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 I

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

Vilnius – 2019

1. Tyrimas

1.1. Tyrimo aktualumas ir naujumas

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. „Heron“ srautinio apdorojimo variklis, kuris 2016 metais buvo išleistas „Twitter“, kaip alternatyva „Apache Storm“ srautinio apdorojimo varikliui [Ram16]. O „Apache Spark“ skiriasi nuo „Apache Storm“ ir „Heron“ tuo, kad jis apdoroja duomenis ne srautais, o mikro-paketais [KKW+15]. Šiame darbe bus naudojamas „Heron“, kadangi tai yra naujesnis ir geresnis srautinio apdorojimo variklis nei „Apache Storm“ [KBF+15]. Taip pat jo duomenų apdorojimo būdas yra kitoks nei „Apache Spark“, kurio balansavimas jau buvo atliktas naudojant mašininį mokymąsi [VC18].

Srautinio apdorojimo sistemų balansavimas (angl. auto-tuning) - tai sistemos konfigūracijos valdymas siekiant užtikrinti geriausią resursų išnaudojimą - duomenų apdorojimas neprarandant greičio, bet ir tuo pačiu naudojant tik reikiamą kiekį resursų. Kadangi srautinio apdorojimo sistemos yra kuriamos lygiagrečios, todėl jos gali būti plečiamos horizontaliai ir vertikaliai [Bea15], keičiant sistemų konfigūraciją. Tačiau lygiagrečių skaičiavimo vienetų keitimas nėra vienintelis būdas optimizuoti resursų išnaudojimą, kiekvienas variklis turi savo rinkinį konfigūruojamų elementų. Pavyzdžiui „Heron“ leidžia valdyti 56 skirtingus konfigūruojamus svertus (angl. levers) [Her19].

Yra skirtingi būdai kaip gali būti parenkama tinkama konfigūracija. Kadangi nėra dar naudojimui pritaikytų sprendimų, kurie balansuotų srautinio apdorojimo sistemos dažniausiai tai daro duomenų inžinieriai, kurie dirba su šiomis sistemomis. Kadangi apkrovos gali būti skirtingų pobūdžiu (duomenų kiekis, skaičiavimų sudėtingumas, nereguliari apkrova), o inžinieriai konfigūruodami išbando keletą derinių ir pasirenka labiausiai tinkanti [FA17], lieka labai daug skirtingų neišbandytų konfigūracijos variacijų. Optimalios konfigūracijos suradimas yra NP sudėtingumo problema [SSP04], todėl žmonėms yra sunku suvokti didelį kiekį konfigūracijos variacijų. Kitas būdas valdyti konfigūraciją buvo pasiūlytas 2017 metų straipsnyje „Dhalion: self-regulating stream processing in heron“, kur autoriai sukūrė sprendimą, kuris leidžia konfigūruoti „Heron“ srautinio apdorojimo sistemas pagal dabartinę situaciją, t.y. jei „Dhalion“ mato, jog apdorojimo elementų išnaudojimas yra >100%, jis padidina lygiagrečiai dirbančių apdorojimo elementų kiekį [FAG+17]. Tačiau šis sprendimas leidžia reguliuoti tik elementų lygiagretumą ir tik reaguoti į resursų pokyčius. Taip pat balansavimui gali būti pritaikytas mašininis mokymasis. Vienas iš tokių bandymų buvo aprašytas 2018 metų straipsnyje „Auto-tuning Distributed Stream Processing Systems using Reinforcement Learning“, kuriame buvo atliktas tyrimas: „Apache Spark“ sistemos balansavimui buvo

naudojamas skatinamojo mokymo REINFORCE algoritmas, kuris pagal „Apache Spark“ duodamas sistemos metrikas keitė konfigūracijas ir buvo nustatyta, jog sprendimas naudojantis mašininį mokymąsi greičiau suranda geresne konfigūraciją nei žmogus per trumpesnį laiką ir taip pat sumažina laukimo trukme (angl. latency) 60-70% palyginus su žmonių pasirinkta konfigūracija [VC18]. Šiame darbe naudojamas „Heron“ variklis leidžia prie savęs prijungti sukurta išorinę metrikų surinkimo programą, kuri gali rinkti tokias sistemų metrikas kaip: naudojama RAM atmintį, CPU apkrovą, komponentų paralelizmas ir kitas.

Skatinamasis mokymasis yra vienas iš mašininio mokymosi sričių. Tai mokymasis, kuris nereikalauja duomenų apmokymui, o mokosi darant bandymus ir klystant ir pagrindinis uždavinys yra surasti balansą tarp naujų sprendimų tyrinėjimo (angl. exploration) ir turimos informacijos išnaudojimo (angl. exploitation) [JOA10]. Vienas iš pagrindinių privalumu naudojant skatinamąjį mokymąsi balansavimui yra tai, kad nereikia turėti išankstinių duomenų apmokymui. Tačiau tokio tipo mašininis mokymasis turi ir problemų: sudėtinga aprašyti teisingo sprendimo apdovanojimo funkciją ir turi būti teisingai aprašytas balansas tarp tyrinėjimo ir išnaudojimo [FA17].

Yra sukurta daug skatinamojo mokymosi algoritmų (Monte Carl, Q-learning, Deep Q Network ir kiti), tad šiame darbe jie bus apžvelgti ir vienas iš jų bus pasirinktas ir pritaikytas išsikeltam uždaviniui. Algoritmas bus pasirinktas pagal tai, kuris bus tinkamiausias srautinių apdorojimų sistemų balansavimui.

2. Darbas

2.1. Darbo tikslas

Ištirti mašininio mokymosi tinkamumą srautinio apdorojimo sistemų balansavimui.

2.2. Uždaviniai

1. Pasirinkti srautinio apdorojimo sistemos metrikas, kurios bus naudojamos balansavimui.
2. Išanalizuoti esamus skatinamojo mokymosi algoritmus ir pasirinkti bei pritaikyti tinkamą algoritmą srautinio apdorojimo sistemos balansavimui.
3. Atlikti eksperimentą ir palyginti rezultatą su alternatyvomis - „Heron“ su standartine konfigūracija, „Heron“ su „Dhalion“ priedu bei „Heron“ balansavimas pritaikius REINFORCE algoritmą.

2.3. Laukiami rezultatai

1. Pritaikant skirtingas apkrovas srautinio apdorojimo sistemoms, surinktos ir surikiuotos metrikos, reikalingos „Heron“ srautinio apdorojimo sistemos balansavimui.
2. Pasirinktas skatinamojo mokymosi algoritmas Deep Q Network ir pritaikytas „Heron“ srautinio apdorojimo sistemų balansavimui.
3. Atliktas eksperimentas su „Heron“ ir alternatyvomis ir išanalizuoti bei palyginti gauti rezultatai.

Literatūra

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

[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.

[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.

[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].