

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	3
1. SRAUTINIO APDOROJIMO SISTEMŲ BALANSAVIMO ALGORITMAS	6
1.1. Tikslas	6
1.2. Įeiga.....	7
1.3. Eiga	7
1.4. Algoritmo rezultatas	8
1.5. Algoritmui tikrinti naudojami skatinamojo mokymosi algoritmai	8
1.6. Skatinamasis mašininis mokymasis	8
1.7. Deep Q Network	9
1.8. Soft Actor Critic.....	9
2. ALTERNATYVUS SPRENDIMAI	10
2.1. Rezultatų surinkimas.....	10
2.2. Standartinė konfigūracija	11
2.3. Balansavimas naudojant REINFORCE algoritmą.....	12
3. EKSPERIMENTAS	13
IŠVADOS	14
LITERATŪRA	15

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 vertikalčiai [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 apkrovą 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 trumpesnę laiką nei žmonės 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 sukurta 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 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, „Heron“ su „Dhalion“ priedu bei „Heron“ balansavimas pritaikius REINFORCE algoritmą.

Šio darbo tikslas ir uždaviniai

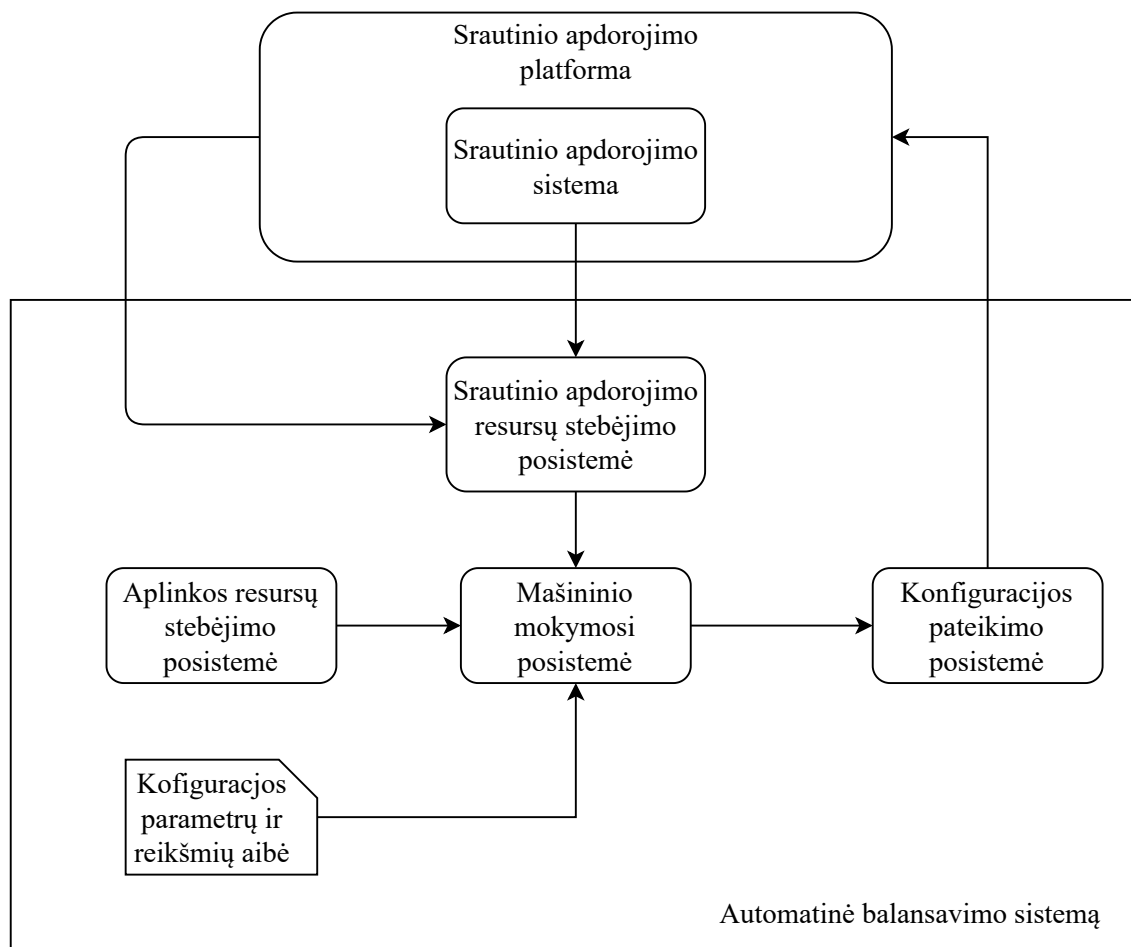
Tikslas: apibrėžti darbo metodą ir sukurti bei pagrįsti algoritmą, kuris bus naudojamas eksperimentui.

Uždaviniai:

1. Apibrėžti algoritmą srautinio apdorojimo sistemų balansavimui naudojami Deep Q Network ir Soft Actor Critic skatinamojo mokymosi algoritmus.
2. Apibrėžti eksperimento eigą ir alternatyvius 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 konfigūracijos parametrus siekiant palaikyti sistemą stabilią ir optimaliai naudojančią resursus. 1 pav. pavazduota sistema yra integruojama į srautinio apdorojimo sistemos platformą iš kurios ji gauna reikiamą informaciją 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 konfigūraciją pateikia į srautinio apdorojimo platformą per komandų įvedimo eilutę.

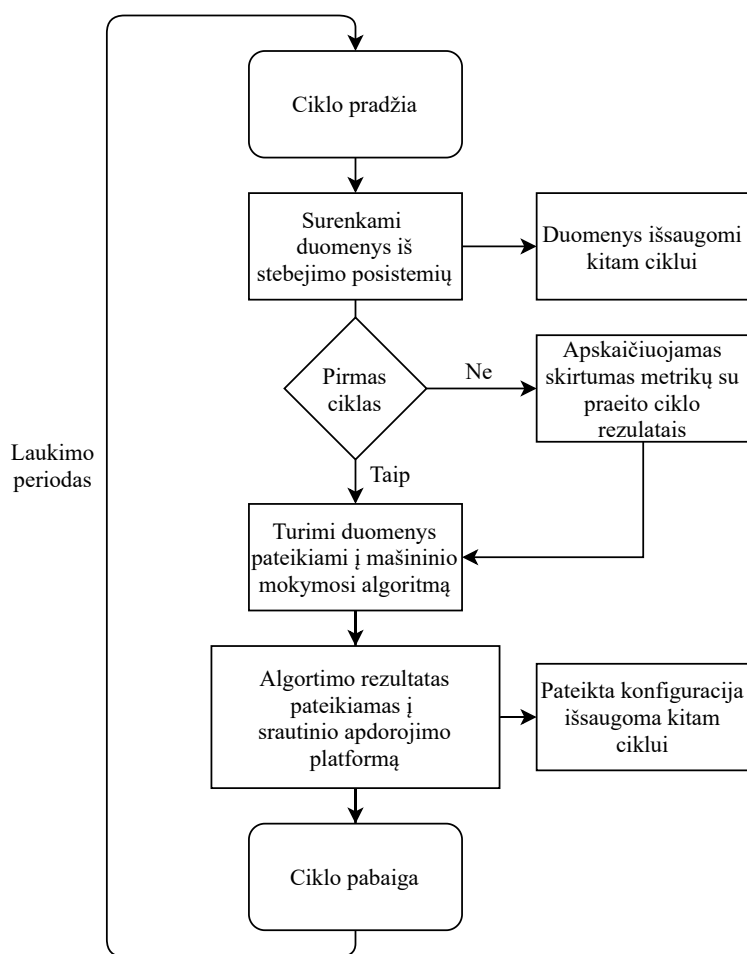
1.2. Įeiga

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 konfigūracija - komponentų lygiagretumo parametrai, konteinerių parametrai, konfigūracijos elementai ir jų reikšmės.
- Konfigūruojami parametrai ir jų reikšmių ribos.
- Mašinio mokymosi algoritmo konfigūracija - sprendimų priėmimo periodiškumas.
- Buvusio ciklo pradiniai duomenys ir priimti sprendimai.

1.3. Eiga

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



2 pav. Algoritmo veikimas

Tarpas tarp ciklų apibrežiamas leidžiant balansavimo sistemai ir yra skirtas srautinio apdorojimo sistemai atsinaujinti su naujais konfigūracijos parametrais bei kad surinkti duomenys būtų pakankamai svarūs pateikti į mašininių mokymosi algoritmą.

1.4. Algoritmo rezultatas

Mašinio mokymosi posistemė, atlikusi skaičiavimus, gražina naują konfigūracijos parametrų rinkinį, kuris yra vėliau pateikiamas į srautinio apdorojimo sistemų platformą. Po srautinės apdorojimo sistemos pradedamas naujas ciklas sistemos stebėjimo ir naujų konfigūracijos parametrų kurimo, kuris taip pat atsižvelgia į metrikų skirtumo ir konfigūracijos 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 mokymosi algoritmo ir gali būti įgyvendintas naudojant kitus algoritmus.

1.6. Skatinamasis mašininis mokymasis

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

Standartiniame skatinamojo mašinio mokymosi algoritme agentas (angl. agent) yra prijungtas prie aplinkos (angl. environment) per stebėjimus (angl. perception) ir veiksmus (angl. action). Kiekvieną žingsnį agentas atlieka veiksmą ir to veiksmo vertė yra perduodama agentui per skatinamąjį signalą. Agentas turi rinktis veiksmus, kurie per ilgą laiko tarpą didins veiksmo įverčius. Tai pasiekiamą per tam tikrą laiko tarpą sistematiškai atliekant bandymus ir klystant, su papildoma algoritmu pagalba siekiant padidinti efektyvumą ir sprendimų stabilumą [KLM96].

1.7. Deep Q Network

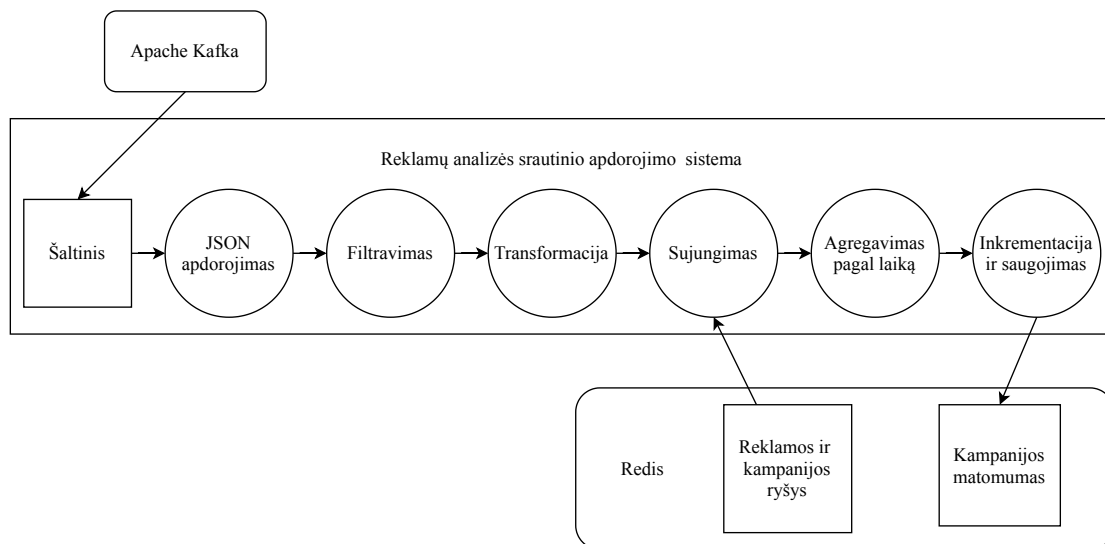
1.8. Soft Actor Critic

2. Alternatyvus sprendimai

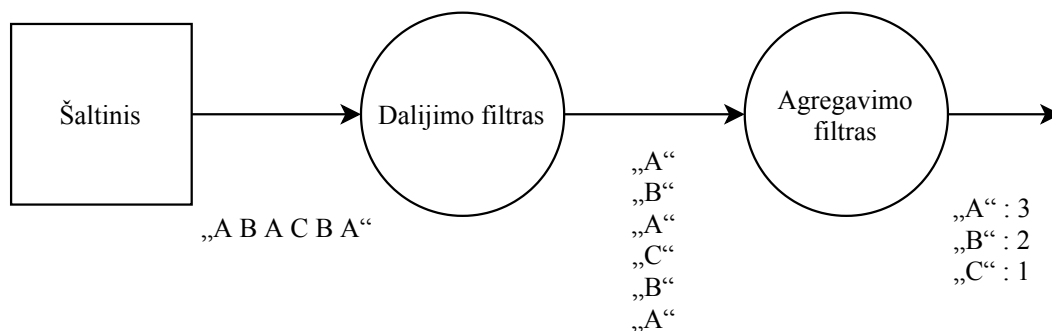
Siekiant užtikrinti sprendimo validumą ir efektyvumą bus atliekamas eksperimentas ir rezultatai bus lyginami su alternatyviais sprendimais.

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 posistemę, kuri apskaičiuoja vėlinimą ir saugo jį tekstiniam įrašui, o WordCount sistemai įvertinti bus naudojama pačios Heron platformos įrašai siekiant kuo mažiau daryti įtakos rezultatams.

2.2. Standartinė konfigūracija

Sukurto sprendimo rezultatai bus lyginami su standartine konfigūracija 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 konfigūracijos naudojami balansavimo sprendimai [KBF⁺15]:

- Srautinės apdorojimo sistemos priešslėgis (angl. backpressure) – kai pastebimas jog tam tikras skaičiavimo komponentas nespėja susidoroti su ateinančiu duomenų kiekiu ir praneša prieš tai einantiems skaičiavimo komponentams lėtinti duomenų patekimo greitį. Heron yra implementuotas šaltinio priešslėgis, kuris veikia kaip buferis ant kiekvienos jungties tarp komponentų ir kai buferis prisipildo iki tam tikro taško įjungiamas priešslėgio režimas, kuris sulėtina srautinio apdorojimo sistemos greitį iki lėčiausio skaičiavimo komponento greičio.
- Šiukšlių surinktuvo (angl. garbage collector) optimizacija - kai srautinio apdorojimo sistema gaudavo dideli kiekį duomenų vienu metu ir užsipildydavo visa turima sistemos atmintis, kiekvieną kartą gaunant naują įrašą 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 priskirtą resursų kiekį ir prieš paleidžiant srautinio apdorojimo sistemą Heron rezervuoja reikiamus resursus ir jeigu komponentai bando pasiekti daugiau resursų 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] aprašytą REINFORCE algortimu. Straipsnis nagrinėja automatinį balansavimą srautinio apdorojimo sistemų Apache Spark platformoje. Straipsnyje nagrinėjamas sprendimas susidaro iš trijų sistemų:

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

Eksperimentinis sprendimas buvo sukonfigūruotas kas 5 minutes atnaujinti vieną konfigūracijos elementą. Autoriams pavyko pasiekti 70% sumažinta vėlinimą 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.
- [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>.
- [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].