VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMŲ STUDIJŲ PROGRAMA

Srautinio apdorojimo sistemų balansavimas taikant mašininį mokymąsi

Balancing stream processing systems using machine learning

Mokslo tiriamasis darbas III

Atliko:	Vytautas Žilinas	(parašas)
Darbo vadovas:	Andrius Adamonis	(parašas)
Recenzentas:	Prof. dr. Aistis Raudys	(parašas)

Vilnius – 2021

TURINYS

Įvadas

Realaus laiko duomenų apdorojimas (angl. real–time data processing) yra jau senai nagrinėjamas kaip vienas iš būdų apdoroti didelių kiekių duomenis (angl. Big data). Viena iš didelių duomenų apdororojimo tipinių architektūrų yra srautinis 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].

Yra keli pagrindiniai srautinio apdorojimo varikliai: „Apache Storm“, „Apache Spark“, „Heron“ ir kiti. „Apache Storm“ ir „Heron“ apdoroja duomenis duomenų srautais, o „Apache Spark“ mikro–paketais [KKW+15]. „Heron“ srautinio apdorojimo variklis, buvo išleistas „Twitter“ įmonės 2016 metais kaip patobulinta alternatyva „Apache Storm“ srautinio apdorojimo varikliui [Ram16]. Šiame darbe bus naudojamas „Heron“, kadangi tai yra naujesnis ir greitesnis srautinio apdorojimo variklis nei „Apache Storm“ [KBF+15].

Srautinio apdorojimo sistemų balansavimas (angl. auto–tuning) – tai sistemos konfigūracijos valdymas siekiant užtikrinti geriausią resursų išnaudojimą – duomenų apdorojimas neprarandant greičio, bet ir naudojant tik reikiamą kiekį resursų. Kadangi srautinio apdorojimo sistemų komponentai yra kuriami kaip lygiagretus skaičiavimo elementai, todėl jie gali būti plečiami horizontaliai ir vertikaliai [Bea15] keičiant sistemų konfigūraciją. Tačiau lygiagrečių elementų kiekio keitimas nėra vienintelis būdas optimizuoti resursų išnaudojimą. Kiekvienas variklis turi savo rinkinį konfigūruojamų elementų. Pavyzdžiui, darbe naudojamas „Heron“ variklis leidžia optimizuoti sistemas naudojant 56 konfigūruojamus parametrus [Her19].

Yra skirtingi būdai kaip gali būti parenkama tinkama konfigūracija. Kadangi srautinio apdorojimo sistemų apkrovos gali būti skirtingų pobūdžių (duomenų kiekis, skaičiavimų sudėtingumas, nereguliari apkrova), o inžinieriai kurdami ir konfigūruodami taikomasias sistemas išbando tik kelis derinius ir pasirenka labiausiai tinkanti [FA17], lieka daug skirtingų neišbandytų konfigūracijos variacijų. Optimalios konfigūracijos suradimas yra NP sudėtingumo problema [SSP04], kadangi žmonėms yra sunku suvokti didelį kiekį konfigūracijos variacijų. Vienas iš būdų automatiškai valdyti konfigūraciją buvo pasiūlytas 2017 metų straipsnyje „Dhalion: self–regulating stream processing in heron“, kuriame autoriai aprašo savo sukurtą sprendimą „Dhalion“, kuris konfigūruoja „Heron“ srautinio apdorojimo sistemas pagal esamą apkrova ir turimus resursus, tai yra jei apdorojimo elementų išnaudojimas išauga virš 100%, „Dhalion“ padidina lygiagrečiai dirbančių apdorojimo elementų kiekį [FAG+17]. Tačiau toks sprendimas leidžia reguliuoti tik elementų lygiagretumą

ir tai daro tik reaktyviai.

Vienas iš naujausių būdų balansuoti srautinio apdorojimo sistemas – mašininis mokymasis. Vienas iš tokių bandymų aprašytas 2018 metų straipsnyje „Auto–tuning Distributed Stream Processing Systems using Reinforcement Learning“[VC18] kuriame atliktas tyrimas – „Apache Spark“ sistemos balansavimui naudojamas skatinamojo mokymo REINFORCE algoritmas, kuris, pagal dabartinę konfigūraciją ir renkamas metrikas, keitė srautinio apdorojimo sistemos konfigūracijos parametrus. Šiame tyrime pasiūlytas sprendimas, naudojantis mašininį mokymąsi, suranda efektyvesne konfigūraciją per trumpesnį laiką nei žmonės, o tokiu būdu išskaičiuotą konfigūraciją naudojanti sistema pasiekia 60–70% mažesnį vėlinimą, nei naudojanti ekspertų rankiniu būdu nustatytą konfigūraciją. [VC18]. Šiame darbe naudojamas „Heron“ variklis leidžia prie savęs prijungti sukurtą išorinę metrikų surinkimo programą, kuri gali rinkti tokias sistemų metrikas kaip: naudojama RAM atmintis, CPU apkrova, komponentų paralelizmas ir kitas, kurios gali būti naudojamos balansavimui.

Skatinamasis mokymasis yra vienas iš mašininio mokymosi tipų. Šis mokymasis skiriasi nuo kitų, nes nereikia turėti duomenų apmokymui, o programos mokosi darydamos bandymus ir klysdamos. Vienas iš pagrindinių privalumų naudojant skatinamąjį mokymąsi balansavimui – nereikia turėti išankstinių duomenų apmokymui, kas leidžia jį paprasčiau pritaikyti skirtingoms srautinio apdorojimo sistemų apkrovoms. Tačiau tokio tipo mašininis mokymasis turi ir problemų: sudėtinga aprašyti tinkamos konfigūracijos apdovanojimo (angl. reward) funkciją ir balansą tarp tyrinėjimo ir išnaudojimo tam, kad nebūtų patiriami nuostoliai [FA17].

Yra sukurta daug skatinamojo mokymosi algoritmų (Monte Carl, Q–learning, Deep Q Network ir kiti), šiame darbe jie apžvelgti, pasirinkti ir pritaikytas išsikeltam uždaviniui.

Magistro darbo tikslas: Ištirti mašininio mokymosi tinkamumą srautinio apdorojimo sistemų balansavimui.

Magistro darbo uždaviniai:

1. Sudaryti srautinio apdorojimo sistemų balansavimo modelį ir nustatyti valdymo metrikas ir jų siekiamas reikšmes, kurios bus naudojamos eksperimentinėje sistemoje.
2. Parinkti skatinamojo mokymosi algoritmą eksperimentui, atsirenkant iš algoritmų, aprašomų literatūroje.
3. Sukurti eksperimentinį sprendimą su pasirinktu algoritmu ir atlikti eksperimentus.
4. Palyginti eksperimento rezultatus su alternatyvomis – „Heron“ su standartine konfigūracija bei „Heron“ balansavimas pritaikius REINFORCE algoritmą.

Antroje magistro darbo dalyje atlikta literatūros analizė, sudarytas balansavimo modelis ir parinkti

skatinamojo mokymosi algoritmai, o šiame mokslo tyriamajame darbe – trečio magistro darbo dalyje – siekta tikslo : apibrėžti darbo metodą ir sukurti bei pagrįsti algoritmą, kuris bus naudojamas eksperimentui. Ir įgyvendinti šie uždaviniai:

1. Apibrėžti algoritmą srautinio apdorojimo sistemų balansavimui naudojanti Deep Q Network ir Soft Actor Critic skatinamojo mokymosi algoritmus.
2. Apibrėžti eksperimento eigą ir altenatyvius sprendimus, kurių rezultatai bus naudojami algortimo įvertinimui.

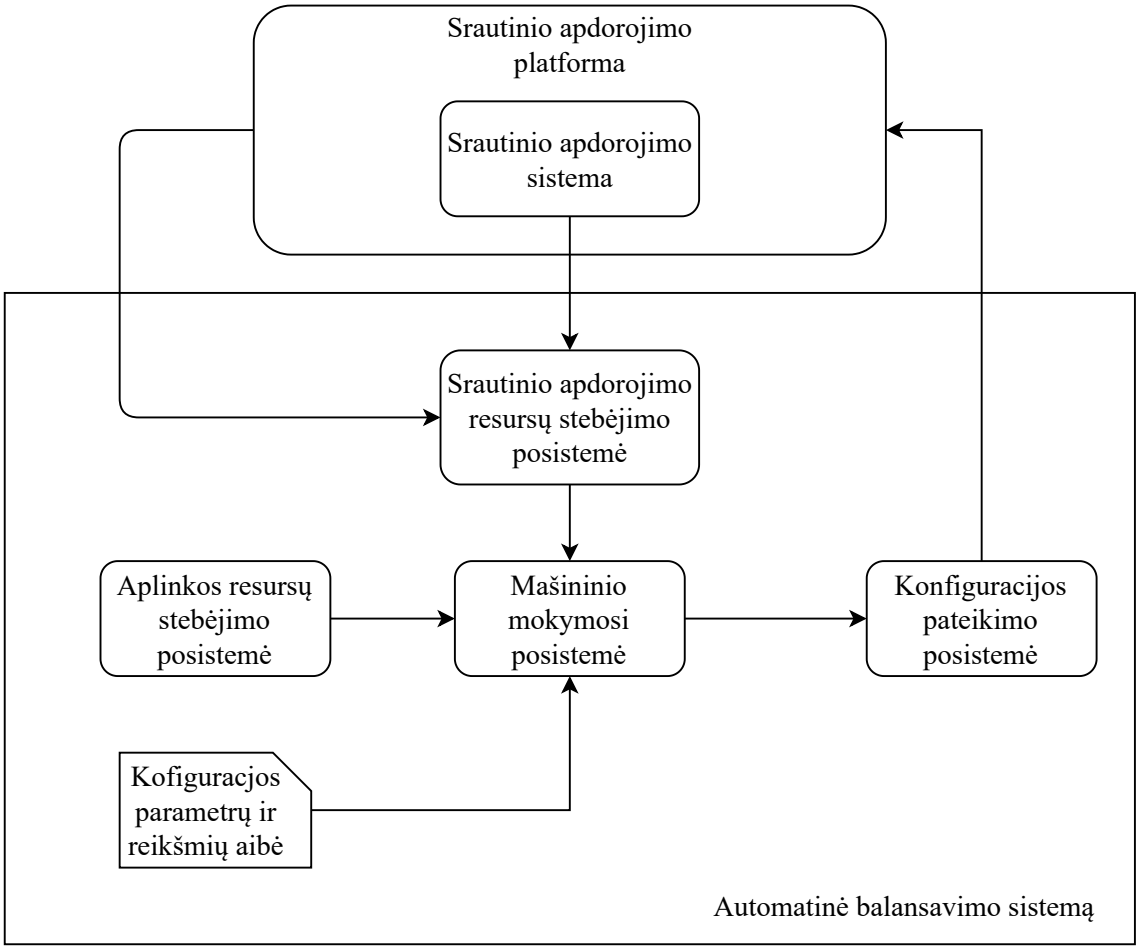
1. Srautinės architektūros sistemos, valdomos grįžtamuuoju ryšiu, modelis

2. Optimizavimo algoritmas

3. Eksperimento tyrimo planas

4. Srautinio apdorojimo sistemų balansavimo algoritmas

Šiame skyriuje aptariamas algoritmas, kuris bus naudojamas srautinių sistemų balansavimui.



1 pav. Pagrindiniai algoritmo elementai ir jų tarpusavio ryšiai

4.1. Tikslas

Algoritmo tikslas savarankiškai reguliuoti srautinio apdorojimo sistemos konfiguracijos parametrus siekiant palaikyti sistemą stablią ir optimaliai naudojančia resursus. 1 pav. pavazduota sistema yra integruojama į srautinio apdorojimo sistemos platformą iš kurios ji gauna reikiamą informacija apie srautinio apdorojimo sistemos metrikas ir resursus, bei turėti informaciją apie esamus aplinkos naudojamus ir turimus resursus. Balansavimo algoritmas eksperimente bus implementuojamas kaip posistemė, kuri gauna metrikas iš srautinio apdorojimo sistemos platformos ir operacinės sistemos, o atnaujintą konfiguraciją pateikia į srautinio apdorojimo platformą per komandų įvedimo eilutę.

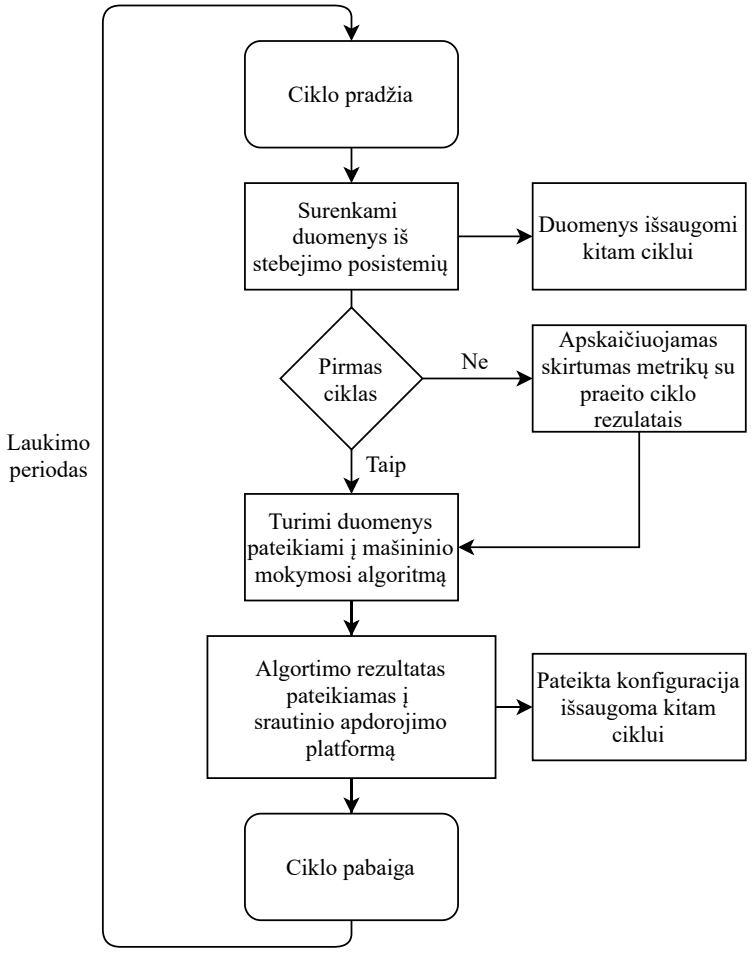
4.2. Įeiga

Kad mašinio mokymosi posistemė galėtų daryti sprendimus, jai paduodami šie duomenys:

- Srautinio apdorojimo sistemos naudojami resursai ir metrikos – procesoriaus apkrova, operatyviosios atminties apkrova, priešslėgis (angl. backpressure), srautinio apdorojimo sistemos įeigos komponento atsilikimas nuo duomenų srauto.
- Aplinkos naudojami resursai – procesoriaus apkrova, operatyviosios atminties apkrova.
- Srautinio apdorojimo sistemos pradinė konfiguracija – komponentų lygiagretumo parametrai, konteinerių parametrai, konfiguracijos elementai ir jų reikšmės.
- Konfiguruojami parametrai ir jų reikšmių ribos.
- Mašinio mokymosi algoritmo konfiguracija – sprendimų priemimo periodiškumas.
- Buvusio ciklo pradiniai duomenys ir priimti sprendimai.

4.3. Eiga

Sistema naudojanti mašininį mokymąsi veikia pastoviai vienodo periodo ciklais (2 pav.).



2 pav. Algortimo veikimas

Tarpas tarp ciklų apibrežiamas leidžiant balansavimo sistemai ir yra skirtas srautinio apdorojimo sistemai atsinaujinti su naujais konfiguracijos parametrais bei kad surinkti duomenys būtų pakankamai svarūs pateikti į mašininį mokymosi algoritmą. Tarp ciklų taip pat yra renkamos metrikos apie būseną, kurie bus veliau paduodami mašininio mokymosi algoritmui.

4.4. Algoritmo rezultatas

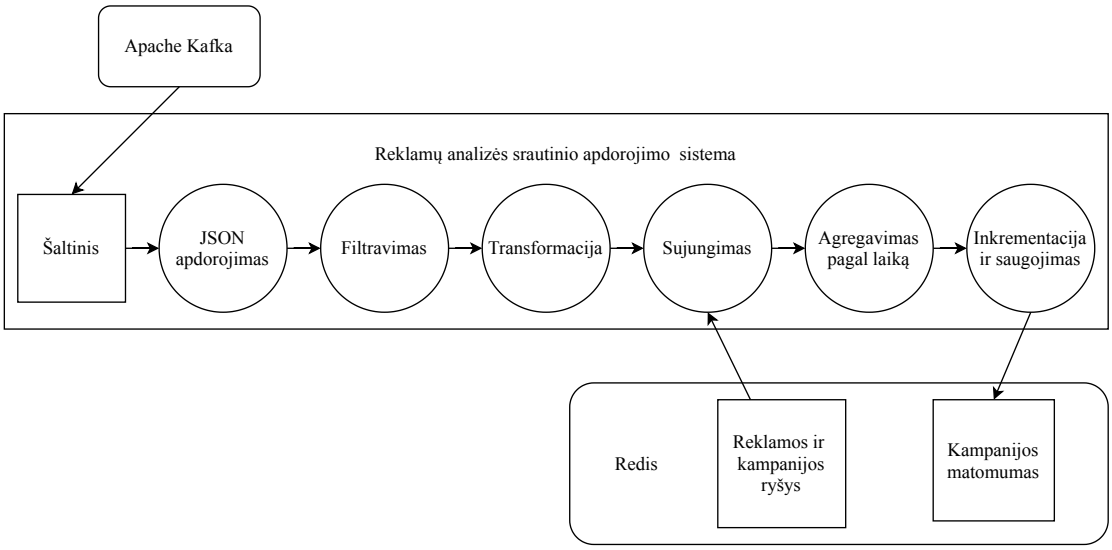
Mašinio mokymosi posistemė, atlikusi skaičiavimus, gražiną naują konfiguracijos parametrų rinkinį, kuris yra veliau pateikiamas į srautinio apdorojimo sistemų platformą. Po srautinės apdorojimo sistemos pradedamas naujas ciklas sistemos stebėjimo ir naujų konfiguracijos parametrų kurimo, kuris taip pat atsižvelgia į metrikų skirtumo ir konfiguracijos pakeitimo rezultatus iš ankstesnių ciklų.

5. Alternatyvus sprendimai

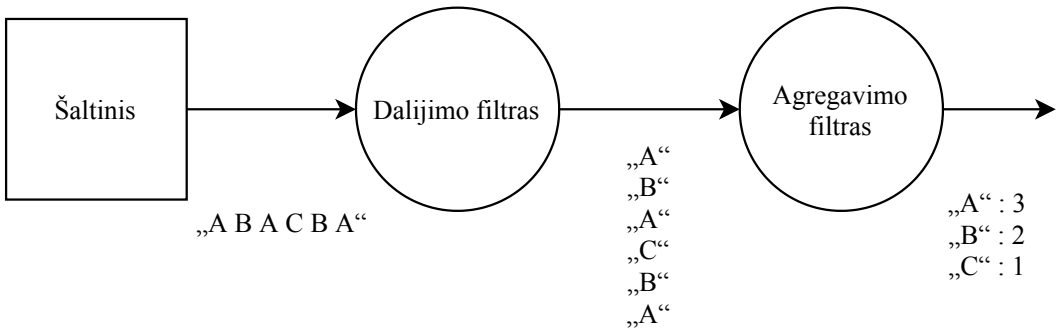
Siekiant užtikrinti sprendimo validumą ir efektyvumą bus atliekamas eksperimentas ir rezultatai bus lyginamį su alternatyviais sprendimais.

5.1. Rezultatų surinkimas

Pagal literatūros analizę atlikta MTDII matoma, kad dauguma autorių renkasi vertinti tik pagal vieną metriką ir dažniau matavimui naudojamas vėlinimas (angl. latency). Taip pat [VC18], naudojantis skatinamąjį mokymą, matavimui naudoja vėlinimą ir [CDE+16] straipsnis, kuris siūlo srautinio apdorojimo sistemų vertinimo sprendimą, naudoja vėlinimą. Todėl eksperimente rezultatai bus vertinami naudojant vėlinimą.



3 pav. Reklamų analizės sistema [CDE+16]



4 pav. WordCount sistemos pavyzdys

Taip pat pagal MTDII literatūros analizę tyrimai bus atliekami su Reklamų analizės siste-

ma (3 pav.), kadangi ši sistema sukurta srautinių apdorojimo variklių vertinimui ir su WordCount srautinio apdorojimo sistemą (4 pav.), kuri neturi pašalinių elementų sistemoje, kadangi Reklamų analizės sistema naudoja „Apache Kafka“, „Redis“ vertinimui. Kadangi šios abi sistemos turi savo duomenų generavimo komponentus todėl jos bus naudojamos kaip tyrimo duomenis. Reklamų analizės sistema taip pat turi ir posisteme, kuri apskaičiuoją vėlinimą ir saugo jį tekstiniame įrašę, o WordCount sistemai įvertinti bus naudojama pačios Heron platformos įrašai siekiant kuo mažiau daryti įtakos rezultatams.

Išvados

Literatūra

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

[CDE+16] S. Chintapalli, D. Dagit, B. Evans, R. Farivar ir k.t. Benchmarking streaming computation engines: storm, flink and spark streaming. *2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, p. 1789–1792, 2016.

[FA17] Avrilia Floratou ir Ashvin Agrawal. Self-regulating streaming systems: challenges and opportunities. *Proceedings of the International Workshop on Real-Time Business Intelligence and Analytics*, BIRTE '17, 1:1–1:5, Munich, Germany. ACM, 2017. ISBN: 978-1-4503-5425-7. DOI: 10.1145/3129292.3129295. URL: http://doi.acm.org/10.1145/3129292.3129295.

[FAG+17] Avrilia Floratou, Ashvin Agrawal, Bill Graham, Sriram Rao ir Karthik Ramasamy. Dhalion: self-regulating stream processing in heron. *Proceedings of the VLDB Endowment*, 10:1825–1836, 2017-08. DOI: 10.14778/3137765.3137786.

[Her19] Heron. Heron documentation on cluster configuration. http://heron.incubator.apache.org/docs/cluster-config-overview/, 2019.

[KBF+15] Sanjeev Kulkarni, Nikunj Bhagat, Maosong Fu, Vikas Kedigehalli, Christopher Kellogg, Sailesh Mittal, Jignesh M. Patel, Karthik Ramasamy ir Siddarth Taneja. Twitter heron: stream processing at scale. *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, SIGMOD '15, p. 239–250, Melbourne, Victoria, Australia. ACM, 2015. ISBN: 978-1-4503-2758-9. DOI: 10.1145/2723372.2742788. URL: http://doi.acm.org/10.1145/2723372.2742788.

[KKW+15] Holden Karau, Andy Konwinski, Patrick Wendell ir Matei Zaharia. *Learning spark: lightning-fast big data analysis*. " O'Reilly Media, Inc.", 2015.

[Ram16] Karthik Ramasamy. Open sourcing twitter heron, 2016.

[SSP04] David G. Sullivan, Margo I. Seltzer ir Avi Pfeffer. Using probabilistic reasoning to automate software tuning. *SIGMETRICS Perform. Eval. Rev.*, 32(1):404–405, 2004-06. ISSN: 0163-5999. DOI: 10.1145/1012888.1005739. URL: http://doi.acm.org/10.1145/1012888.1005739.

[VC18] Luis M. Vaquero ir Felix Cuadrado. Auto-tuning distributed stream processing systems using reinforcement learning, 2018. arXiv: 1809.05495 [cs.DC].