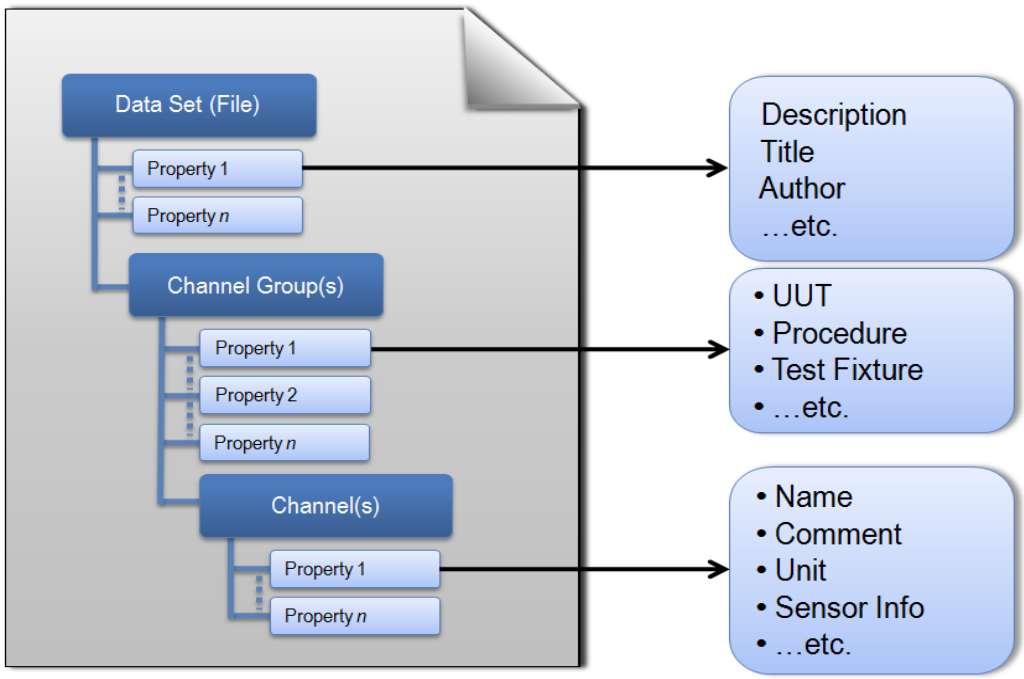
Každé meranie je uložené do jedného súboru. Merania prebiehajú na hardvéri od spoločnosti National Instruments, ktorý vyvinul vlastný formát súboru označený ako TDMS (Technical Data Management Streaming). Binárny TDMS súborový formát je ľahko kon-

Zariadenie	Device_A	Device_B	Device_C	Device_D
Dátum prvého merania	17/05/31	18/05/21	16/11/23	15/09/10
Dátum posledného merania	17/08/20	18/12/01	17/07/11	15/10/09
Vzorkovacia frekvencia [Hz]	102400	102400	51200	102400
Počet vzoriek v jednom meraní	307200	102400	51200	204800
Čas jedného merania [s]	3	1	1	2
Počet meraní (súborov)	1909	7972	7113	6887
Celková veľkosť [GB]	14	183	5.5	16
Počet osadených senzorov	6	6	4	4

Tabuľka 4.1: Parametre meraní na zariadeniach

vertibilný, inherentne štrukturovaný, schopný prenosu na vysokej rýchlosti. Formát je po spojení s technológiami spoločnosti National Instruments schopný rýchleho systému vyhľadávania, odstraňujúc tak potrebu pre náročný databázový dizajn, architektúru a údržbu [24].

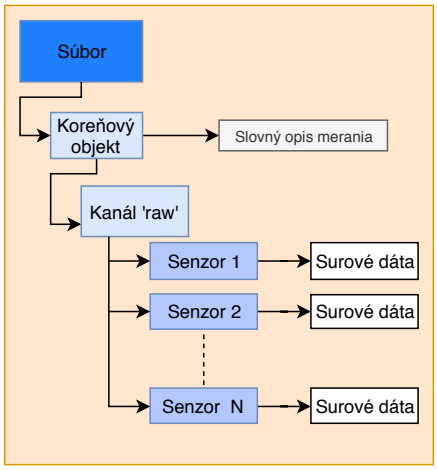
TDMS je hierarchicky organizovaný. Využíva tri hierarchické úrovne, ktoré sú zobrazené v obrázku 4.1. Obsahuje úroveň pre súbor, skupiny a kanály. Počet skupín a kanálov v rámci jedného súboru je neobmedzený. Táto štruktúra umožňuje logické usporiadanie skalárnych dát podľa spoločných vlastností, s vhodným popisom v každej úrovni hierarchie.



Obr. 4.1: Každý TDMS súbor obsahuje deskriptívne informácie na úrovni pre súbor, skupinu a kanál [24].

Jeden súbor merania obsahuje dáta z viacerých senzorov, ktoré sú osadené na stroji. V usporiadaní TDMS súboru sa to odrazí na počte kanálov v rámci jednej skupiny. Koreňový odkaz súboru odkazuje na slovný opis súboru a meraní a na jedinú skupinu označenú

raw. V tejto skupine sa nachádzajú kanály so surovými dátami, uloženými v poliach. Počet kanálov je minimálne rovnaký ako počet senzorov na zariadení. V prípade že sa do súboru zapisujú dodatočné informácie, môžu byť uložené v osobitnom kanáli. Takéto využitie TDMS štruktúry je ilustrované v obrázku 4.2.



Obr. 4.2: Hierarchická štruktúra súborov so surovými dátami od partnerskej firmy.

Presný čas merania je zaznamenaný v názve súboru, ktorý má štruktúru `raw_XX_YYMMDD_HHMMSS.tdms` (*XX* je identifikácia merania využívaná partnerskou firmou, pre účely mojej práce nepotrebná).

Kapitola 5

Návrh aplikácie pre spracovanie a monitorovanie dát

V kapitole je opísaný postup návrhu výslednej aplikácie. Na základe stanovených požiadaviek sú prezentované možnosti riešenia jednotlivých častí jednotlivé časti systému, z ktorých je odôvodnený výber výsledného riešenia.

Požiadavky na výslednú aplikáciu

Aby mala táto práca zmysel, musí výsledný produkt spĺňať určité požiadavky. Tie sú určené dvoma stranami - partnerskou firmou a doplnené sú mnou.

Pre partnerskú firmu bude práca prínosná, ak vo výsledku dostanú aplikáciu, ktorá nebude mať problém integrácie s ich systémom a dátami. Funkčnosť by mala zahŕňať:

- Spracovávanie surových dát
- Monitorovanie sledovaných zariadení v reálnom čase
- Zobrazovanie a spracovávanie historických dát
- Dôraz na možnosť integrácie s vnútorným systémom partnerskej firmy

Firma vyvíja rôzne algoritmy na spracovávanie surových dát, ktoré chce testovať a zobrazovať ich výsledky. V budúcnosti by teda mala existovať možnosť pridávať a zamieňať algoritmy, čo zabezpečí modularita systému. Požadované bolo aj preskúmanie možností a vytvorenie prostredia pre integráciu prediktívnej analýzy dát a aplikovanie umelej inteligencie v budúcnosti.

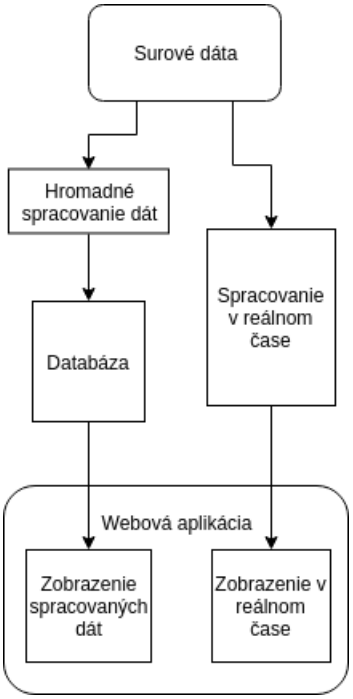
Z mojej strany som doplnil požiadavky tak, aby aplikácia vhodne prezentovala výsledky zákazníkom. Po uvážení som pristúpil ku možnosti webovej aplikácie s vhodným systémom prezentácie dát. Užívateľské rozhranie by malo byť jednoduché a intuitívne.

Schéma aplikácie

Pre výsledný systém spracovania dát som sa inšpiroval lambda architektúrou, opísanou v podkapitole 3.1 (obrázok 3.2). Lambda architektúra zabezpečuje splnenie požiadaviek duálneho prístupu k dátam - v reálnom čase aj v zhlukoch.

Schéma celkovej aplikácie je zobrazená v obrázku 5.1. K surovým dátam pristupuje modul pre hromadné spracovanie a modul pre spracovanie dát v reálnom čase. Výsledky

spracovaní dát sa ukladajú do databázy. Užívateľské rozhranie pre zobrazovanie výsledkov a ovládanie systému reprezentuje webová aplikácia.



Obr. 5.1: Schéma aplikácie.

Výber programovacieho jazyka

Pre dátovú analytiku a prácu s veľkými a industriálnymi dátami existuje viacero programovacích jazykov. Medzi populárne patria napríklad *Python*, *R*, *Java*, *Scala*, *SQL* a iné. Pre účely celkového navrhovaného systému som vyberal viacúčelový jazyk, ktorý je vhodný na analyzovanie a reprezentovanie dát.

- **Python** je jednoduchý, univerzálny, multiparadigmový programovací jazyk. Najväčšou silou Pythonu je obrovské množstvo knižníc, ktoré pomáha riešiť rôzne typy úloh, ako napríklad grafické používateľské rozhranie, automatizáciu, multimédiá, databázy, text a spracovanie obrazu [28]. V minuloročnom prieskume [6] spoločnosti Kaggle na vzorke takmer 24000 dátových analytikov sa umiestnil Python ako najpoužívanejší programovací jazyk.

- Programovací jazyk **R** je často používaný jazyk v dátovej analytike a štatistike. Je silne objektovo orientovaný, čo je jeho výhodou oproti iným výpočtovým jazykom [28]. R slúži ako voľne dostupná alternatíva k spoplatneným štatistickým softvérom ako *MATLAB* alebo *SAS* [19]. Aj keď je v obecnej problematike veľkých dát jazyk R populárny, pre špecifickejšie potreby priemyselného prostredia a priemyselných dát nie je až tak často využívaný.

- **Java** je jedným z najviac praktických jazykov, ktoré boli zatiaľ navrhnuté. Je populárny tak pre vývoj backendových systémov, ako aj desktopových aplikácií. Portabilita

a virtuálne nástroje Javy (*JVM*) sú jednými z najväčšími výhod Javy v oblasti priemyselnej informatiky [19].

Pre skoro všetky časti výsledného systému som si ako jazyk vybral **Python**. Dôvodom je hlavne univerzálnosť jazyka a veľké množstvo dostupných knižníc. Do budúcnosti je Python taktiež vhodný pre integráciu umelej inteligencie do systému, čo bola jedna s vedľajších požiadaviek partnerskej firmy. V inom jazyku je navrhnutá len databáza, ktorá je opísaná v 5.1.

Súborový systém

Pri práci s dátami a veľkými dátami je potrebné zvoliť kde a ako budú dáta uložené.

Tradičný **diskový súborový systém** reprezentuje uloženie dát na disku, ktorý je interne zabudovaný v počítači alebo externe pripojený. V kontexte veľkých dát sa ako jednotné úložisko pre dáta nepreferuje uloženie na jednom disku. Dôvod je rastúca cena diskov, ktoré sú schopné uložiť dáta veľkého objemu. Klasickými predstaviteľmi sú *NTFS* (Windows) a *ext4* (Linux) .

Distribuovaný súborový systém je založený na myšlienke rozdeľovania úložiska a výpočtovej sily medzi väčší počet zariadení. Jedným z procesov implementácie distribuovaného systému je centralizované priraďovanie prístupu a kontroly nad dátovými úložiskami, ktoré je spravované servermi. V rámci veľkých dát je takýto systém pre správu súborov preferovanou voľbou, nakoľko rozdeľujú výpočtové zaťaženie na vzájomne prepojené zariadenia. Populárne distribuované súborové systémy sú napríklad *Hadoop distributed file system (HDFS)* (Apache Software Foundation) a *Google File System (GFS)* (Google Inc.).

Dáta od partnerskej spoločnosti sú uložené na disku s využitím klasického súborového systému. Keďže veľkosť poskytnutých dát (rádovo stovky GB, presnejšie v 4.1) neprekračuje hodnoty, pri ktorých je potrebné využívať distribuovaný systém, rozhodol som sa výslednú aplikáciu prispôsobiť aktuálnemu súborovému systému.

5.1 Databáza

Databáza musí obsahovať základné informácie o sledovaných zariadeniach, ich senzoroch a grafoch, ktoré zobrazujú výsledky spracovania dát. Hlavná porcia dát sú časovo označené dáta pre grafy.

MySQL

Najpopulárnejší voľne dostupný opensource SQL systém pre správu databáz, vyvíjaný a distribuovaný spoločnosťou *Oracle Corporation* [13]. MySQL je relačný databázový systém, ktorý podporuje viacvláknové prístupy. Napísaný je v jazyku C a C++. Jeho využitie je široké a okrem klasických využití relačných databáz ponúka aj spojenie so svetom veľkých dát. Umožňuje integráciu do frameworkov pre veľké dáta a výpočty v pamäti pre využitie spracovania v reálnom čase [10].

PostgreSQL

PostgreSQL je opensource objektovo-relačný systém pre správu databáz [14]. Je vyvíjaný spoločnosťou *The PostgreSQL Global Development Group*. Využíva jazyk SQL a oproti iným

systémom pre správu databáz vyniká jednoduchšou porgramovateľnosťou aplikácií, ktoré využívajú dáta z databáze.

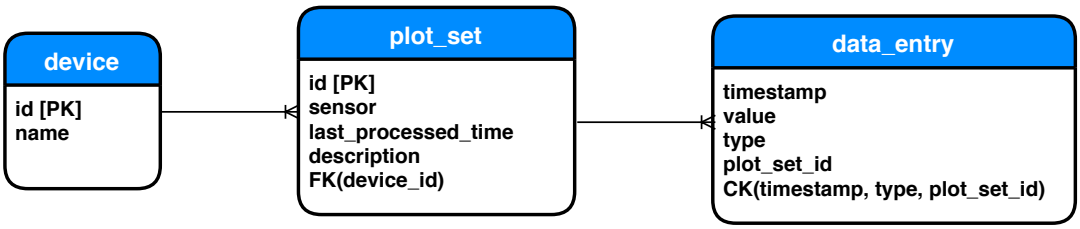
InfluxDB

InfluxDB je databáza časových radov navrhnutá tak, aby zvládla vysoké zaťaženie pri zápise a dotazoch. Je určená pre použitie v akomkoľvek prípade, kde sa zaobchádza z veľkým počtom dát, ktoré sú časovo označené. Využitie zahŕňa sledovanie vývoja a výroby produktov, aplikácií, dát zo senzorov obsiahnutých v IoT, a analytiku v reálnom čase [11].

Ako databázový systém som si vybral **MySQL** z viacerých dôvodov. Ponúka spoľahlivé a bezpečné riešenie a integráciu s nástrojmi pre veľké dáta pre budúci vývoj. Hlavným faktorom bolo, že partnerská firma už využíva MySQL vo svojom systéme. Integrácia s výslednou aplikáciou, ktorá používa rovnaký databázový systém, bude rozumnejšia a jednoduchšia. V prípade, že by nebola potrebná integrácia s existujúcou MySQL databázou, by rozumnejšou voľbou bola *InfluxDB*. Je určená práve pre časové rady, ktoré tvoria hlavnú časť údajov vo výslednej databáze.

Schéma databázy

Jednoduchá schéma, pozostávajúca z troch entít, postačí potrebám výsledného systému. Navrhnuté sú entity pre zariadenie (*device*), sadu grafov (*plot_set*) a dátový záznam v grafe (*data_entry*). Schéma je zobrazená na obrázku 5.2



Obr. 5.2: Schéma databázy.

Entita pre dátový záznam v grafe obsahuje časový záznam, hodnotu a typ spracovania dát. Viaže sa väzbou *N:1* ku entite pre sadu grafov. Primárny kľúč je zložený z kombinácie časového záznamu, identifikácie sady grafov a typu spracovania. Zložený primárny kľúč zabezpečí, že v grafe nebude jednému časovému záznamu priradených viac hodnôt.

Entita pre sadu grafov reprezentuje spoločné údaje pre grafy, ktoré sú výsledkami rôznych spôsobov spracovaní nad rovnakými dátami, ktoré pochádzajú z meraní jednotlivých senzorov na zariadeniach. Obsahuje atribút pre zaznamenanie času posledného výpočtu nad dátami, ďalej názov senzoru a slovný popis sady grafov. Väzbou *N:1* sa viaže na entitu identifikujúcu zariadenie.

Pre získanie potrebných dát pre špecifický graf teda slúži kombinácia sady grafov a na ňu naviazané dátové záznamy jednotného typu spracovania.

5.2 Modul pre hromadné spracovanie dát

Hromadné spracovanie v zhlukoch prebieha v praxi väčšinou periodicky. Spracováva väčšie množstvo dát, čo spôsobuje dlhšie trvanie výpočtov.

Hlavnou funkciou modulu je v pravidelných časových intervaloch načítať namerané surové dáta uložené na disku, aplikovať vybrané metódy pre spracovanie signálu a výsledky zapísať do databázy.

Predspracovanie dát

Predspracovanie dát do vhodného formátu pre výpočty začína načítaním všetkých absolútnych ciest k TDMS súborom (opísané v 4.1.2) nameraných z jedného nástroja. Ku každej absolútnej ceste, ktorá reprezentuje súbor v ktorom je uložené meranie, sa priradí časový záznam, v ktorom bolo toto meranie zaznamenané.

Následne sa odstránia cesty k súborom, ktoré už boli spracované. Záznam o poslednom časovom spracovaní pre aktuálnu sadu dát sa nachádza v databáze.

Pre predspracovanie a aj spracovanie dát som sa rozhodol použiť knižnicu **Pandas**. Pandas je opensource *Python* knižnica, ktorá ponúka vysoko výkonné, ľahko používateľné dátové štruktúry a dátove analýzy [9].

Spracovanie dát

Najnáročnejšie na výpočtovú techniku je z hromadného spracovania dát načítavanie súborov so surovými dátami do pamäte a následné výpočty nad týmito hodnotami.

Pre zrýchlenie spracovania som sa rozhodol optimalizovať modul pomocou rozdelenia spracovania do viacerých vlákien. Každý súbor sa musí načítať do pamäte, otvoriť a pre každý kanál v TDMS štruktúre, ktorý obsahuje dáta, je potrebné vyrátať niekoľko hodnôt. Najpomalšie je práve načítavanie súboru, preto nemá význam otvárať jeden súbor vo viacerých vláknach a v nich paralelne analyzovať dáta. Ako najrozumnejšiu verziu súbežných vlákien som teda zvolil rozdelenie súborov na určitý počet skupín a každá skupina bude spracovaná v osobitom vlákne.

Samotné výpočty nad vibračnými dátami ako napríklad priemer hodnôt, šikmosť alebo efektívna hodnota zabezpečuje knižnica Pandas.

Výsledky z každého vlákna sa zapisujú do databázy. V každom vlákne sa otvára osobitné pripojenie s databázou a výsledkom je jeden príkaz na vloženie, ktorý zahŕňa všetky hodnoty. Po úspešnom ukončení výpočtov a zápise sa ešte zapíše čas posledného spracovaného súboru do databázy.

Prophet

Pri študovaní možností spracovania dát, najmä v prípade časovo označených dát, ma zaujal projekt *Prophet* vyvíjaný spoločnosťou *Facebook*. Prophet je procedúra na predpovedanie údajov v časových radoch, založených na aditívnom modely, kde nelineárne trendy vyhovujú ročnej, týždennej a dennej sezónnosti. Je odolný voči chýbajúcim údajom a posunom v trende a spravidla dobre zvláda odľahlé výsledky [4].

Predpovedá teda, ako budú vyzerať dáta v budúcnosti na základe trendov za dlhšie časové obdobie. V prípade industriálnych dát by mohol odhadnúť napríklad vyťaženie strojov. Rozhodol som sa teda do výslednej aplikácie pridať túto procedúru.

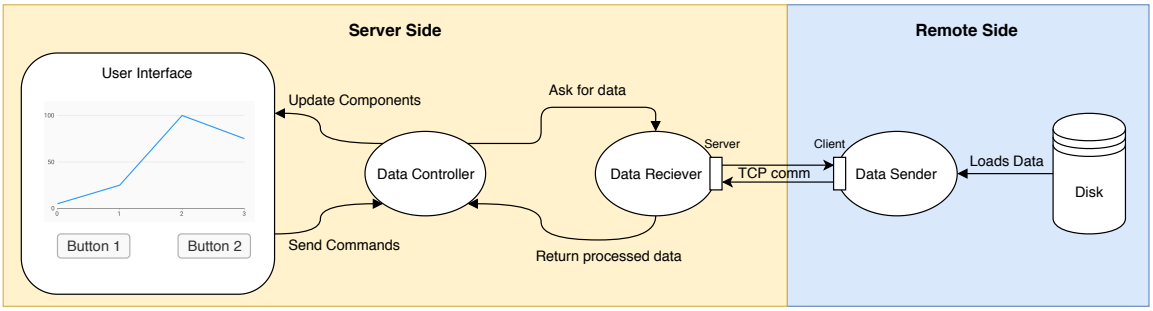
Vstupom nie sú surové, ale už spracované dáta. Pre výpočet sa však využívajú všetky dáta doposiaľ dostupné v rámci jedného senzoru.

Aj keď funkcia *Prophet* je formou spracovania dát v zhluku, nebude sa volať periodicky a pre všetky dáta, ale len na vyžiadanie používateľa.

5.3 Modul pre spracovanie dát v reálnom čase

Úlohou modulu je načítanie dát, výpočet analýzy nad dátami a zobrazovanie výsledkov v takmer reálnom čase. Podľa schémy na obrázku 5.1 je spracovanie v reálnom čase na vstupe so surovými dátami a na výstupe s užívateľským rozhraním webovej aplikácie, pomocou ktorého sa dá časť systému ovládať.

Návrh je rozdelený na tri základné časti - objekt pre správu práce s dátami (*Data Controller*), objekt, ktorý prijíma dáta (*Data Reciever*) a objekt pre spracovanie a odosielanie surových dát (*Data Sender*). Rozloženie a väzby medzi takýmito objektmi je znázornené na obrázku 5.3.



Obr. 5.3: Modul pre spracovanie dát v reálnom čase.

Objekt pre kontrolovanie dát má väzby na komponenty v grafickom rozhraní aplikácie. Reaguje na aktivitu užívateľa, najmä stlačenia tlačidiel, ktorými ovláda ovládanie simulácie spracovania dát. Naviazaný je aj na graf, ktorý napĺňa dátami od objektu pre prijímanie dát. Grafické a funkčné komponenty vo frameworku *Dash* umožňujú takéto väzby a ponúkajú možnosť ináleho načítavania údajov do grafov a iných prvkov.

Ukladanie prijatých spracovaných dát zabezpečuje objekt označený ako *Data Reciever*. Po vytvorení otvorí TCP komunikáciu ako server a čaká na pripojenie klienta, ktorý bude posielať spracované dáta. Reaguje na periodické žiadosti od objektu pre kontrolovanie dát predaním nazbieraných dát.

Objekt pre odosielanie dát postupne spracováva surové dáta z disku a výsledné hodnoty posiela cez TCP komunikáciu, v ktorej figuruje ako klient.

Pre pochopenie simulácie funkčnosti v reálnom čase si je potrebné uvedomiť ako fungujú senzory, ktoré partnerská firma využíva pre sledovanie zariadení. Senzory sú od spoločnosti *National Instruments*, čo je zároveň spoločnosť, ktorá vyvinula súborový formát TDMS, v ktorom sú surové dáta uložené (viac v 4.1.2). Jednotky so senzormi umiestnené na sledovaných zariadeniach zbierajú informácie zo senzorov, zapisujú ich do súborov a až tie odosielajú po sieti cez FTP protokol. Výsledkom na prijímacom konci je teda nový súbor na disku.

Data Sender, zodpovedný za simuláciu posielania v reálnom čase, teda kontroluje príslušný priečinok do ktorého sa prijaté surové dáta ukladajú. V prípade prijatia nového súboru ho spracuje a odošle výsledok spracovania na server, kde sa ďalšie objekty postarajú o to aby bol zobrazený užívateľovi.

Trojicu objektov (*Sender, Reciever, Controller*) som navrhol tak, aby umožnila sledovanie jedného zariadenia. Takáto trojica teda vznikne pre každé zariadenie, čím sa umožní súbežné sledovanie viacerých zariadení naraz.

Opísaný návrh sa zameriava na stanovené požiadavky celkovej aplikácie. Predpokladá industriálne dáta, ktoré sú ale v rozmeroch dodaných od partnerskej firmy, výpočtovú techniku príslušnú menšej firme a klasický súborový systém. Do budúcna by som ale rád navrhol aj riešenie pre väčší počet dát, lepšie reprezentujúci pojem *veľké dáta*.

Pri navrhovaní aplikácie som si naštudoval aj možnosti použiteľné pre väčšie dáta.

5.4 Webová aplikácia

V tejto podkapitole je opísaný postup pri výbere vhodného frameworku použitého pre vývoj aplikácie a návrh riešenia pre užívateľské rozhranie.

5.4.1 Výber frameworku

Framework (aplikačná štruktúra) schopný zabezpečit splnenie požiadaviek systému, je prínosom pre vývoj aplikácie. Framework by mal byť vhodný pre vizuálnu prácu s dátami, dobre dokumentovaný, intuitívny a ideálne napísaný v jazyku *Python*

Flask

Flask je webový "mikroframework", čo znamená že dokáže fungovať bez akýchkoľvek externých knižníc a nástrojov. Napísaný je v jazyku *Python*. Ponúka možnosti pre vývoj jednoduchých webových aplikácií s možnosťou integrácie rôznych nástrojov. Zvláda základné spracovanie a jednoduchú vizualizáciu dát [8].

Bokeh

Bokeh je interaktívny *Python* framework zameraný na vizualizáciu a moderné webové prezentácie. Umožňuje vytváranie komplexných grafov pomocou jednoduchých programovacích prvkov. Ponúka vysoký výkon aj nad dátami veľkého množstva alebo plynulého spracovania dát v reálnom čase. Postavený je nad populárnou platformou *Anaconda Python* [7].

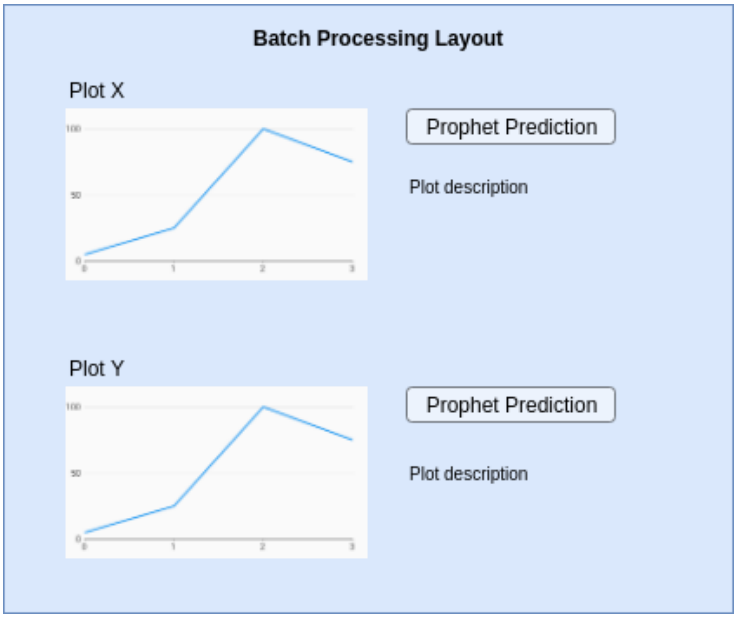
Dash

Dash je produktívny *Python* framework pre vývoj webových aplikácií. Vyvinutý je ako nadstavba nad knižnicami a frameworkami ako *Flask*, *Plot.ly* a *React.js*. Dash je ideálny pre aplikácie, ktoré pracujú s dátami a zameriavajú sa na ich spracovanie a vizualizáciu [12]. Aplikácie napísané v Dash sú vhodné pre väčšinu platforiem, nakoľko sa zobrazujú v internetových prehliadačoch. Dash je voľne dostupný ako opensourcová *Python* knižnica, spravovaná spoločnosťou *Plotly*.

Ako framework pre výslednú webovú aplikáciu som si vybral **Dash**. Ponúka jednoduché riešenia pre dátovú vizualizáciu a spracovanie. Je v neustálom vývoji a aj keď je zo spomínaných frameworkov najmladší, je kvalitne zdokumentovaný a počet užívateľov rapídne rastie.

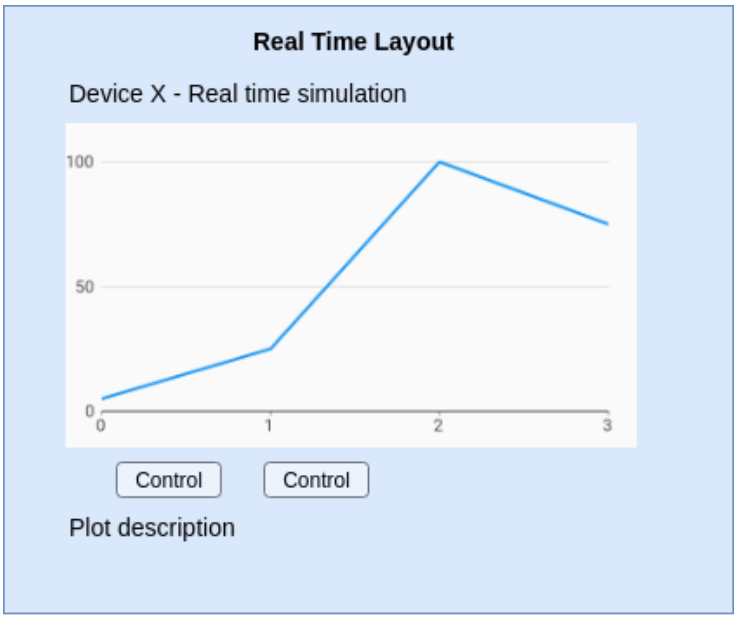
5.4.2 Grafické užívateľské rozhranie

Najdôležitejším prvkom pre užívateľa je sledované zariadenie. Pre každé zariadenie si v užívateľskom rozhraní bude možné vybrať z dvoch hlavných funkcií - prezentovanie výsledkov spracovania v zhlukoch a sledovanie zariadenia v reálnom čase.



Obr. 5.4: Návrh rozloženia užívateľského rozhrania pre prezentovanie výsledkov spracovania dát v zhlukoch.

Pre obidve varianty som vytvoril návrh rozloženia stránok, ktoré sú zobrazené na obrázkoch 5.4 a 5.5. Tieto prvotné návrhy, s menšími zmenami, tvorili aj finálne rozloženie stránok.



Obr. 5.5: Návrh rozloženia užívateľského rozhrania pre spracovanie dát v reálnom čase.

Kapitola 6

Implementácia

Výsledná aplikácia má fungovať ako webová aplikácia, poskytujúca prehľad dostupných zariadení. Každé zariadenie má byť možné sledovať v reálnom čase, teda či senzory, ktoré sú osadené na ložiskách posielajú údaje, tieto údaje agregovať a zobrazovať. Pre každé zariadenie existuje možná vizualizácia historických dát, presnejšie pre každý senzor na zariadení. Hromadné výpočty nad nazbieranými dátami sa spúšťajú periodicky tak aby neblokovali užívateľa vo využívaní aplikácie. Na podnet užívateľa má byť možné analyzovať špecifické dáta vybranými metódami.

6.1 Dash webová aplikácia

Základom výsledného systému je webová aplikácia napísaná vo frameworku **Dash**. Dash aplikácie sú zložené z dvoch základných častí. Prvou je rozloženie stránky (angl. *layout*), ktoré opisuje ako aplikácia vyzerá. Druhú časť reprezentujú funkcie pre spätné volania (*angl. callbacks*), ktoré opisujú interaktivitu aplikácie, reakcie na udalosti vyvolané interne alebo užívateľom.

Inštalácia Dash knižníc prebehla pomocou inštalatéra Python balíčkov *pip*:

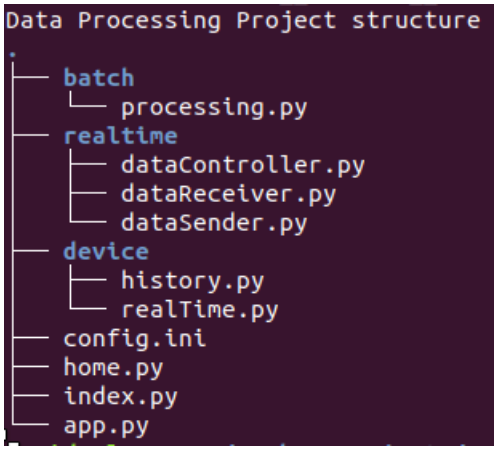
```
pip install dash
```

Takáto inštalácia zabezpečí aj nainštalovanie knižníc `dash_core_components` (ďaej `dcc`) a `dash_html_components` (ďalej `html`), ktoré poskytujú triedy pre všetky vizuálne komponenty pre aplikáciu. Dash je aktívne vyvíjaný a pomerne často vychádza nová verzia. V čase implementácie som použil najaktuálnejšiu verziu **0.39.0**.

Štruktúra projektu

Usporiadanie súborov projektu je zobrazené na obrázku 6.1 a zrkadlí funkčnosť celého systému. Základ aplikácie je vytvorenie inštancie Dash triedy, ktorá reprezentuje celú aplikáciu. Takýto objekt je vytvorený v súbore `app.py`. Obsahuje základné komponenty ako `layout` a všetky `callback` funkcie, spomínané v úvode podkapitoly.

Dash prevádzkuje webovú aplikáciu ako "jednostránkovú" aplikáciu, čo znamená že pri zmene lokácie, zobrazení novej stránky, neprebehne úplné načítanie aplikácie nanovo. Načítanie stránok z rôznych súborov musí teda prebiehať pomocou `callback` funkcie, ktorá reaguje na zmenu komponent `dcc.Location`, ktorý je súčasťou rozloženia prvotnej stránky. Funkcia zanalyzuje vstupnú adresu a získa rozloženie stránky zo súboru, ku ktorému lokácia vedie. Vracia rozloženie žiadanej stránky, ktoré je nahrané do rozloženia tej aktuálnej. Pre



Obr. 6.1: Projektová štruktúra súborov.

reprezentáciu `callback` funkcie a spôsobu presmerovania a načítania stránok je kód funkcie zobrazený vo výpise 6.1.

```
# URL = /device/{realtime|history}/{deviceID}
@app.callback(Output('page-content', 'children'),
              [Input('url', 'pathname')])
def display_page(pathname):
    pathnameArray = str(pathname).split('/')
    if len(pathnameArray) > 1:
        if pathnameArray[1] == "device":
            if pathnameArray[2] == "history":
                return history.returnLayout(pathnameArray[3])
            if pathnameArray[2] == "realtime":
                return realTime.returnLayout(pathnameArray[3])
    return home.layout

app.layout = html.Div([
    dcc.Location(id='url', refresh=False),
    html.Div(id='page-content'),
])
```

Výpis 6.1: Callback funkcia pre presmerovanie na základe URL (súbor index.py)

Súbor `index.py` je teda štartovacím skriptom celej aplikácie, keďže obsahuje základné rozloženie stránky a spracovanie presmerovania a zároveň spúšťa Dash objekt, pomocou príkazu `app.run_server(threaded=True, debug=False)`.

Konfiguračný súbor `config.ini` obsahuje prístupové údaje pre databázu a cestu k priečinkom so surovými dátami. Pre manipuláciu s konfiguračným súborom som využil balíček `configparser`.

Súbory `home.py`, `history.py` a `realTime.py` obsahujú rozloženie a funkcie pre domovskú stránku, stránku pre zobrazovanie historických dát a stránku pre sledovanie zariadení v reálnom čase (v poradí akom boli menované). Priečinok `realtime` obsahuje triedy zabezpečujúce funkčnosť sledovania zariadení a funkcie v súbore `batch/processing.py` spracovávajú dáta v zhlukoch.

Databáza

Databáza je podľa návrhu 5.1 implementovaná v jazyku **MySQL**. Pre vytvorenie a správu databázy som použil nástroj **Adminer**[1], ktorý cez grafické rozhranie webovej aplikácie umožňuje vytvoriť vlastnú databázu. Entity databázy *Device, Plot_set* a *Data_entry* sú navrhnuté na obrázku 5.2 a ich implementácia je v zásade jednoduchá.

Potrebné bolo zabezpečenie toho, aby nebolo možné do databázy uložiť záznam reprezentujúci bod v ľubovoľnom grafe, tak aby sa na jedom grafe nachádzalo viac hodnôt pre jeden časový záznam. Dosiahol som to obmedzením využívajúcim príkaz `unique`:

```
ALTER TABLE data_entry
ADD CONSTRAINT ucData_entry UNIQUE (‘timestamp‘, ‘type‘, ‘plot_set_id‘)
```

Časové údaje sú uložené ako `unix timestamp`, teda celé číslo. Reprezentuje počet sekúnd, ktoré uplynuli od takzvanej "Epochy Linux" (1.1.1970) a predstavuje jednotný spôsob ukladania časových záznamov.

Pre pripojenie do databázy som využil Python balíček pre prácu s MySQL `MySQLdb`.

6.2 Hromadné spracovanie dát

Spracovanie zhromaždených dát zabezpečuje funkcia `processBatches()`. Pre každé zariadenie získa informácie o kanáloch v surových TDMS súboroch, teda názvy senzorov, ktorými boli dáta namerané (opísané v kapitole 4). Z kombinácie zariadenia a senzoru sa identifikujú sady grafov pre aktuálne súbory a z atribútu `plot_set.last_processed_time` je známy čas, v ktorom prebehlo predošlé spracovanie.

Keďže v názve súborov sa nachádza časový záznam ich vzniku, dá sa presne identifikovať ktoré súbory treba spracovať ešte pred ich načítaním do pamäte. Pre vybratie vhodných súborov som využil balíčky `glob` (načítanie názvov súborov z disku), `datetime` a `pandas` (dátový typ DataFrame). Po načítaní všetkých názvov súborov do obejktu detového typu `pandas.DataFrame` sa ku každému názvu súboru uloží s využitím lambda funkcie jeho časový záznam a pomocou maskovania sa vyberú iba žiadané názvy súborov.

```
# Filename example = raw_01bf1219_170531_090613.tdms
dfFilenames['date'] = dfFilenames['filenames'].apply(
    lambda x: datetime.strptime(x[len(x) - 18:len(x) - 5], '%y%m%d_%H%M%S'))
maskForNewFilesOnly = dfFilenames['date'] > lastTimeOfProcessing
dfFilenames = dfFilenames[maskForNewFilesOnly]
```

Výpis 6.2: Využitie maskovania v dátovom type DataFrame pre vybratie potrebných názvov súborov

Zoznam mien súborov na spracovanie a informácie o senzoroch sú vstupom pre funkciu `computeFeatureExtractions()`, ktorá spracuje samotné surové dáta. Pre prácu s TDMS súbormi vyvinula spoločnosť *Natinal Instruments* (vyvinula aj formát TDMS) Python balíček `nptdms`. Umožňuje čítanie a vytváranie súborov, s dátami pracuje ako s `numpy` poliami. Využívam hlavne funkcie `nptdms.TdmsFile(pathToFile)` a *nptdms.as_dataframe()*.

Metódy pre získanie potrebných údajov z vibračných dát, opísané v 3.2 sú implementované ako:

- Šikmosť (*angl. Skewness*) sa vyráta funkcou `skew()` z knižnice `scipy.stats`

[1] https://www.adminer.org/

- Nestálosť (*angl. Variance*) sa vyráta funkcou `var()` z balíčka `numpy`
- Efektívna hodnota (*angl. Root Mean Square, RMS*) pomocou matematických funkcií z balíčka `numpy`

Počas prechodu všetkými súbormi sa zapisujú výsledky do textového reťazca, ku ktorému sa po dokončení cyklu pridá hlavička *SQL* príkazu pre vkladanie do databázy a tento príkaz sa vykoná. Po dokončení sa ešte aktualizuje `plot_set.last_processed_time` príslušnej sady grafov.

Paralelizácia

Podľa návrhu sa výpočty paralelizujú rozdelením do vlákien. Pre túto potrebu som využil balíček `threading`. V každom vlákne je volaná funkcia `computeFeatureExtractions()`, ktorej ako jeden z argumentov predávam časť z názvov súborov, ako je vidieť na výpise 6.3.

```
filenameSplits = np.array_split(np.asarray(filenames), NO_OF_THREADS)
threads = []
for index, filenamesSplit in enumerate(filenameSplits):
    t = threading.Thread(target=computeFeatureExtractions,
                         args=(device.name, filenamesSplit,
                            sensorExtractionTypesInfo, INFO, index))
    threads.append(t)
    t.start()
for thread in threads:
    thread.join()
```

Výpis 6.3: Rozdelenie výpočtov nad surovými dátami pomocou vlákien

Cron

Spúšťanie hromadného spracovania prebieha periodicky v čase, kedy nie je predpokladané, že aplikácia bude využívaná. Pre tento účel je využitá linuxová utilita **cron**. Cron je časový plánovač úloh. Základom je konfiguračný súbor **crontab**, do ktorého sa dopisujú úlohy podľa potreby. Príkaz v súbore crontab pre našu aplikáciu vyzerá nasledovne:
`0 3 * * * /{cesta k Pythonu}/bin/python /{cesta ku projektu}/batch/processing.py`
Kde čísla na začiatku určujú, že sa príkaz vykoná každý deň (0 na pozícií dní) o tretej hodine po polnoci.

Prophet

Ako sa uvádza v návrhu, nástroj na predikciu dát v časových sériách **Prophet** je využívaný ako forma spracovania hromadných dát. Nevykonáva sa ale periodicky, ale iba na podnet užívateľa pomocou grafického rozhrania. Implementovaná je v súbore `history.py` ako `callback` funkcia, ktorá reaguje na stlačenie tlačidla. Výstupom je graf, ktorý porovnáva reálne a predpovedané dáta.

Trieda `Prophet` z balíčku `fbprophet` je zodpovedná za všetky výpočty. Samotná predpoveď je implementovaná v kóde nasledovne:

```
# df = DataFrame of time series data
m = Prophet(seasonality_mode='multiplicative').fit(df)
```

```
future = m.make_future_dataframe(periods=daysToForecast)
fcst = m.predict(future)
```

Parameter `seasonality_mode='multiplicative'` určuje, že pravidelnosť a sezónnosť dát je nepravidelná, narozdiel od hodnôt, ktoré sa viažu na napríklad ročné obdobia alebo sviatky.

Pred tým ako sa ale zavolá funkcia `Prophet.predict()`, je potrebné vstupné dáta upraviť pre potreby triedy *Prophet*. Pre transformáciu dát som sa nechal inšpirovať postupom uvedenom v [1], pre metódu Box-Cox transformácie. Box-Cox transformácia vyberie hodnotu, ktorá dosiahne najoptimálnejšiu aproximáciu k normalizácií dát. Balíček `scipy.stats` ponúka triedu `boxcox` pre výpočet tejto transformácie.

Takto upravené dáta sú vstupom pre Prophet. Výstupom sú ale dáta, ktoré treba inverzne transformovať do pôvodných hodnôt (trieda `scipy.special.inv_boxcox`). Pre vhodné vizuálne zobrazenie ešte spájam pôvodné hodnoty, predikované hodnoty a hraničné hodnoty, ktoré taktiež predpokladá Prophet do jedného grafu. Takto vyhotovený graf je výsledkom predikcie.

6.3 Spracovanie dát v reálnom čase

Implementácia modulu pre spracovanie a zobrazovanie dát v reálnom čase je realizovaná na základe návrhu opísanom v podkapitole 5.3. Vytvorené sú tri triedy - `DataSender`, `DataReciever` a `DataController`. Tieto triedy sú implementované v rovnomenných súboroch (so zmenou začiatočného písmena na malé) v priečinku `realTime`.

DataSender

Inštancia triedy `DataSender` sa po vytvorení pokúša o pripojenie na *TCP* server, ktorý je vytvorený objektom `DataReciever`. Komunikácia prebieha cez socket (balíček `socket`), adresa a port sú predané ako parametre pri vytváraní objektu. Po úspešnom nadviazaní kontaktu z TCP serverom načíta názvy súborov, ktoré má spracovať z disku. Získanie súborov a vyrátanie efektívnej hodnoty (ako zvolenej metódy pre spracovanie v reálnom čase) je takmer identické ako pri hromadnom spracovaní súborov opísanom v 6.2. Rozdielom je že čas, od ktorého je potrebné spracovať údaje je predaný ako parameter, nie získaný z databáze. Po prvom spracovaní všetkých načítaných súborov sa zmení čas na aktuálny a znova sa opakuje cyklus. Ak pribudli na disk nové súbory so surovými dátami, budú spracované. Spracované dáta sa po každom výpočte odosielajú cez TCP komunikáciu ako pole hodnôt o veľkosti dva, kde prvá hodnota je časový údaj vo formáte `timestamp` a druhá hodnota je vypočítaná efektívna hodnota.

Cyklus sa opakuje, pokiaľ je atribút objektu `DataSender.paused` nastavený na pravdivú hodnotu. Tento atribút môže užívateľ prepínať pomocou objektu `DataController` cez grafické rozhranie.

DataReciever

Táto trieda slúži ako sprostredkovateľ medzi triedou `DataReciever` a `DataContorller`. Po inicializácií objektu okamžite vytvára TCP server na adrese a porte, ktoré boli poskytnuté ako parameter. Otvorený port čaká na pripojenie klientom. V hlavnom životnom cykle prijíma dáta, ktoré spracuje z binárnej podoby. Tie ukladá do polí `DataReciever.values`

a `DataReciever.timestamps`. Cyklus prijímania dát prebieha, iba ak je atribút `DataReciever.paused` nastavený na pravdivú hodnotu.

Objekt `DataController`, ktorý obsahuje odkaz na `DataReciever`, môže meniť stav `DataReciever.paused` a volať funkciu `DataReciever.getData()`, ktorá odovzdá nahromadené prijaté dáta ako štruktúru vhodnú pre pridanie do grafu. Po odoslaní dát sa vyprázdnia atribúty, ktoré tieto dáta držali.

DataController

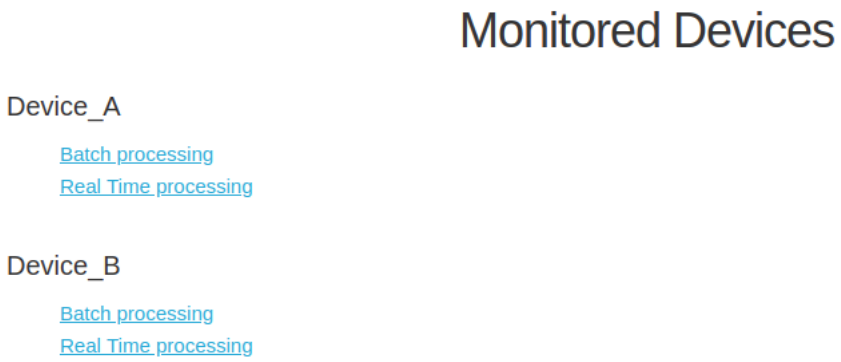
`DataController` pri vytvorení dostane ako parametre objekty `DataReciever` a `DataSender`, ktoré ovláda na podnet užívateľa. List objektov `DataController` je globálna premenná zadefinovaná v `app.py` a obsahuje záznam o objekte ku každému sledovanému zariadeniu. Pri načítaní stránky pre sledovanie zariadenia sa objekt načíta z globálnej premennej. Ak tento objekt ešte neexistuje, vytvoria sa postupne objekty `DataReceiver` a `DataSender`, ktoré sa potom predajú ako parameter pre vytvorenie objektu pre kontrolovanie týchto inštancií. Každé zariadenie teda predstavuje jedna trojica objektov, ktorá sa stará o prenos a zobrazovanie dát.

Pre aktualizovanie grafov som využil komponent `dcc.Interval(interval=300ms)`, ktorý sa využije ako vstup pre `callback` funkciu, ktorá cez kontrolér získava dáta pre graf.

6.4 Užívateľské rozhranie

Dash využíva základné *html* prvky v spojení so zložitejšími grafickými prvkami (napríklad `Graph, Scatter`) pre vytvorenie celkového grafického prostredia.

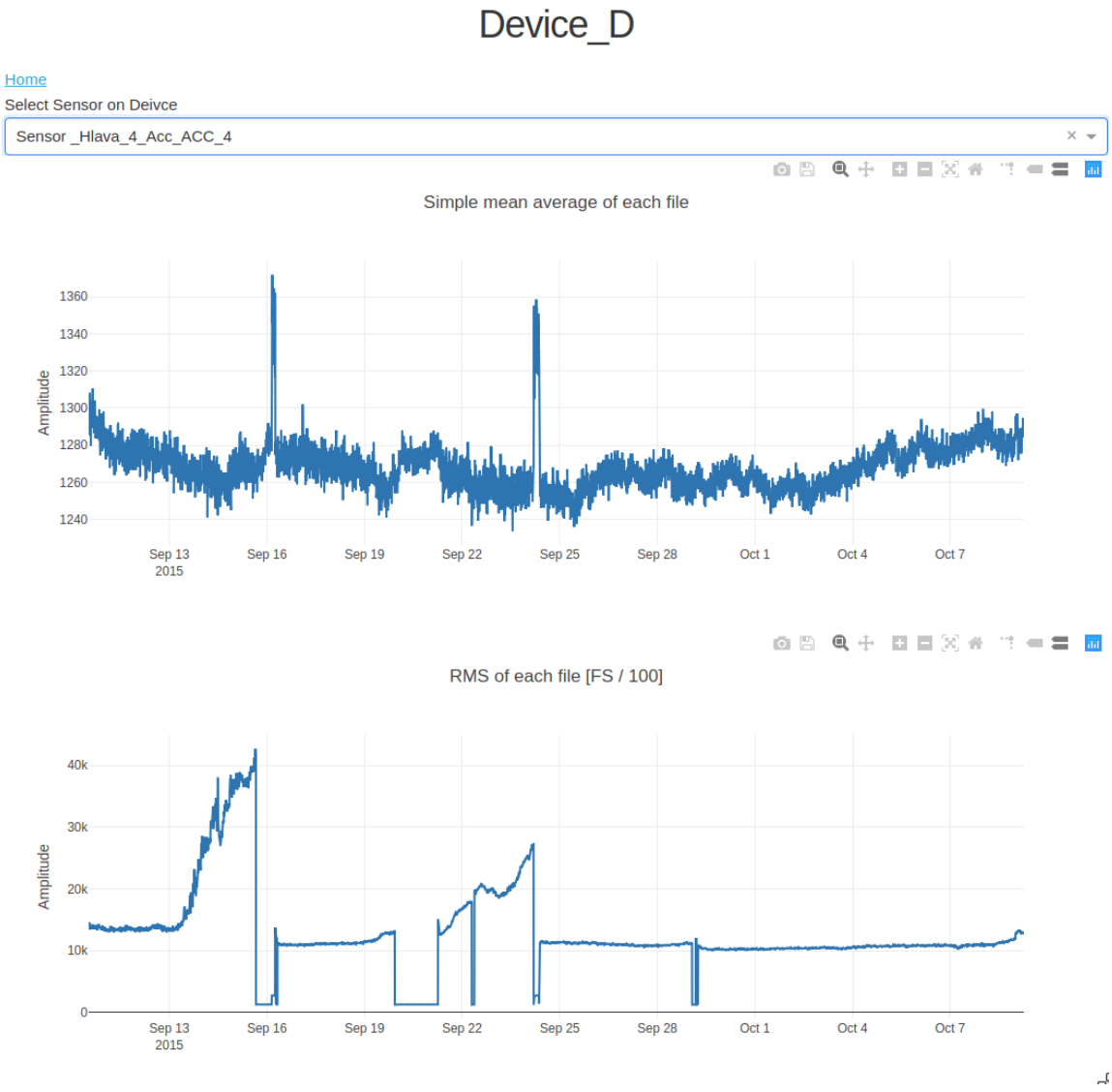
Úvodná stránka predstavuje navigáciu medzi dostupnými zariadeniami s výberom možnosti pre zobrazenie historických dát alebo sledovanie v reálnom čase. Úvodná stránka je zobrazená na obrázku 6.2. Odkazy sú v rozložení implementované objektami `dcc.Link()`.



Obr. 6.2: Domovská obrazovka aplikácie.

V prehľade historických dát sa dáta zobrazujú po vybratí senzoru. Voľba senzoru je implementovaná grafickým komponentom `dcc.Dropdown()`, ktorý svoje možnosti výberu získa z databázy. Na vybratie hodnoty reaguje `callback` funkcia, ktorá následne generuje grafy hodnotami z databáze. Hlavička takejto funkcie je zobrazená v 6.4, príklad prehľadu spracovaných dát z vybraného senzoru je zobrazená na obrázku 6.3.

```
1 @app.callback(Output('output-history-graph', 'children'), [
```



Obr. 6.3: Spracované dáta po výbere senzoru na zariadení.

```
            Input('sensor-dropdown', 'value')
])
def historyGraphOutput(plot_setId):
    # Access Database, populate dcc.Graph objects
    # ...
    return graphs
```

Výpis 6.4: Hlavička callback funkcie generujúcej grafy s historickými dátami

Pri sledovaní zariadení v reálnom čase ovláda užívateľ celú simuláciu pomocou tlačidiel, ktoré sú implementované pomocou komponentu `html.Button()`. Má možnosť zastaviť alebo obnoviť posielanie a prijímanie dát, alebo reštartovať celú simuláciu. Rozloženie stránky je zobrazené na obrázku 6.4.

Každý graf je vytvorený komponentom `dcc.Graph()`, ktorý je navrhnutý na základe a zdieľa rovnakú syntax ako grafy z opensource knižnice `plotly.py`. Tieto grafy majú im-

Obr. 6.4: Grafické rozhranie sledovania zariadenia v reálnom čase.

plementované nástroje pre prácu s nimi ako napríklad vyrezávanie, približovanie, fixovanie osí, uloženie grafu ako obrázok na disk a iné.

6.5 Nasadenie webovej aplikácie na server

V predvolenom nastavení sú aplikácie Dash spustené lokálne na adrese `localhost:8050`, bez možnosti vonkajšieho prístupu. Ak má byť Dash aplikácia verejná a prístupná z internetu, je potrebné ju nasadiť na server. Od partnerskej firmy som dostal k dispozícií linuxový server, na ktorom aplikácia môže bežať.

Dash aplikácie bežia ako nadstavba `Flask` aplikácie, čo zjednodušuje nasadenie na server, nakoľko nasadenie `Flask` aplikácií je zdokumentované a otestované komunitou vývojárov. Rozhodol som sa využiť aplikačný kontajner **uWSGI** a opensource softvér pre spravovanie webových serverov **Nginx**.

Na serveri partnerskej firmy som musel vytvoriť virtuálne prostredie v ktorom bude aplikácia bežať. Pre tento účel som využil Python balíček `virtualenv`. Pre správu projektu som využil voľne dostupný systém pre kontrolu verzie **GitHub**[2]. Inštalácia uWSGI aplikácia, nastavenie služby Nginx a konfiguračné súbory k tomu potrebné sú opísané v informačnom súbore `readme.md`, priloženom pri zdrojových kódoch. Doména, na ktorej aplikácia beží, je poskytnutá partnerskou firmou.

[2] https://github.com/

Kapitola 7

Vyhodnotenie aplikácie

V kapitole sa nachádza vyhodnotenie celej aplikácie, porovnanie výslednej aplikácie s požiadavkami, opísané poznané nedostatky. Možnosti rozšírenia sa nachádzajú v poslednej podkapitole.

7.1 Testovanie aplikácie

Pre testovanie aplikácie boli navrhnuté testy pre tri oblasti funkcionality. Testovaná bola hraničná situácia spracovania zhluku dát, predpoveď dát pomocou knižnice `Prophet` a monitorovanie viacerých zariadení v reálnom čase paralelne.

Testovanie spracovania zhlukov dát

Pre otestovanie spracovania hromady dát som zvolil situáciu, kedy vopred neboli spracované žiadne dáta. Reálne by takáto situácia nastala, ak by sa za jeden deň prijalo množstvo dát, ktoré zodpovedá zhromaždeným dátam pre túto prácu. V tejto situácií som sledoval, či systém zvládne záťaž v podobe dlhodobých výpočtov vo viacerých vláknach. Taktiež som si overil či je čas pravidelného spúšťania výpočtov vhodný, teda či sa výpočty stihnú ukončiť pred začatím bežnej pracovnej doby (7 hodín ráno). Výsledky sú zobrazené v tabuľke 7.1. Stĺpec *Čas spracovania* obsahuje najdlhší časový úsek výpočtu zo všetkých vlákien a symbol $\sum$ v poslednom riadku nereprezentuje sumu týchto hodnôt, ale celkový čas behu programu.

Zariadenie	Počet súborov	Veľkosť súborov [GB]	Čas spracovania
Device_A	1909	14	00h:14m:32s
Device_B	7972	183	02h:51m:39s
Device_C	7113	5.5	00h:12m:18s
Device_D	6887	16	00h:23m:53s
$\sum$	22163	218.5	3h:43m:19s

Tabuľka 7.1: Testovanie spracovania všetkých zhromaždených dát. Počet vlákien pre každé zariadenie = 3.

Vyše 22 tisíc súborov zo štyroch zariadení aplikácia spracovala a zapísala do databázy za niečo málo cez 3 hodiny. Najviac zabralo spracovanie súborov zo zariadenia *Device_B*,

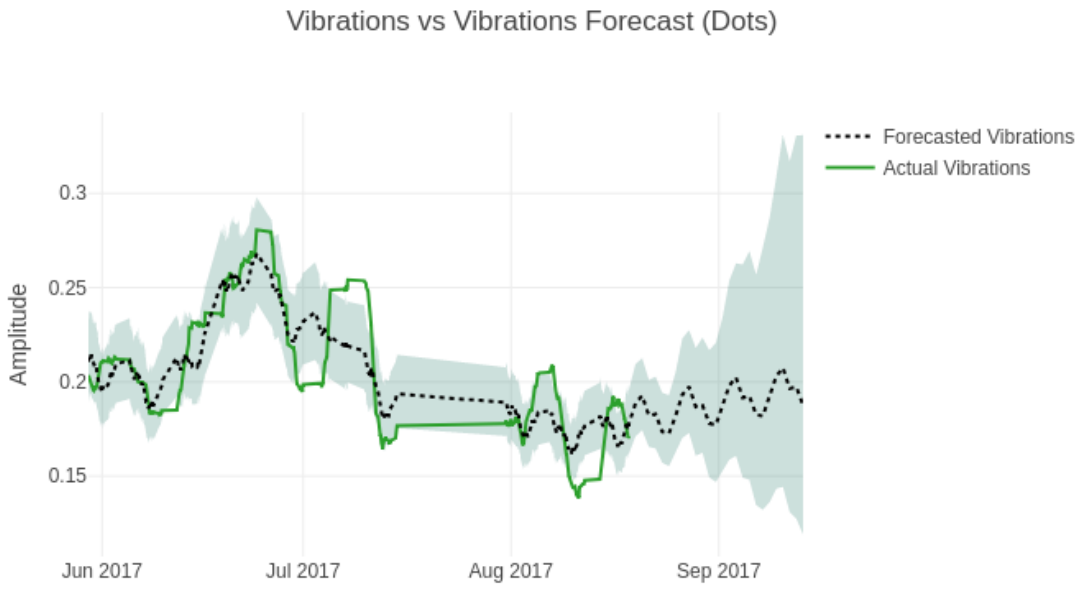
čo sa dá odôvodniť tým, že spomedzi zariadení obsahuje najviac nameraných súborov, ktoré majú zároveň najvyšší počet záznamov na jeden súbor.

Výsledkom testu bolo overenie, že systém takúto záťaž zvládne, ale zaberie to značný čas. Spúšťanie výpočtov som teda prestavil na polnoc (namiesto pôvodných 3 hodín ráno).

Prophet predikcie

Predpoveď dát pomocou knižnice `Prophet` ukázala, že má potenciál využitia v priemyselnej výrobe. Na základe historických dát boli vykonávané odhady vibrácií na tretinu časového úseku z dĺžky celého úseku analyzovaných dát.

Na grafoch sú vykreslené reálne dáta (zelená farba), odhadované dáta (čierna prerušovaná) a výplň medzi dolnou a hornou hranicu odhadu predpovedaných dát (slabo zelené podfarbenie). Pri nepravidelných dátach s veľkými výkyvmi neboli predpovede presné a odzrkadlilo sa to aj na istote predikcií, ako je možné vidieť na obrázku 7.1. Pri pravidelnejších dátach však trend bol badateľný, predpovedané dáta od začiatku výpočtov lemujú priebeh skutočných dát, čo je viditeľné na obrázku 7.2.

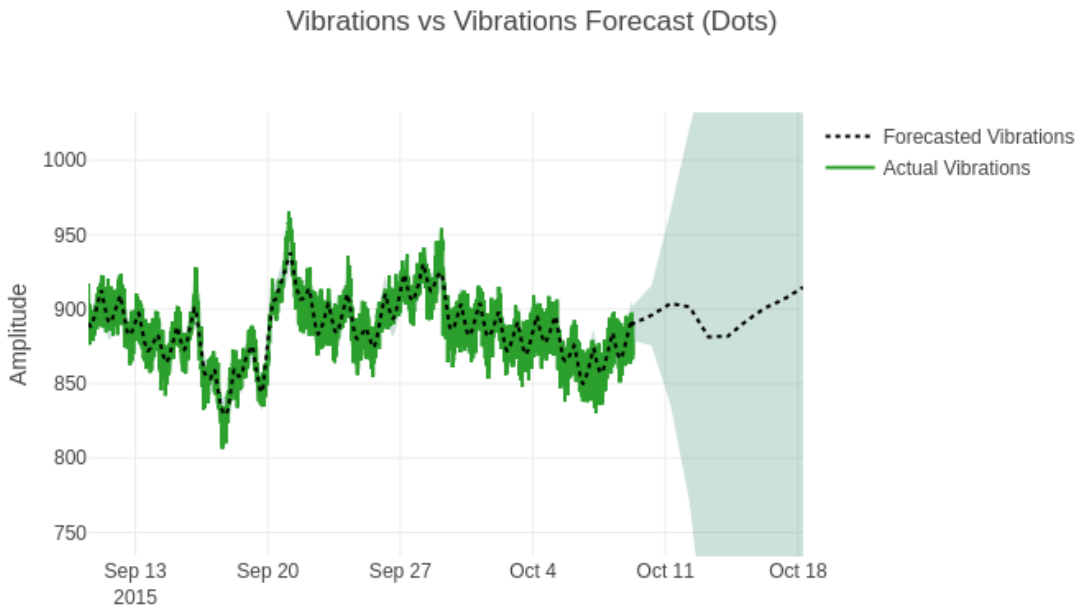


Obr. 7.1: Prophet predikcia neefektívna pre daný typ dát. Zariadenie = *Device_A*, senzor = *_raw_3*, graf = Spracované partnerskou firmou.

Rozhodovanie a zmena výrobných procesov na základe `Prophet` predikcií nie je pravdepodobná. Pri väčšom počte dát, za dlhšie časové obdobie, by však predpovede boli pravdepodobne presnejšie.

Monitorovanie v reálnom čase

Pri testovaní prijímania a zobrazovania dát v reálnom čase som chcel overiť funkcionalitu systému pri otvorení viacerých okien webového prehliadača, v ktorých by boli sledované



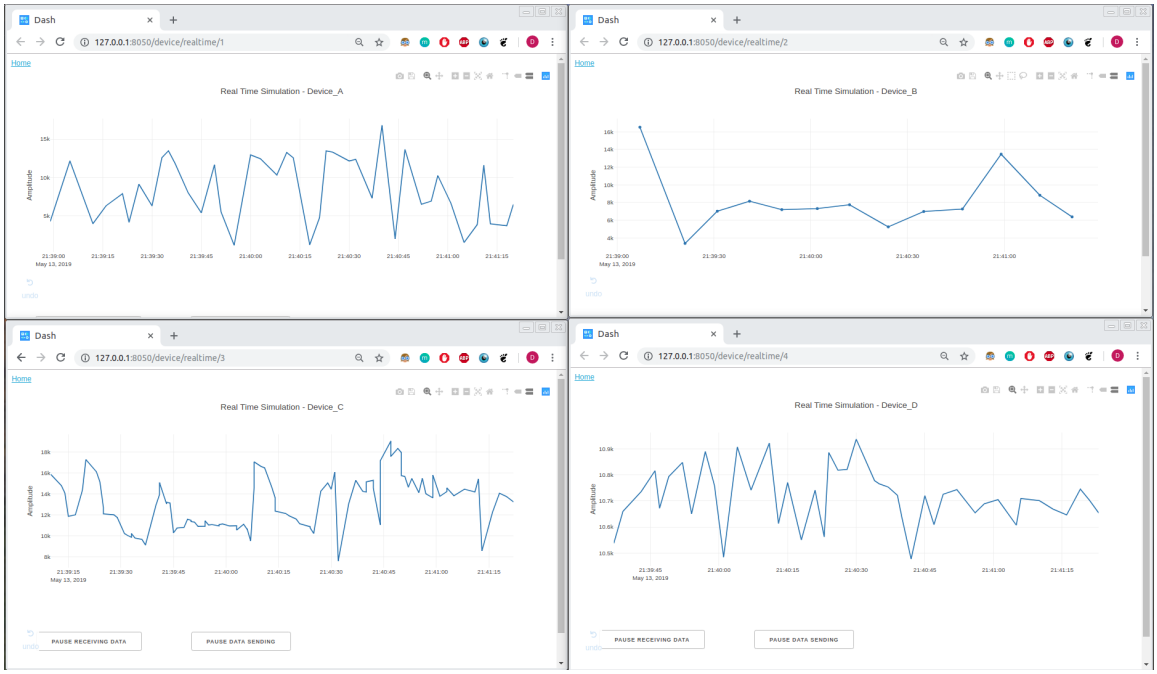
Obr. 7.2: Prophet predikcia s primerane dobrou mierou predpovede. Zariadenie = *Device_D*, senzor = *_Hlava_5_Acc_ACC_5*, graf = Efektívna hodnota.

rôzne zariadenia. Pri teste som otvoril štyri okná, každé s iným zariadením. Po načítaní všetkých rozhraní som postupne spustil simulácie odosielania dát. Po určitej časovej dobe (5 min) som zastavil niektoré zo simulácií a následne znova spustil. Sledoval som reakčnú dobu tlačidiel, ktorá bola citeľne väčšia ako pri sledovaní len jedného zariadenia. Postupne však simulácie reagovali na tlačidlá tak ako mali. Snímka obrazovky, zaznamenaná pri sledovaní všetkých zariadení je na obrázku 7.3.

7.2 Vyhodnotenie požiadaviek a možnosti rozšírenia

Výsledná aplikácia bola navrhnutá a implementovaná s ohľadom na požiadavky stanovené v spolupráci s partnerskou firmou. Aplikácia mala byť schopná spracovávať surové dáta, poskytnuté partnerskou firmou, spracované dáta ukladať a zobrazovať. Pomocou aplikácie malo byť možné sledovať aktuálne prichádzajúce dáta zo zariadení v reálnom čase. Celá aplikácia bola prispôsobovaná softvéru, dátam a prostrediu partnerskej firmy tak, aby sa mohla využívať v rámci tejto firmy.

Surové dáta sú vo výsledku spracovávané viacerými metódami vhodnými pre analýzu vibračných ložísk. Prehľad vybraných metód sa nachádza v 3.2. Výsledkom sú historické grafy, na ktorých je zobrazený priebeh stavu ložiska v čase. Pri vhodnej kombinácií dát a metódy spracovania je jasne vidieť degradácia kondície ložiska a následná výmena za nové, ako napríklad na obrázku 7.4. Iné grafy nedávajú na pohľad veľký význam bežnému pozorovateľovi, no stále môžu byť prospešné odborníkom, ktorý sledujú kondície týchto ložísk.



Obr. 7.3: Testovanie sledovania všetkých zariadení v rovnakom čase.

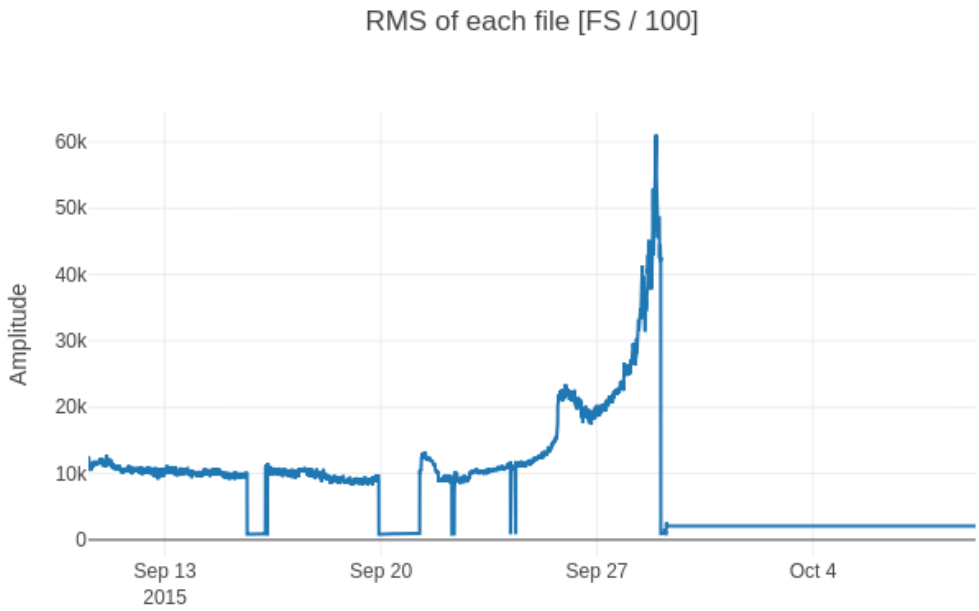
Spracovanie zhlukov dát, nazbieraných zo senzorov, prebieha automaticky. Nastavený čas pre spustenie výpočtov bol na základe testovania a predpokladu, že by výpočty mali prebiehať v dobe, kedy nie je systém vyťažený vysokou prevádzkou (tzn. v nočných hodinách) nastavený na polnoc.

Sledovanie zariadení a samotné spracovanie dát v reálnom čase bolo navrhnuté na základe fungovania softvéru, ktorý zariaďuje samotný zber dát. Tento softvér (ako aj senzory a formát súboru, vyvinutý spoločnosťou *National Instruments*) agreguje dáta a tie až ako TDMS súbory nahráva pomocou FTP protokolu na server. Sledovanie v reálnom čase je v skutočnosti teda nepretržité kontrolovanie, či bol nahratý nový súbor na disk. Po nahratí je spracovaný a odoslaný sieťovou komunikáciou do webovej aplikácie a v reálnom čase zobrazený v grafe.

Takéto spracovanie monitorovania je adekvátne pre partnerskú spoločnosť, no ako forma prúdového spracovania veľkých dát nie je vhodná.

Pri navrhovaní systému som sa zameriaval viac na detaily, ktoré riešili problémy spojené so špecifikáciami od partnerskej firmy, napríklad využívanie klasického súborového systému alebo spôsob sledovania prichádzajúcich TDMS súborov. To ma viedlo k navrhnutiu vlastného systému, ktorý je schopný integrácie s prostredím partnerskej firmy, dokonca aj spĺňa základné požiadavky na spracovanie a monitorovanie dát. Pri implementácií a testovaní som ale narazil na to, že aplikácia nieje vhodne rozšíriteľná a pravdepodobne by nezvládla objemy dát, ktoré sú obecne považované za veľké dáta.

Do budúcna by som prehodnotil využitie softvérových riešení, ktoré som pri návrhu zamietol a to hlavne distribuovaný súborový a výpočtový systém, napríklad *Hadoop*. V spojení s frameworkom *Apache Spark* by mohli predstavovať ideálne riešenie pre očakávaný nárast



Obr. 7.4: Viditeľná degradácia ložiska v čase, reprezentovaná stúpajúcou amplitúdou vibrácií a následná výmena za nové, nepoškodené ložisko.

industriálnych dát, ktoré bude partnerská firma produkovať. *Hadoop* aj *Spark* sú opísané sú v podkapitole 2.3.2.

Distribuované systémy pre veľké dáta pracujú v organizovaných zoskupeniach výpočtových zariadení. Výpočtová časť výslednej aplikácie by mohla poslúžiť ako inšpirácia pre funkcionalitu jednotky v takomto zoskupení (*cluster*). Aplikácia je napísaná v jazyku Python a výstupy ukladá do databázy MySQL a obe jazyky sú vysoko kompatibilné z väčšinou softvérových riešení pre veľké dáta.

Kapitola 8

Záver

Cieľom mojej práce bolo vytvorenie aplikácie pre spracovanie priemyselných veľkých dát. Na práci som spolupracoval s partnerskou firmou, ktorá pôsobí v oblasti priemyslu 4.0, presnejšie sa zaoberá sledovaním a analyzovaním dát z ložísk obrábacích a tvarovací priemyselných zariadení. Aplikácia bola vyvíjaná, tak aby spracovávala a sledovala poskytnuté dáta a bola využiteľná v prostredí partnerskej firmy.

Aplikácia bola navrhnutá podľa *Lambda* architektúry, ktorá spája spracovanie prúdových dát v reálnom čase a spracovanie zhlukov dát v pravidelnom časovom úseku. Takáto architektúra umožňuje zobrazovanie dát, ktoré prichádzajú zo sledovaných zariadení a zároveň ponúka možnosť zobrazenia historických spracovaných dát.

Výsledná aplikácia spracuje dáta z disku a výstupom je zápis do databázy, v prípade spracovania v zhlukoch, alebo vykresľovanie do grafu v reálnom čase, v prípade monitorovania zariadení. Priemyselné dáta spracuje známymi metódami v oblasti sledovania kondície ložísk, ako napríklad sledovanie efektívnej hodnoty, odchýlky alebo šikmosti. Ako variantu spracovania v zhlukoch ponúka aj možnosť predikcie dát s využitím projektu *Prophet*. Prijímanie dát a práca so súbormi je podmienená

Aplikácia spĺňa podmienky stanovené partnerskou firmou. Zariadenia je možné sledovať v reálnom čase a periodicky prebiehajú výpočty nad zhlukmi dát, ktoré sú spracované a výsledky sú uložené do databázy.

Aj keď aplikácia zvláda prácu s dátami, ktoré sú dodané partnerskou firmou, nie je vhodná pre prácu s klasicky veľkými dátami, veľkosť ktorých presahuje stovky gigabajtov. Aplikácia bola prispôsobovaná systému, ktorý funguje v partnerskej firme, ale nevyužíva klasickú architektúru spracovania veľkých dát, najmä distribuovaný systém uloženia dát a rozdelenie výpočtovej sily. V budúcnosti by pri prechode na systém, ktorý pre spracovanie a ukladanie dát využíva zoskupenie zariadení mohla výsledná aplikácia predstavovať základ fungovania jednotky v takomto zoskupení.

Literatúra

[1] Forecasting in Python with Prophet. [Online; navštívené 1.03.2019].
URL https://mode.com/example-gallery/forecasting_prophet_python_cookbook/#data-preparation-and-exploration

[2] Spark Overview. [Online; navštívené 1.05.2019].
URL https://spark.apache.org/docs/2.2.0/streaming-programming-guide.html

[3] Why Striim for manufacturing. [Online; navštívené 1.03.2019].
URL https://www.striim.com/solutions/real-time-data-integration-analytics-manufacturing/

[4] Prophet: forecasting at scale. 2017, [Online; navštívené 1.05.2019].
URL https://facebook.github.io/prophet/

[5] Umelá inteligencia a Priemysel 4.0. *ATP Journal*, 07 2017: s. 290 – 295, [Online; navštívené 1.03.2019].
URL https://www.atpjournal.sk/novetrendy/umela-inteligencia-apriemysel-4.0.html?page_id=25248&from=rss

[6] 2018 Kaggle ML & DS Survey. 2018, [Online; navštívené 1.04.2019].
URL https://www.kaggle.com/kaggle/kaggle-survey-2018

[7] Bokeh. 2018, [Online; navštívené 1.05.2019].
URL https://bokeh.pydata.org/en/1.1.0/

[8] Flask Doumentation. 2018, [Online; navštívené 1.05.2019].
URL http://flask.pocoo.org/docs/1.0/

[9] Pandas. 2018, [Online; navštívené 1.05.2019].
URL https://pandas.pydata.org/about.html

[10] Guide to MySQL and NoSQL - Delivering the Best of Both Worlds. 2019, [Online; navštívené 1.04.2019].
URL https://www.mysql.com/why-mysql/white-papers/guide-to-mysql-and-nosql-delivering-the-best-of-both-worlds/

[11] InfluxDB 1.7 documentation. 2019, [Online; navštívené 1.04.2019].
URL https://docs.influxdata.com/influxdb/v1.7/

[12] Introduction to Dash. 2019, [Online; navštívené 1.04.2019].
URL https://dash.plot.ly/introduction

[13] What is MySQL? 2019, [Online; navštívené 1.04.2019].
URL https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html

[14] What is PostgreSQL? 2019, [Online; navštívené 1.04.2019].
URL https://www.postgresql.org/docs/11/intro-whatis.html

[15] https://conferences.oreilly.com/strata/strata-eu 2018/public/schedule/detail/65353: Audi's journey to an enterprise big data platform. [Online; navštívené 1.03.2019].

[16] Arnanz, R.; A. Gallego de Santiago, M.; Renones Dominguez, A.; aj.: *Monitoring and Fault Diagnosis in Manufacturing Processes in the Automotive Industry*. 01 2011, ISBN 978-953-307-999-8, doi:10.5772/13307.

[17] Baheti, R.; Gill, H.: The Impact of Control Technology: Cyber-physical Systems. *IEEE Control Systems Society*, 2011: s. 1–6, cited By 1.

[18] Caesarendra, W.; Tjahjowidodo, T.: A Review of Feature Extraction Methods in Vibration-Based Condition Monitoring and Its Application for Degradation Trend Estimation of Low-Speed Slew Bearing. *Machines*, ročník 5, 09 2017: str. 21, doi:10.3390/machines5040021.

[19] Deoras, S.: Top 10 Programming Languages For Data Scientists to Learn In 2018. 1 2018, [Online; navštívené 1.04.2019].
URL https://dzone.com/articles/what-is-data-science-programming-top-7-languages

[20] Filová, J. P.: Průmysl 4.0 pohledem stážistů ŠKODA AUTO. *EkonTech.cz: časopis pro studenty techniky a ekonomie*, ročník 8(42), 2019: s. 20–21, ISSN 2336-307X.

[21] Foundation, T. A. S.: HDFS Architecture. [Online; navštívené 1.04.2019].
URL http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html

[22] Guillemin P, F. P.: Internet of things strategic research roadmap, The Cluster of European Research Projects, Tech. Rep., September 2009. [Online; navštívené 1.03.2019].
URL http://www.internet-of-things-research.eu/pdf/IoT_Cluster_Strategic_Research_Agenda_2009.pdf

[23] Hung, M.: Leading the IoT, Gartner Insights on How to Lead in a Connected World, 2017. [Online; navštívené 1.03.2019].
URL https://www.gartner.com/imagesrv/books/iot/iotEbook_digital.pdf

[24] Instruments, N.: The NI TDMS File Format, January 09, 2019. [Online; navštívené 1.03.2019].
URL http://www.ni.com/product-documentation/3727/en/

[25] Laney, D.: The Importance of 'Big Data': A Definition.

[26] Lee, J.: *Industrial Big Data. China: Mechanical Industry Press*. 2015, ISBN 978-7-111-50624-9.

[27] Mourtzis, D.; Vlachou, E.; Milas, N.: Industrial Big Data as a Result of IoT Adoption in Manufacturing. *Procedia CIRP*, ročník 55, 2016: s. 290 – 295, ISSN 2212-8271, doi:https://doi.org/10.1016/j.procir.2016.07.038, 5th CIRP Global Web Conference - Research and Innovation for Future Production (CIRPe 2016).
URL http://www.sciencedirect.com/science/article/pii/S2212827116307880

[28] Patel, N.: Top 6 Languages for Data Science. 8 2018, [Online; navštívené 1.04.2019].
URL https://dzone.com/articles/what-is-data-science-programming-top-7-languages

[29] Reis, M. S.; Gins, G.: Industrial Process Monitoring in the Big Data/Industry 4.0 Era: from Detection, to Diagnosis, to Prognosis. *Processes*, ročník 5, č. 3, 2017, ISSN 2227-9717, doi:10.3390/pr5030035.
URL http://www.mdpi.com/2227-9717/5/3/35

[30] Schaedel, M.: Batch Processing vs. Stream Processing. April 2018. [Online; navštívené 1.04.2019].
URL https://dzone.com/articles/batch-processing-vs-stream-processing

[31] Schatsky, C. G. R., David; Muraskin: Cognitive Technologies: The real opportunities for business. *Deloitte Review*, 2015.
URL http://www2.deloitte.com/tr/en/pages/technology-media-and-telecommunications/articles/cognitive-technologies.html

[32] Teti, R.; Jemielniak, K.; O'Donnell, G.; aj.: Advanced monitoring of machining operations. *CIRP Annals - Manufacturing Technology*, ročník 59, 12 2010: s. 717–739, doi:10.1016/j.cirp.2010.05.010.

[33] Wingerath, F. G. S. F. e. a., Wolfram: Real-time stream processing for Big Data. *Big Data Analytics / Erhard Rahm. it - Information Technology, 58.4*, 2016: s. 186–194, doi:doi:10.1515/itit-2016-0002, retrieved 15 Apr. 2019, from.