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). 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].

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. Darbe naudojamas „Heron“ variklis leidžia optimizuoti sistemas naudojant 56 konfigūruojamus parametrus [Her19].

Yra skirtingi būdai kaip gali būti parenkama tinkama konfigūracija. Kadangi dar nėra naudojimui paruoštų sprendimų, kurie galėtų balansuoti srautinio apdorojimo sistemas savarankiškai, dažniausiai už tai yra atsakingi inžinieriai, kurie dirba su šiomis sistemomis. Kadangi srautinio apdorojimo sistemų apkrovos gali būti skirtingų pobūdžių (duomenų kiekis, skaičiavimų sudėtingumas, nereguliari apkrova), o inžinieriai konfigūruodami išbando tik kelis derinius ir pasirenka labiausiai tinkanti [FA17], lieka labai 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, t.y. jei apdorojimo elementų iš-

naudojimas išauga >100%, „Dhalion“ padidina lygiagrečiai dirbančių apdorojimo elementų kiekį [FAG+17]. Tačiau šis 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“, 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 nustatyta, jog sprendimas, naudojantis mašininį mokymąsi, suranda optimalesnę konfigūraciją per trumpesnį laiką nei žmones ir taip pat surastą konfigūraciją naudojančios srautinio apdorojimo sistemos vėlinimas (angl. latency) yra 60–70% mažesnis nei tyrimo metu ekspertų derinamos konfigūracijos [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. Pagrindinis uždavinys naudojant skatinamąjį mokymąsi – surasti balansą tarp naujų sprendimų tyrinėjimo (angl. exploration) ir turimos informacijos išnaudojimo (angl. exploitation) [JOA10]. 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 bus apžvelgti ir vienas iš jų bus pasirinktas ir pritaikytas išsikeltam uždaviniui.

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

Numatomi magistro darbo uždaviniai:

1. Sudaryti srautinio apdorojimo sistemų balansavimo modelį ir nustatyti valdymo metrikas irjų 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, „Heron“ su „Dhalion“ priedu bei „Heron“ balansavimas pritaikius REINFORCE algoritmą.

Šio darbo tikslas ir uždaviniai

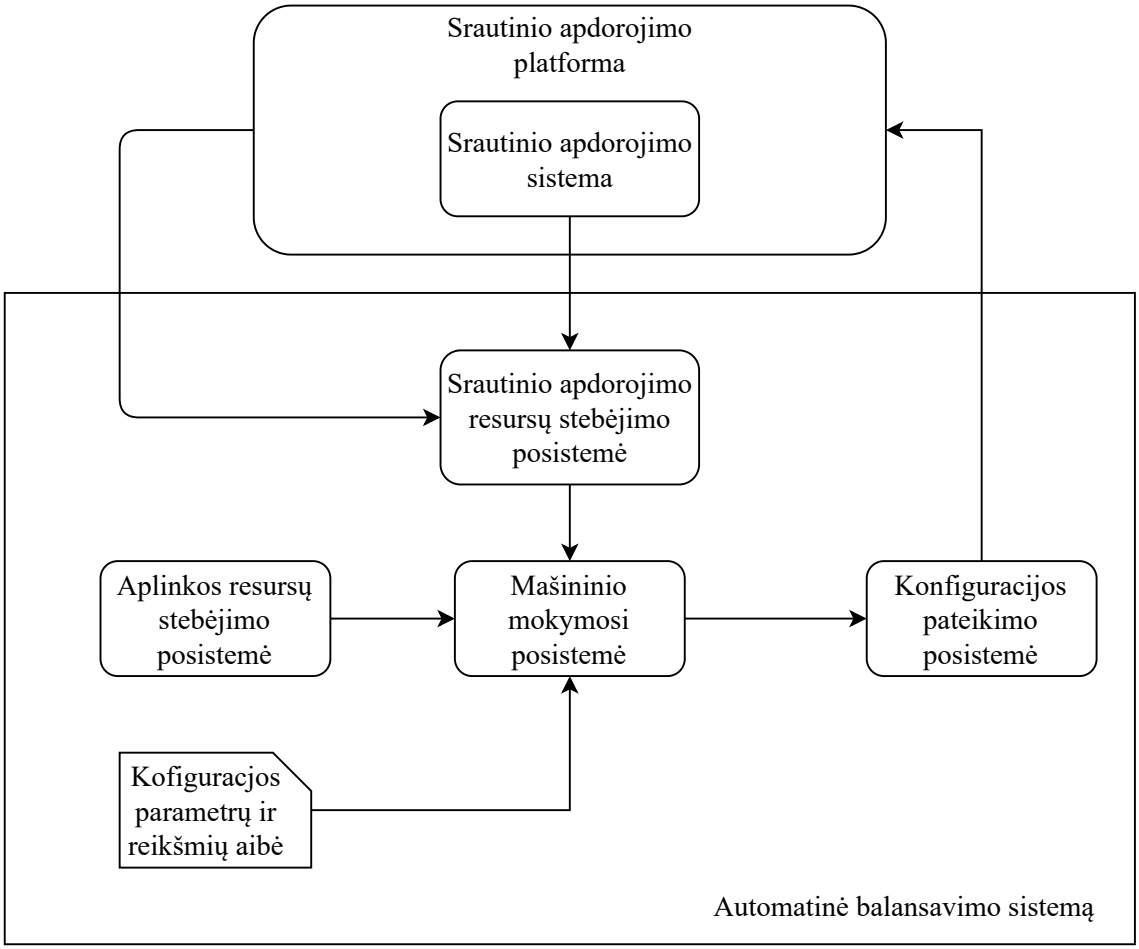
Tikslas: apibrėžti darbo metodą ir sukurti bei pagrįsti algoritmą, kuris bus naudojamas eksperimentui.

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. Srautinio apdorojimo sistemų balansavimo algoritmas

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



1 pav. Pagrindiniai algoritmo elementai ir jų tarpusavio ryšiai

1.1. 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.

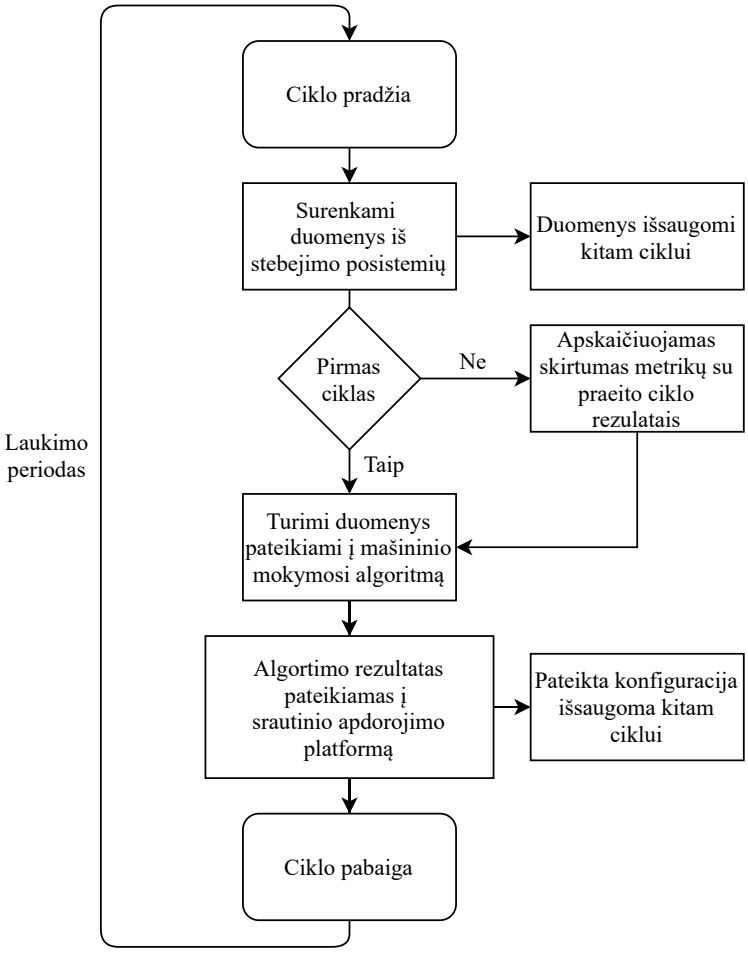
1.2. Įeiga

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.

1.3. 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.

1.4. 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.

1.5. Algoritmui tikrinti naudojami skatinamojo mokymosi algoritmai

Šiame skyriuje bus aprašomi skatinamojo mokymosi algoritmai, kurie bus naudojami eksperimentui. Šie algoritmai pasirinkti pagal MTD II literatūros analizę, tačiau pats balansavimo algoritmas yra nepriklausomas nuo naudojamo skatinamojo mokysimosi algortimų ir gali būti įgyvendintas naudojant kitus algoritmus. Pasirinkti algortimai skyriasi veiksmo erdve – Deep Q Network atlieka skaičiavimus su diskrečia veiksmu erdve, o Soft Actor Critic su tęstinę veiksmų erdve.

1.6. Skatinamasis mašininis mokymasis

Skatinamasis mokymasis yra algoritmais priklausantys mašininio mokymosi algoritmų grupei. Mašininis mokymasi algoritmai - algoritmai, kurie kuria matematinius modelius paremtus mokymosi duomenimis ir darinčius sprendimus, kurie nėra suprogramuoti [Bis]. Mašininis mokymasis neseniai išpopuliarėjo dėko poreiko dirbti su dideliais kiekiais nerušiuotų ir nestrukturizuotų duomenų ir dėl namų kopiuterių galios padidėjimo iki lygio, kuris leidžia spręsti mašininio mokymosi skaičiavimus [TEA].

Standartiniame skatinamojo mašininio mokymosi algoritme agentas (angl. agent) yra prijungtas prie aplinkos (angl. environment) per stebėjimus(angl. perception) ir veiksmus(angl. action). Kiekvieną žingsnį agentas atlieką veiksmą ir to veiksmo vertė yra perduodama agentui per skatinamąjį signalą. Agentas turi rinktis veiksmus, kurie per ilgą laiko tarpą didins veiksmo įvertčius. Tai pasiekiamą per tam tikrą laiko tarpą systematiškai atliekant bandymus ir klystant, su

papildoma algoritmu pagalba siekiant papdidinti efektyvumą ir sprendimų stabilumą [KLM96].

1.7. Deep Q Network algoritmas

Deep Q Network yra Q Learning implementacija naudojant giliuosius neuroninius tinklus. Q Learning - skatinamojo mokymosi algortimas, kuris bet kokiam baigtiniam Markovo pasirinkimo procesui randa optimalius sprendimus maximizuojančius galutinio rezultato gavimą per bet kokį kiekį žingsnių pradedant nuo esamos būsenos [Mel01]. Q Learning naudoja Q funkciją, kurios įeigą yra būsenos ir veiksmo kombinacija, o rezultatas yra atlygio apromiksimacija. Pradžioje visos Q reikšmės yra 0 ir algoritmas atlikdamas veiksmus pildo lentelę atnaujintaumis reikšmėmis. Tačiau, kai veiksmų ir būsenų pasidaro per daug Q Learning algortimo nebeužtenka ir tenka naudoti giliuosius neuroninius tinklus.

Deep Q Network skyriasi nuo Q Learning tuo, kad vietoj būsenos ir veiksmų į jį paduodama būseną ir jis gražina Q reikšmę visų įmanomų veiksmų. Taip pat Deep Q Network naudoja patirties pakartojimą (angl. expirience replay) - vietoj to, kad kurti sprendimą pagal paskutinį veiksmą yra paduodamas rinkinys atsitiktinių veiksmų pagal kurį algoritmas gali efektyviau mokytis.

Deep Q Network algoritmo veikimas[Cho19]:

1. Perduoti aplinkos būseną į Deep Q Network, kuris gražins visus įmanomus veiksmus būsenai.
2. Pasirinkti veiksmą naudojant ϵ-greedy strategiją, kuriuo metu pasirenkamas atsitiktinis veiksmas arba pasirenkamas veiksmas turintis didžiausia Q reikšmę.
3. Įvykdomas veiksmas ir pereinama į naują būseną. Šis perėjimas išsaugomas kaip patirties pakartojimo kortežas susidarantis iš būsenos, veiksmo, atlygio ir naujos būsenos.
4. Pasirenkamas atsitiktinis rinkints perėjimų iš patirties paraktojimo kortežų rinkinio ir apskaičiuojamas nuostolis (angl. loss).

$$Loss = (r + \gamma max_{a'} Q(s',a'; \theta') - Q(s,a; \theta))^2$$

5. Atlikti gradiento nusileidimą su tikrais tinklo parametrais siekiant sumažinti nuostolį.
6. Po kiekvienos iteracijos, perkelti tikrojo tinklo svorius į pasirinkto tinklo svorius.
7. Visi žingsniai kartojami iki nustatytos pabaigos.

1.8. Soft Actor Critic algortimas

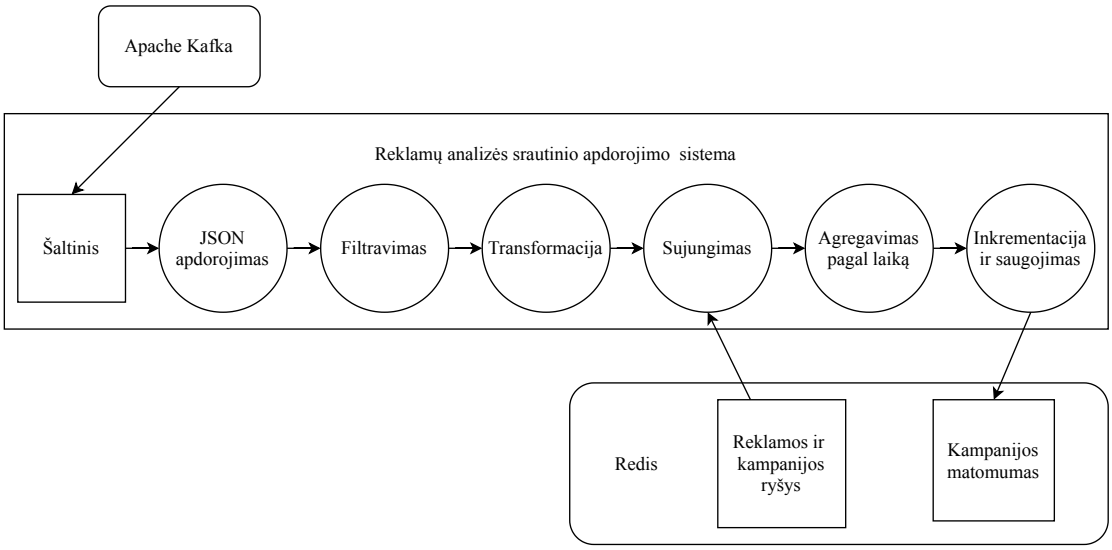
Soft Actor Critic skiriasi nuo kitų skatinamojo mokymosi algoritmų tuo kad jis bando ne tik pasiekti didžiausią bendrą atlygį, tačiau ir didinti strategijos entropiją. Entropijos didinimas reiškia, kad Soft Actor Critic algoritmas yra labiau linkęs į tyrinėjimą (angl. exploration) nei kiti algoritmai, ko pasekoje užtikrinama, kad algoritmas nesirinks visada tokio pačio veiksmo.

2. Alternatyvus sprendimai

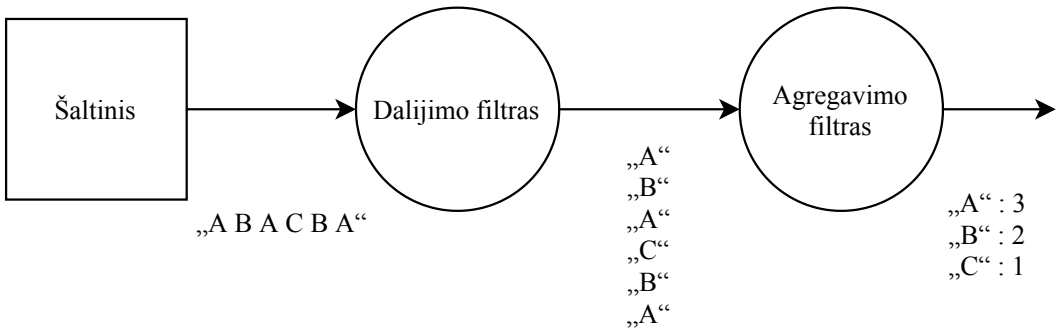
Siekiant užtikrinti sprendimo validumą ir efektyvumą bus atliekamas eksperimentas ir rezultatai bus lyginamį su alternatyviais sprendimais.

2.1. 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.

2.2. Standartinė konfiguracija

Sukurto sprendimo rezultatai bus lyginami su standartine konfiguracija tik su pakeistu skaičiavimo komponentų lygiagretumu atsižvelgiant į įrangos skaičiavimo gijų kiekį. Šiuo rezultatu bus tikrinama, ar Apache Heron pats negali taip pat arba geriau susitvarkyti su įeinančiu duomenų kiekiu nedidinant vėlinimo. Standartinės konfiguracijos naudojami balansavimo sprendimai [KBF+15]:

- Srautinės apdorojimo sistemos priešslėgis (angl. backpressure) – kai pastebimas jog tam tikras skaičiavimo kompoenentas nespėja susidoroti su ateinančiu duomenu kiekiu ir praneša prieš tai einantiems skaičiavimo komponentams lėtinti duomenų patekimo greiiti. Heron yra implementuotas šaltinio priešslėgis, kuris veikia kaip buferis ant kiekvienos jungties tarp komonentų ir kai buferis prisipildo iki tam tikro taško įjungiamas priešslėgio režimas, kuris sulėtina srautinio apdorojimo sistemos greiti iki lėčiausio skaičiavimo komponento greičio.
- Šiukšių surinktuvo (angl. garbage collector) optimizacija - kai srautinio apdorojimo sistema gaudavo dideli kiekį duomenų vienu metu ir užsipildydavo visa turima sistemos atmintis, kiekvieną karta gaunant nauja įraša buvo paleidžiamas šiukšlių surinktuvas, kuris naudoja daug resursų. Kad išspręsti šią problemą Heron periodiškai tikrina srautinio apdorojimo sistemos atminties talpą ir patenkančių duomenų dydį ir jeigu duomenų dydis pasidaro didesnis nei talpa tai duomenų paėmimo greitis sumažinamas per pusę ir taip kartojama kol pasiekiamas stabilus greitis.
- Resursų rezervacija - kiekvienas konteineris (komponentų rinkinys) turi jam priskrita resursų kiekį ir prieš palaiedžiant srautinio apdorojimo sistema Heron rezervuoja reikiamus resursus ir jeigu komponentai bando pasiekti daugiau resursu negu jiems rezervuota yra lėtinama sistema taip užtikrinant stabilumą [FMK+15].

2.3. Balansavimas naudojant REINFORCE algoritmą

Sukurtas sprendimas naudojantis Deep Q network ir Soft Actor Critic taip pat bus lyginamas su [VC18] aprašytų REINFORCE algortimu. Staripsnis nagrinėja automatinį balansavimą srautinio apdorojimo sistemų Apache Spark platformoje. Straipsnyje nagrinėjamas sprendimas susidaro iš trijų sistemų:

1. Sistema, kurioje už anksto sugeneruoti konfiguracijų deriniai leidžiami srautinio apdorojimo sistemose ir surenkamos metrikos bei konfiguracijos įverčiai. Gauti duomenis analyzuojami naudojant Factor Analysis + k-means ir gaunamas sąrašas pagrindinių metrikų bei konfiguracijos elementai darantys daugiausiai įtakos greitaveikai. Ši sistema naudojama vieną kartą prieš leidžiant sekančią sistemą.
2. Sistema, kurioje surinktos metrikos ir konfiguracijos elementai yra leidžiami iš naujo ir naudojant Lasso path analizę konfiguracijos elementų sąrašas surušiuojamas pagal įtaką greitaveikai. Tai daroma siekiant statistiškai užtikrinti, jog pasirinkti konfiguracijos elementai yra įtakingiausi. Ši sistema naudojama vieną kartą prieš leidžiant sekančią sistemą.
3. Pagrindinė mašininio mokymosi sistema, naudojanti REINFORCE algoritmą, kuri naudodamas surikiuotų konfiguracijos elementų sąrašą ir pagrindinių metrikų sąraša periodiškai atnaujina konfiguraciją. Ši sistema paleidžiama tuo pačiu metu kaip ir srautinio apdorojimo sistema ir veikia visą laiką, kol paleistas ekspermientas.

Eksperimentinis sprendimas buvo sukonfiguruotas kas 5 minutes atnaujinti vieną konfiguracijos elementą. Autoriams pavyko pasiekti 70% sumažinta vėlinimmą po 50 minučių ir mokymas pilnai konvergavo po 11 valandų. Magistro darbe bus naudojamas supaprastintas eksperimentas: bus implementuojamas REINFORCE algoritmas ir lyginamas su Deep Q Network ir Soft Actor Critic algoritmais.

3. Eksperimentas

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.

[Bis] CM Bishop. Pattern recognition and machine learning (information science and statistics)(springer-verlag new york, inc., secaucus, nj, usa, 2006).

[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.

[Cho19] Ankit Choudhary. A hands-on introduction to deep q-learning using openai gym in python, 2019.

[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.

[FMK+15] Maosong Fu, Sailesh Mittal, Vikas Kedigehalli, Karthik Ramasamy ir k.t. Streaming@ twitter. *IEEE Data Eng. Bull.*, 38(4):15–27, 2015.

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

[JOA10] Thomas Jaksch, Ronald Ortner ir Peter Auer. Near-optimal regret bounds for reinforcement learning. *J. Mach. Learn. Res.*, 11:1563–1600, 2010-08. ISSN: 1532-4435. URL: http://dl.acm.org/citation.cfm?id=1756006.1859902.

[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.

[KLM96] Leslie Pack Kaelbling, Michael L. Littman ir Andrew W. Moore. Reinforcement learning: A survey. *CoRR*, cs.AI/9605103, 1996. URL: https://arxiv.org/abs/cs/9605103.

[Mel01] Francisco S Melo. Convergence of q-learning: a simple proof. *Institute Of Systems and Robotics, Tech. Rep*:1–4, 2001.

[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.

[TEA] DATAFLAIR TEAM. Why is machine learning so popular? https://data-flair.training/blogs/why-machine-learning-is-popular/.

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