VILNIAUS UNIVERSITETAS
INFORMATIKOS INSTITUTAS
PROGRAMŲ SISTEMŲ KATEDRA

Srautinio apdorojimo sistemų balansavimas taikant mašininio mokymosi algoritmus

Balancing stream processing systems using machine learning algorithms

Magistro baigiamajo darbo planas

Atliko:	Vytautas Žilinas	(parašas)
Darbo vadovas:	Partn. Doc. Andrius Adamonis	(parašas)
Recenzentas:	doc. dr. Vardauskas Pavardauskas	(parašas)

Vilnius – 2019

1. Tyrimo objektas ir aktualumas

Šiame darbe yra nagrinėjamas rodiklių duomenų apdorojimas ir kuriamas eksperimentinis sprendimas, galintis prisitaikyti, kai keičiama rodiklių duomenų struktūra. Rodiklių duomenimis vadinkime pasikartojančių įvykių parametrus aprašančius duomenis. Pavyzdžiui, įvairių matuoklių – temperatūros, resursų suvartojimo – fiksuojamus rodmenis, kasmėnesinius veiklos indikatorius, tokius kaip veiklos finansines ataskaitas ar veiklos procesų indikatorius. Taip pat rodiklių struktūra gali būti keičiama laikui bėgant: objektų atributų taksonomija (pvz., mirties priežasčių sąrašas, finansinių sąskaitų sąrašas) arba įrašo atributų sąrašai. Surenkamų rodiklių duomenų kiekis visada didėja, taip pat ir duomenų kiekis, kuriuos reikia apdoroti pagal rodiklius, auga, todėl standartiniai sprendimai, pavyzdžiui, reliacinės duomenų bazės, netinka dėl ilgos apdorojimo trukmės. Rodiklių duomenų bazės pasižymi tuo, kad duomenys į jas patenka iš daug skirtingų tiekėjų ir patekimo laikas tarp tiekėjų nėra sinchronizuojamas, o suagreguotą informaciją vartotojai gali užklausti bet kurio metu. Todėl šiame darbe bus nagrinėjamas srautinis duomenų apdorojimas, kuris patenkančius duomenis apdoros realiu laiku ir saugos jau apdorotus.

Realaus laiko duomenų apdorojimas (angl. Real-time data processing) yra jau senai nagrinėjamas kaip vienas iš būdų apdoroti didelių kiekių duomenis (angl. Big data). Vienas iš realaus laiko apdorojimo sprendimų yra srautinis duomenų apdorojimas. Srautinis duomenų apdorojimas (angl. stream processing) – lygiagrečių programų kūrimo modelis, pasireiškiantis sintaksiškai sujungiant nuoseklius skaičiavimo komponentus srautais, kad kiekvienas komponentas galėtų skaičiuoti savarankiškai [Bea15]. Darbe yra analizuojami jau egzistuojantys srautinio apdorojimo sprendimai pagal srautinio apdorojimo programų savybes aprašytas Michael Stonebraker, pasirenkamas vieną srautinio apdorojimo sprendimas ir su juo daromas eksperimentas.

Kadangi rodikliai laikui bėgant gali būti keičiami, reikia, kad sprendimas galėtų prisitaikyti prie pokyčių. Yra keli būdai kaip tokie sprendimai gali būti kuriami:

- Rankinio atnaujinimo sprendimas. Sukuriamas sprendimas pagal esamus reikalavimus, o atsiradus naujiems reikalavimams būtų kuriamos naujos arba keičiamos esamos apdorojimo sistemos.
- Universalus sprendimas. Sukuriamas universalus parametrizuojamas sprendimas ir pritaikomas užduotims nustatant parametrus.
- Kodo generavimo sprendimas. Sukuriamas sprendimas, kuris generuoja srautinio duomenų apdorojimo sistemas pagal iš anksto aprašytą struktūrą.

Jei nėra numatomas kitimas, pagal ką turi būti apdorojami duomenys, tai galima pasirinkti ir rankinio apdorojimo sprendimą, kadangi nėra didelės tikimybės, kad teks keisti sprendimą. Toks sprendimas tiktų apdorojant išmaniųjų skaitiklių duomenis [Nev17]. Universalus sprendimas taip pat gali būti tinkamas, jei pirminiai rodiklių duomenis yra specifiški ir yra poreikis juos visus apdoroti. Toks sprendimas gali būti aktualus apdorojant duomenis iš sensorių, kurie matuoja namų būseną (temperatūrą, drėgmę ir t.t.) ir bet koks naujas sensorius taip pat turi būti prijungtas ir apdorotas [Yan17].

Pagal Jack Herrington 2003 metų knygą „Code Generation in Action“ kodo generavimas — tai kūrimas programinės įrangos, kuri kurs reikiamą programinę įrangą problemai spręsti. Tai da-

roma tokiais atvejais, kai sprendžiama problema reikalauja daug rankinio darbo, kurį įmanoma automatizuoti. Kuo didesnio sprendimo reikalauja uždavinys, tuo patraukliau tampa naudoti kodo generavimą sprendimo kūrimui. Kodo generavimas suteikia tokius privalumus:

- Architektūrinį nuoseklumą:
  - Verčia programuotojus labiau mąstyti apie architektūrą.
  - Jei sunku „priversti“ generatorių generuoti reikiamą kodą, problema gali būti architektūroje.
  - Geros dokumentacijos buvimas sumažina problemą, kai nariai palieka projektą.
- Abstrakciją:
  - Programuotojai galės kurti naujus šablonus, kurie leis esamą funkcionalumą pritaikyti kitomis kalbomis, sprendimams daug paprasčiau negu rankomis parašytą kodą.
  - Verslo analitikai galės apžvelgti ir patvirtinti sprendimo abstrakciją.
  - Abstrakcija padės paprasčiau paruošti dokumentaciją, testavimo atvejus, produkto palaikymo medžiagą ir t.t.
- Aukštą komandos moralę — rašomas kodas bus nuoseklus ir kokybiškas, todėl kels komandos pasitikėjimą savimi.
- Tinkamas sprendimas judriajam (angl. agile) programavimui, kadangi kodo generavimo kuriami sprendimai yra lankstesni, tai leidžia ateityje juos lengviau keisti ir atnaujinti.

Kodo generavimas tampa tikrai naudingas tada, kai jis naudojamas didelių kiekių rankiniam kodavimui pakeisti [Her03].

Darbe nagrinėjami architektūra ir eksperimentinis sprendimas, generuojantis srautinio apdorojimo sistemas pagal rodiklių duomenų modelius. Rodiklių duomenis apdorojantis eksperimentinis sprendimas turi pasižymėti tokiomis savybėmis:

- Srautinių apdorojimo sistemų generavimas pagal deklaratyvų aprašymą.
- Galimybė keisti esamas apdorojimo sistemas, kai pakeičiamas rodiklių duomenų modelis.
- Išvestinių rodiklių gavimas iš daugiau nei vieno rodiklio transformacijos.
- Išvestinių rodiklių apdorojimas pagal iš anksto apibrėžtas funkcijas.

1.1. Darbo tikslas

Ištirti mašininio mokymosi tinkamumą srautinio apdorojimo sistemų balansavimui.

1.2. Uždaviniai

1. Pasrinkti srautinio apdorojimo sistemos metrikas, kurios bus naudojamos balansavimui.
2. Išanalizuoti ”Reinforcement Learning” algoritmus ir pasirinkti tinkamą tyrimui.
3. Pritaikyti pasirinktą mašininio mokymosi algoritmą srautinio apdorojimo sistemos balansavimui.
4. Atlikti eksperimentą ir palyginti rezultatą su alternatyvomis - standartinė konfiguracija, reaktyvus balansavimas.

1.3. Laukiami rezultatai

1. Pritaikant skirtingas apkrovas srautinio apdorojimos sistemoms, surinktos ir surikiuotos metrikos, reikalingos ”Heron” srautinio apdorojimo sistemos balansavimui.
2. Pasirinktas ”Reinforcement Learning” mašininio mokymosi algoritmas.
3. Pasirinktas mašininio mokymosi algoritmas pritaikytas ”Heron” srautinio apdorojimo sistemų balansavimui.
4. Atliktas eksperimentas naudojant sukurtą balansavimo implementaciją ir srautinio apdorojimo sistemos dirbtines apkrovas.
5. Eksperimento rezultatai palyginti su standartine konfiguracija ir reaktyviu balansavimu.

Literatūra

[Bea15] Jonathan Beard. A short intro to stream processing. `http://www.jonathanbeard.io/blog/2015/09/19/streaming-and-dataflow.html`, 2015-09.

[Her03] Jack Herrington. *Code generation in action*. Manning Publications Co., 2003.

[Yan17] Shusen Yang. Iot stream processing and analytics in the fog. *IEEE Communications Magazine*, 55(8):21–27, 2017.

[Nev17] Mantas Neviera. Išmaniųjų apskaitų didelių duomenų kiekių apdorojimas modernioje duomenų apdorojimo architektūroje, 2017.