VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMŲ STUDIJŲ PROGRAMA

Srautinio apdorojimo sistemų balansavimas taikant skatinamąjį mokymąsi

Balancing stream processing systems using reinforcement learning

Magistro baigiamasis darbas

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

Vilnius – 2021

Santrauka

TODO

Raktiniai žodžiai: srautinis apdorojimas, mašininis mokymasis, skatinamasis mašininis mokymasis, ”Heron”, ”Dhalion”, balansavimas

Summary

TODO

Keywords: stream processing, machine learning, reinforcement learning, ”Heron”, ”Dhalion”, balancing

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 apdorojimo 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 taikomąsias 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 apkrovą 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 atlygio (angl. reward) funkciją ir balansą tarp tyrinėjimo ir išnaudojimo tam, kad nebūtų patiriami nuostoliai [FA17].

Yra sukurta daug skatinamojo mokymosi algoritmų (Monte Carlo, Q–learning, Deep Q Network ir kiti), šiame darbe yra apžvelgiami algoritmai, kuriuos naudojo kiti autoriai savo tyrimuose susijusiuose su srautinio apdorojimo sistemų veikimo gerinimu ir pasirenkami keli iš jų siekiant patikrinti ar skatinamasis mokymasis yra tinkamas srautinių sistemų balansavimui ir kuris skatinamojo mokymosi algoritmas pasiekia geriausius rezultatus po tam tikro kiekio apmokymo. Darbe naudojamas REINFORCE skatinamojo mokymosi algoritmas, ir Deep Q network bei Actor–Critic with Experience Replay giliojo skatinamojo mokymosi algoritmai ir bus tiriama kaip kiekvienas iš jų pasirodo su vienodu kiekiu apmokymo.

Tikslas: Ištirti skatinamojo mokymosi tinkamumą srautinio apdorojimo sistemų balansavimui.

Uždaviniai:

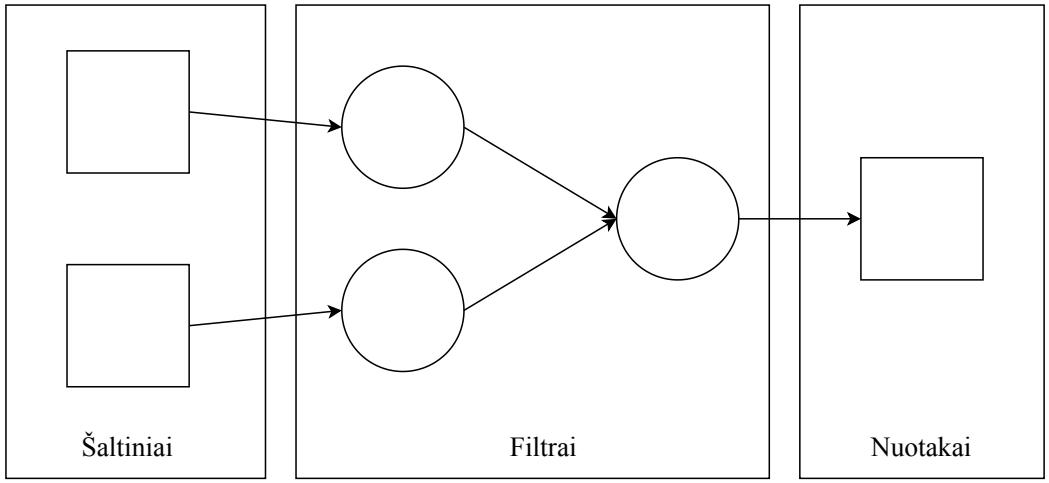
1. Sudaryti srautinio apdorojimo sistemų balansavimo modelį ir nustatyti valdymo metrikas ir jų siekiamas reikšmes, kurios bus naudojamos eksperimentinėje sistemoje.

2. Parinkti srautinę apdorojimo sistemą ir skatinamojo mokymosi algoritmus eksperimentui, atsirenkant iš algoritmų ir sistemų, aprašomų literatūroje.
3. Sukurti eksperimentinį sprendimą su pasirinkta srautinio apdorojimo sistema ir atlikti eksperimentus su skirtingais skatinamojo mokymosi algoritmais.
4. * Gal šito iš vis nereikia * Palyginti eksperimento rezultatus su alternatyvomis – „Heron“ su standartine konfigūracija * ar reikia –> * bei „Heron“ balansavimas rankiniu būdu.

1. Srautinio apdorojimo sistemų matavimas ir derinimas

1.1. Srautinio apdorojimo sistemos

Srautinis duomenų apdorojimas (angl. stream processing) – terminas naudojamas apibrėžti sistemas sudarytas iš skaičiavimo elementų (angl. modules) galinčių skaičiuoti lygiagrečiai ir kurios bendrauja kanalais. Tokių sistemų elementai dažniausiai skirstomi į tris klases: šaltinius (angl. sources), kurie paduoda duomenis į sistemą, filtrus (angl. filters), kurie atlieka tam tikrus vienetinius (angl. atomic) skaičiavimus ir nuotakus (angl. sink), kurie perduoda duomenis iš sistemų [Ste97].



1 pav. Srautinio apdorojimo sistemos pavyzdys

Srautinio apdorojimo sistemos literatūroje yra vaizduojamos orientuotais grafikais (1 pav.). Srautinio apdorojimo sistemos skiriasi nuo reliacinio modelio šiais aspektais [BBD+02]:

- Duomenys į sistemą patenka tinklu, o ne iš fizinių talpyklų.
- Duomenų patekimo tvarka negali būti kontroliuojama.
- Duomenų kiekis yra neapibrėžtas.
- Duomenys apdoroti srautinio apdorojimo sistema yra pašalinami arba archyvuojami, t.y. juos pasiekti yra sunku.

1.1.1. Duomenų vykdymas

Srautinio apdorojimo sistemų veikimui reikalingas srautinio apdorojimo variklis (angl. stream processing engine). Šie varikliai yra skirti srautinio apdorojimo sistemų vykdymui, dislokavimui, plečiamumo (angl. scaling) užtikrinimui ir gedimų tolerancijai (angl. fault–tolerance)

[ZGQ+17]. Populiariųjų srautinio apdorojimo variklių pavyzdžiai: „Apache Storm“, „Apache Heron“, „Apache Spark“, „Apache Samza“ ir t.t [RM19]. Duomenų vykdymas gali būti išskaidytas į tris elementus [ZGQ+17]:

- Planavimas (angl. scheduling) – duomenų apdorojimo užduočių planavimas daro įtaką bendram srautinio apdorojimo sistemos veikimui [FY11]. Pavyzdžiui, „Apache Samza“ naudoja „Apache YARN“ resursų valdymo sistemą, kuri turi planavimo posistemę, kuri skirsto resursus [NPP+17]
- Plečiamumas (angl. scalability) – apibrėžia daug apdorojimo branduolių turinčios sistemos gebėjimą apdoroti didėjanti kiekį užduočių ir galimybę didinti pačią sistemą, kad ji galėtų susidoroti su didėjančiu kiekiu duomenų [Bon00]. Srautinio apdorojimo varikliai turi užtikrinti srautinio apdorojimo sistemų plečiamumą [SÇZ05].
- Išskirstytas skaičiavimas (angl. Distributed computation) – tarpusavyje nesusiję skaičiavimo elementai turi naudotojui atrodyti kaip viena darni sistemą [TV07]. Srautinio apdorojimo varikliai turi užtikrinti darbų paskirstymą ir skaičiavimo įrenginių koordinaciją, kad kuo daugiau duomenų būtų apdorojami vienu metu [ZGQ+17].

Srautinio apdorojimo sistemos turi viena pagrindinį elementą – srauto procesorių (angl. stream processor), kuris apibrėžia sistemos elementus, aprašo kaip šie sistemos elementai sujungti ir pateikia nustatymus elementams [ZGQ+17]. Pavyzdžiui, „Apache Storm“ šis elementas vadinimas „topology“, kuris yra užrašomas Java kalba, naudojant „Apache Storm“ pateiktą biblioteką [IS15].

1.1.2. Duomenų priėmimas

Į srautinio apdorojimo sistemą duomenys patenka per šaltinius, kurie šiuos duomenis perduoda tolimesniems elementams. Dažniausiai duomenis perduodami į sistemą naudojant žinučių eiles (angl. message queues), nes jos turi buferi, kuris leidžia mažinti greičių skirtumus tarp duomenų gavimo ir duomenų apdorojimo ir žinučių eilių brokeriai gali išfiltruoti duomenis ir nukreipti juos į tinkamus šaltinius [KF16]. Tačiau šaltiniai turi turėti galimybę rinkti išsaugotus duomenis ir priimti ateinančius naujus duomenis [SÇZ05], todėl, nors ir šaltiniai dažniausiai skirti priimti srautinius duomenis, jie turi taip pat gebėti naudoti duomenis iš talpyklų [ZGQ+17].

1.2. Srautinio apdorojimo sistemų matavimas

Svarbiausias srautinio apdorojimo sistemų reikalavimas – duomenų apdorojimas ir rezultatų grąžinimas negali turėti atsilikimo – didelių apimčių srautiniai duomenys turi būti apdorojami taip

pat greitai kaip jie ateiną [SÇZ05].

1.2.1. Srautinio apdorojimo sistemų metrikos

Pagrindinės kitų autorių naudojamos metrikos:

- Pralaidumas (angl. Throughput) – per tam tikrą laiko tarpą apdorojamų įvykių kiekis.
- Vėlinimas (angl. Latency) – laiko intervalas nuo apdorojimo arba įvykio pradžios iki apdorojimo pabaigos.

Vėlinimas ir pralaidumas dažniausiai nepriklauso vienas nuo kito – sistemos, apdorojančios srautus mikro–paketais, turi didesnį pralaidumą, tačiau atsiranda papildomas vėlinimas, kol laukiama duomenų paketo apdorojimo pradžios [KRK+18].

[SÇZ05] straipsnyje minima, jog srautinio apdorojimo sistemos naudotojas turi išbandyti savo sistemą su tiksliniu darbo krūviu ir išmatuoti jos pralaidumą ir vėlinimą prieš naudodamas ją realiomis sąlygomis. [KRK+18] lygina srautinio apdorojimo variklius ir matavimui naudoja vėlinimą, kurį išskaido į įvykio vėlinimą (angl. event–time latency) – laiko intervalas nuo įvykio laiko iki rezultato gavimo iš srautinio apdorojimo sistemos ir apdorojimo vėlinimą (angl. processing–time latency) – laiko intervalas nuo duomens patekimo į srautinio apdorojimo sistemą iki rezultato grąžinimo. Autoriai atlieka šį skaidymą, nes sistemų vertinime dažnai ignoruojamas įvykio laikas ir rezultatuose gaunamas daug mažesnis vėlinimas, nei tikras. Taip pat autoriai išskiria darnų pralaidumą (angl. sustainable throughput) – didžiausia apkrova įvykių, kurią sistema gali apdoroti be pastoviai augančio įvykio vėlinimo, todėl savo eksperimentuose autoriai užtikrina, kad duomenų generavimo greitis atitiktų sistemos darnų pralaidumą. Kad sužinoti darnų pralaidumą sistemos autoriai pradžioje leidžia labai didelį srautą duomenų ir mažina jį kol sistemos apdorojimas susivienodina su generavimo greičiais. Visus vėlinimo rezultatus autoriai pateikia maksimalaus pralaidumo apdorojimo ir 90% pralaidumo apdorojimo vidurkiais, minimumais, maksimumais ir kvantiliais (90, 95, 99). [HSS+14] autoriai nagrinėja srautiniam apdorojimui galimas optimizacijas ir matavimui naudoja normalizuotą pralaidumą (naudojamas vienetas kaip vidurkis), kadangi tai leidžia lengviau palyginti santykinę greitaveiką. Taip pat, [HSS+14] pastebi, nors ir yra daug metrikų, kuriomis galima matuoti optimizacijos efektus: pralaidumas, vėlinimas, paslaugos kokybė (angl. quality of service), energijos ir tinklo panaudojimas, tačiau dažniausiai pagerinus pralaidumą pagerėja ir visos kitos metrikos. [QWH+16] srautinių apdorojimo sistemų matavimui naudoja pralaidumą (skaičiuojama baitais per sekundę) ir vėlinimą, kaip vidurkį nuo duomens patekimo į sistemą iki apdorojimo pabaigos. Taip pat, kadangi autoriai lyginą srautinio apdorojimo variklius,

jie įveda metriką gedimų toleravimo (angl. fault tolerance) matavimui – išjungiamas tam tikras kiekis elementų ir matuojamas pralaidumas ir vėlinimas. [ZYL+20] palyginimui naudoja sistemos įvykdymo vėlinimą (angl. system completion latency), kuris rodo vidutinį laiko tarpą per kurį duomuo nukeliauja nuo šaltinio iki sistemos galutinio taško. Autoriai skaičiavo vidutinį laiką 5 sekundžių intervalais. Taip pat autoriai matavimui naudoja kiekvienos instancijos (angl. instance) CPU apkrovą, kiekvieno darbinio mazgo (angl. worker node) CPU apkrovą ir apkrovą tarp instancijų/mazgų, kadangi [ZYL+20] užduotis – patobulinti esamą planavimo posistemę. [FAG+17] matavimui naudoja pralaidumą per minutę. [VC18] tyria labai panašią problemą – srautinių apdorojimo sistemų balansavimą taikant skatinamąjį mokymą ir matavimui naudoja vėlinimo 99 kvantilį. [CDE+16] srautinio apdorojimo variklių vertinimo tyrimui naudoja vėlinimą.

1 lentelė. Metrikos naudojamos tiriant srautinio apdorojimo sistemų greitaveiką

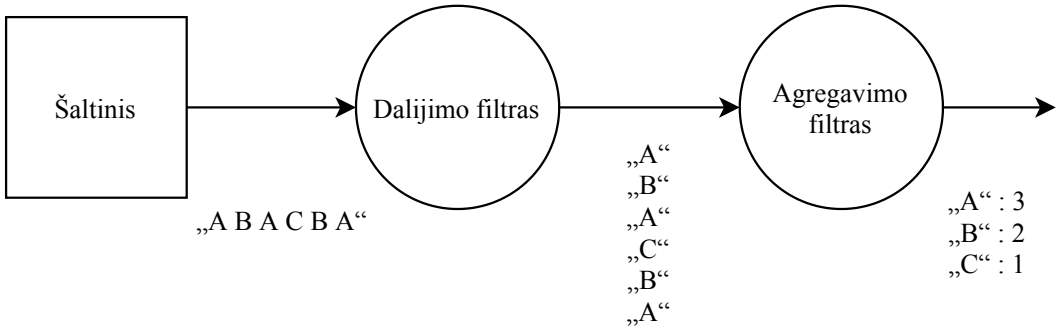
Šaltinis	Vėlinimas	Pralaidumas
[SÇZ05]	Taip	Taip
[KRK+18]	Taip	Taip
[HSS+14]	Ne	Taip
[QWH+16]	Taip	Taip
[ZYL+20]	Taip	Ne
[FAG+17]	Ne	Taip
[VC18]	Taip	Ne
[CDE+16]	Taip	Ne

Pagal literatūros analizę (1 len.) matoma, kad dauguma autorių renkasi vertinti tik pagal vieną metriką ir dažniau matavimui naudojamas vėlinimas. Taip pat [VC18], naudojantis skatinamąjį mokymą, matavimui naudoja vėlinimą ir [CDE+16] straipsnis, kuris siūlo srautinio apdorojimo sistemų vertinimo sprendimą, naudoja vėlinimą.

1.2.2. Srautinio apdorojimo sistemos pobūdis

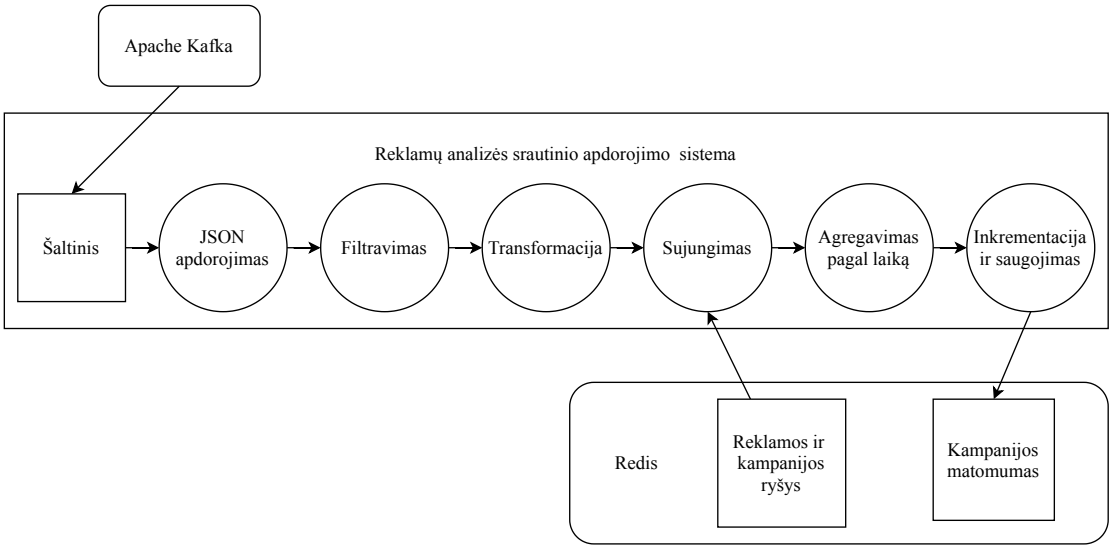
Srautinio apdorojimo sistemos gali turėti skirtingą elementų išsidėstymą ir nuo to priklausys jų greitaveiką. [KRK+18] matavimui naudoja du filtrus – agregavimo, kuris skaičiuoja visus pirkimus ir jungimo (angl. join), kuris skaičiuoja duomenis pagal tam tikrą bendrą rodiklį iš abiejų duomenų srautų. [QWH+16] srautinio apdorojimo variklių palyginimui naudoja septynis skirtingus uždavinius. Vienas iš jų yra WordCount uždavinys, kuris yra plačiai priimtas kaip didelių duome-

nų apdorojimo sistemos matavimo standartas [HHD+10]. Šis uždavinys susidaro iš dviejų filtrų: pirmas išskaido teksto eilutę į žodžius, o antras agreguoja kiekvieno žodžio bendrą skaitiklį ir atnaujina bendrą žodžių panaudojimo dažnio rezultatą, kuriame raktas – žodis, o reikšmė skaičius, kuris rodo kiek kartojosi šis žodis (2 pav.).



2 pav. WordCount sistemos pavyzdys

[ZYL+20] matavimui naudoja WordCount sistemą, kuri yra paprastesnė, nei pavaizduota 2 paveikslėlyje, nes šaltinis generuoja ir siunčia tik po vieną žodį ir todėl yra tik agregavimo filtras. Taip pat autoriai naudoja SentenceWordCount sistemą, kuri yra identiška 2 paveikslėlyje pavaizduotai sistemai. Bei autoriai sukūrė FileWordCount sistemą, kuri atlieka tą patį kaip ir SentenceWordCount, tačiau šaltinis negeneruoja žodžius, o skaito iš tekstinio dokumento ir taip pat naudoja egzistuojančią Yahoo srautinio apdorojimo vertinimą (angl. benchmarking) [CDE+16]. [FAG+17] autoriai naudoja WordCount eksperimentui. [VC18] eksperimentams naudoja Yahoo srautinio apdorojimo variklių vertinimą [CDE+16] ir taip pat atlieka bandymus su realiais daiktų interneto (angl. internet of things) įmonės duomenimis. [CDE+16] apibrėžia srautinio apdorojimo sistemą skirtingu srautinio apdorojimo variklių vertinimui. Pateikiama sistema analizuoja reklamas pagal kampaniją ir matomumą ir rezultatus deda į Redis duomenų bazę. Ši sistema sukurta taip, kad aprėptų visas srautinio apdorojimo sistemos savybes (3 pav.).



3 pav. Reklamų analizės sistema [CDE+16]

Straipsniai ([FAG+17; HHD+10; QWH+16]) naudoja WordCount (2 pav.), o [CDE+16; VC18] naudoja Reklamų analizės srautinę apdorojimo sistemą (3 pav.). Šiame darbe tyrimai atliekami su Reklamų analizės sistema, kadangi ši sistema sukurta srautinių apdorojimo variklių vertinimui ir turi mechanizmą valdyti srautą ir gauti tikslius vėlinimo duomenis,

1.2.3. Srautinio apdorojimo sistemų matavimo duomenys

Vertinant sistemų greitaveiką reikia atsižvelgti ir į testavimui naudojamus duomenis. [KRK+18] naudoja žaidimų kūrimo įmonės Rovio duomenis ir naudoja du duomenų srautus – pirkimo srautas, kuriame siunčiami kortežai (angl. tuples) sudaryti iš nupirktos valiutos kiekio, laiko ir naudotojo, kuris ją nupirko ir reklamų srautas, kuris siunčia valiutos reklamas tam tikru laiku. Šiame sprendime duomenis generuojami naudojant normalizuotą paskirstymą ant raktinio lauko. [QWH+16] naudoja tekstinius duomenis iš AOL paieškos variklio ir apdoroja juos pagal pasirinktus uždavinius. [ZYL+20] matavimui naudoja šaltinių generuojamą tekstą, kadangi lyginamas tas pats srautinio apdorojimo variklis tik su patobulinta planavimo posisteme ir naudoja iš anksto sugeneruotą tekstą patalpintą į tekstinį dokumentą. [CDE+16] aprašo sistemą, kuri daro skirtingų srautinio apdorojimo variklių vertinimą. Šiam vertinimui naudojami duomenys simuliuojantys reklamas ir reklamų kampanijas. Autoriai naudoja savo duomenų generatorių. Kadangi darbe naudojamas [CDE+16] pateikiamą Reklamos analizės sistemą, todėl šios sistemos duomenų kūrimui bus naudojamas pateiktas duomenų generatorius.

1.3. Srautinio apdorojimo sistemų derinimas

Sistemų greitaveika yra tiesiogiai susijusi su konfigūravimo parametrais, kurie valdo tokius aspektus kaip: atminties valdymas, gijų skaičius, planavimas, resursų valdymas [LCH+19]. Taip pat, neteisingi nustatymai turi nuostolingus efektus sistemos greitaveikai ir stabilumui [HLL+11].

[HCL20] išskiria 3 pagrindinius automatinio derinimo iššūkius:

1. Didelė ir sudėtinga parametrų erdvė – „Apache Spark“ ir „Apache Storm“ turi virš 150 konfigūruojamų parametrų [BC17; PGT16]. Taip pat, nustatymų reikšmės, kurios tinka vienam uždaviniui, gali turėti neigiamos įtakos kitam [HLL+11; JC16].
2. Sistemų mastas ir sudėtingumas – Sistemų administratoriai turi gebėti konfigūruoti didelius kiekius skaičiavimo mazgų, kurie gali turėti skirtingus CPU, atminties, tinklo tipus [HCL20].
3. Pradinių duomenų statistikos trūkumas – įvedimo duomenys srautinėse apdorojimo sistemose yra realus srautai, kurie stipriai varijuoja savo apimtimi [DP18].

[TLW17] nagrinėjantis tinkamos konfigūracijos radimą naudojant genetinius algoritmus „Apache Storm“ srautinio apdorojimo sistemoms nustatė, jog lygiagretumo laipsnis labiausiai daro įtaką srautinio apdorojimo sistemų greitaveikai. „Apache Heron“ srautinio apdorojimo variklis, kuris yra „Apache Storm“ su patobulinimais [KBF+15], pateikia naują būdą kontroliuoti srautą – priešslėgis (angl. backpressure), kuris leidžia filtrui sulėtinti prieš jį einantį elementą, kas leidžia sumažinti vėlinimą ir taip pat gali būti naudojamas kaip greitaveikos praradimo indikatorius [BCB+18]. Taip pat [BCB+18] nagrinėja „Apache Heron“ automatinį konfigūravimą naudojant iš anksto aprašytas taisykles.

2. Mašininis mokymasis

2.1. Mašininis mokymasis srautinio apdorojimo sistemų derinimui

[HCL20] aprašo skirtingus sprendimus automatiniam konfigūravimui ir išskiria šiuos mašininio mokymosi privalumus:

- Nebūtina suprasti sistemos, užduočių ir duomenų, kadangi naudojamas juodos dėžės (angl. black–box) principas.
- Mašininio mokymosi modelis pats save tobulina, ir yra vis tikslesnis kuo daugiau gauna duomenų.

Šio straipsnio autoriai išskiria mašininio mokymosi iššūkius:

- Parametrų parinkimas – kadangi konfigūruojamų parametrų kiekis yra didelis [BC17; PGT16] ne visi iš jų vienodai daro įtaką greitaveikai, todėl pirma verta išsirinkti aktualiausius parametrus resursų valdymo, užduočių planavimo ir duomenų valdymo užduotims. Tam dažnai naudojama eksperto pagalba [WXH16], gidai arba eksperimentavimas. Tačiau galima naudoti mašininio mokymosi algoritmą koreliacijos nustatymui tarp parametrų ir greitaveikos [YLL+12; VC18]
- Mašininio mokymosi modelio pasirinkimas – kadangi yra nemažai skirtingų mašininio mokymosi metodų kurie tinka derinimo uždaviniui.

Taip pat autoriai pateikia paketinio ir srautinio apdorojimo derinimą naudojant mašininį mokymąsi straipsnius (2 lentelėje pateikiami tik išrinkti srautinio apdorojimo pavyzdžiai).

2 lentelė. Srautinių sistemų derinimo naudojant mašininį mokymąsi pavyzdžiai [HCL20]

Šaltinis	Įvesties savybės	Mašininio mokymosi metodai
Zacheilas et al. [ZKZ+15]	Rinkinys kelių konfigūracijos parametrų	Gaussian Processes
Li et al. [LTX16]	Atminties dydžiai ir branduolių ir gijų kiekis skirtingose stadijose	Support Vector Regression
Trotter et al. [TLW17]	Darbinių procesų kiekis, vykdytojų kiekis	Genetic Algorithm, Bayesian Optimization
Trotter et al. [TWH19]	Vykdytojai, šaltinių ir filtrų lygiagretumas, acker lygiagretumas	Genetic Algorithm, Support Vector Machines
OrientStream [WMG+17]	Įvairios duomenų, plano, filtrų ir klasterio lygio savybės	Ensemble/ Incremental ML
Vaquero et al. [VC18]	Parametrai ir metrikos parinkti faktorinės analizės (angl. factor analysis) pagalba	Reinforcement Learning

2.2. Skatinamasis mašininis mokymasis

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 atliekant bandymus ir klystant, su papildoma algoritmu pagalba siekiant padidinti efektyvumą ir sprendimų stabilumą [KLM96].

2.2.1. Skatinamasis mokymasis srautinio apdorojimo veikimo gerinimui

[VC18] nagrinėja srautinių sistemų derinimą naudojant skatinamąjį mokymąsi. Autoriai pradžioje pasirenka Lasso path analysis algoritmą, kurio pagalba atlieka parametrų atranka, kad išrinktų svarbiausius parametrus pasirinktos metrikos valdymui. Konfigūracijos valdymui pasirink-

tas modifikuotas REINFORCE algoritmas. Atliktas eksperimentas su „Apache Spark“, kuris rodo 60–70% sumažintą vėlinimą.

[NLY+19] nagrinėja resursų valdymo problemą srautinio apdorojimo sistemose ir siūlo sprendimą naudojanti skatinamąjį mokymą, kuris daro optimizacijas pagal srautinės apdorojimo sistemos grafus. Eksperimentas atliekamas naudojant REINFORCE [Wil92] algoritmą su Adam optimizacijos funkcija [KB14] ir atliekamas eksperimentas naudojant tyrimui parašytą srautinio apdorojimo variklį ir sistemas.

[LXT+18] nagrinėja planavimo problemos (apkrovos paskirstymas darbiniams elementams) sprendimą naudojant skatinamąjį mokymąsi. Autoriai siūlo naudoti Actor–Critic lillicrap2015continuous metodą naudojant Deep Q Learning [MKS+15] tinklą kaip Actor ir bet kokį gilųjį neuroninį tinklą kaip Critic. Autorių rezultatai rodo 45% greitaveikos padidėjimą lyginant su integruota „Apache Storm“ planavimo posisteme.

[RCP19] nagrinėja srautinių sistemų dislokavimą naudojant skatinamąjį mokymą. Sprendžiama problema dislokavimo valdymo su skirtingais skaičiavimo mazgų tipais. Naudojamas Q Learning algoritmas su įvairioms modifikacijom (kombinuojama su tiksliais modeliais). Autoriai matuoja dislokavimo tikslumą ir konvergavimo greitį.

3 lentelė. Skatinamojo mokymosi naudojimas

Šaltinis	Skatinamojo mokymosi algoritmas
[VC18]	Adaptuotas REINFORCE [Wil92]
[NLY+19]	REINFORCE [Wil92] su Adam optimizacijos funkcija [KB14]
[LXT+18]	Deep Q Learning [MKS+15] ir Actor–Critic [LHP+15]
[RCP19]	Q–learning [MKS+15] su papildomomis funkcijomis

2.2.2. Balansavimas naudojant REINFORCE algoritmą

Straipsnis [VC18] nagrinėja automatinį balansavimą srautinio apdorojimo sistemų Apache Spark platformoje. Straipsnyje nagrinėjamas sprendimas susidaro iš trijų sistemų:

1. Sistema, kurioje iš 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 greitaveikai. 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 surūšiuojamas pagal įtaką greitaveikai. 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 naudodamas surikiuotų konfigūracijos elementų sąrašą ir pagrindinių metrikų sąrašą periodiškai atnaujina 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.

2.2.3. Deep Q Network algoritmas

Deep Q Network yra Q Learning įgyvendinimas naudojant giliuosius neuroninius tinklus. Q Learning - skatinamojo mokymosi algoritmas, kuris bet kokiam baigtiniam Markovo pasirinkimo procesui randa optimalius sprendimus maksimizuojančius 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 aproksimacija. Pradžioje visos Q reikšmės yra 0 ir algoritmas atlikdamas veiksmus pildo lentelę atnaujintomis reikšmėmis. Tačiau, kai veiksmų ir būsenų pasidaro per daug Q Learning algoritmo nebeužtenka ir tenka naudoti giliuosius neuroninius tinklus.

Deep Q Network skiriasi 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. experience 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 rinkinys perėjimų iš patirties pakartojamo kortežų rinkinio ir ap-

skaičiuojamas 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.

2.2.4. Actor–Critic with Experience Replay

2.2.5. Apibendrinimas

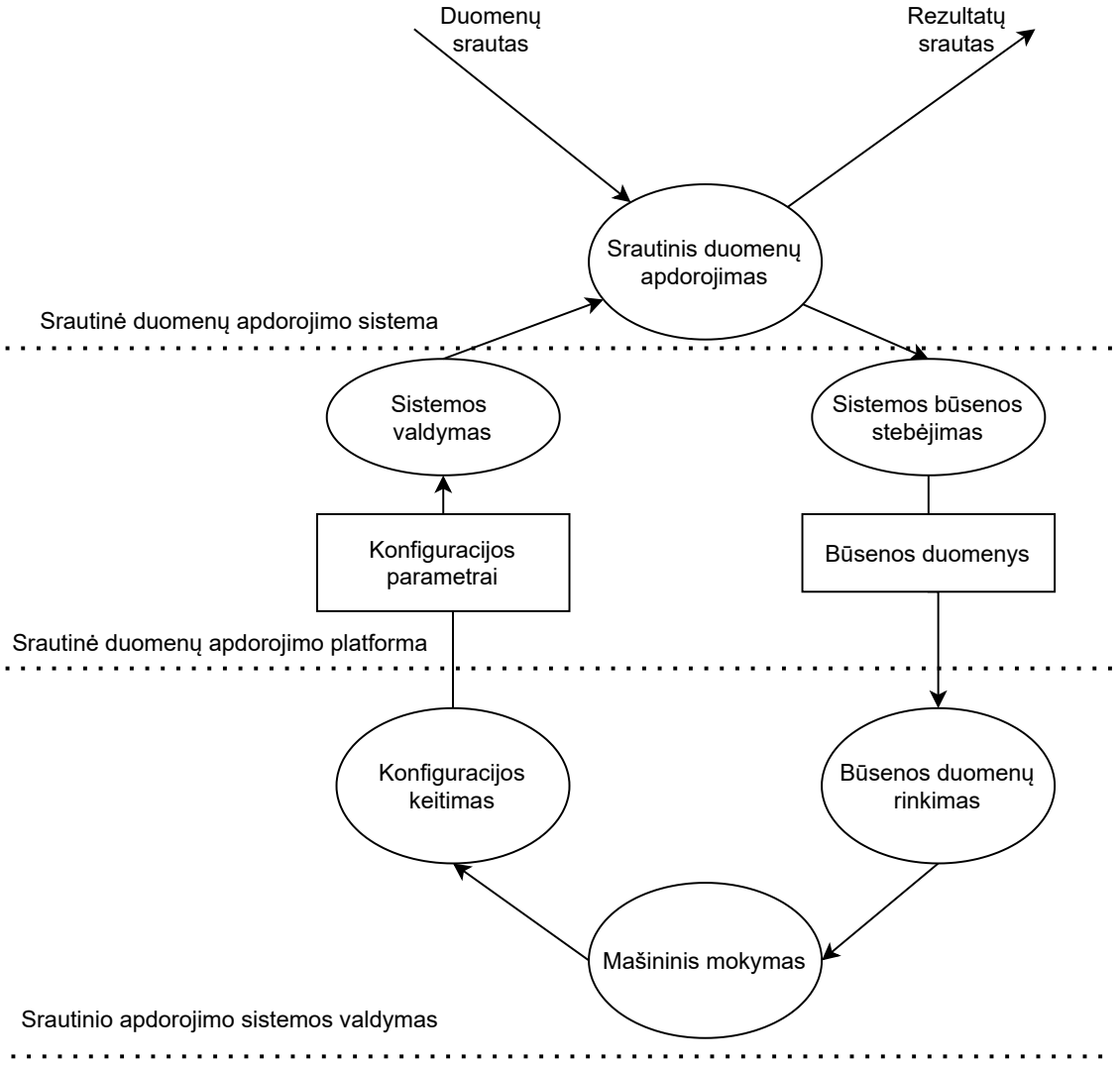
Šiame darbe norint patikrinti ar skatinamasis mokymasis tinka srautinio apdorojimo sistemų balansavimui naudojami keli algoritmai siekiant užtikrinti, jog rezultatai atitiktų skatinamojo mokymosi gebėjimą balansuoti, o ne pavienio algoritmo. Pasirinkti algoritmai:

- REINFORCE sukonfigūruotas pagal [Wil92] pateiktą konfigūracija.
- Deep Q Network remiantis [MKS+15].
- Actor–Critic with Experience Replay remiantis [WBH+16], kadangi jam reikia mažiau duomenų nei Deep Q Network.

3. Srautinės duomenų apdorojimo sistemos, valdomos mašininiu mokymusi, modelis

Tyrime nagrinėjamas srautinės sistemos, balansuojamos mašininių mokymų, modelis (4 pav.) susidaro iš trijų pagrindinių elementų: srautinio duomenų apdorojimo sistemos, srautinio duomenų apdorojimo platformos ir valdymo sistemos.

3.1. Modelis



4 pav. Srautinės apdorojimo sistemos modelis

Srautinės duomenų apdorojimo sistemą sudaro šie elementai:

- Duomenų srautas – duomenys patenkantys į srautinio apdorojimo sistemą nepertraukiamu srautu, iš anksto nežinomu greičiu ir nekontroliuojamu kiekiu.

- Srautinis duomenų apdorojimas – srautinio duomenų apdorojimo sistema, atliekanti skaičiavimus su duomenimis ateinančiais iš duomenų srauto. Tyrime naudojamos Heron srautinio apdorojimo sistemos pasižymi individualiais skaičiavimo komponentais, kurie skaičiuoja kiekvieną patenkantį duomenį ir yra parašyti užtikrinant lygiagretumą komponento lygyje.
- Rezultatų srautas – duomenys apdoroti srautinio duomenų apdorojimo sistemos ir perduoti iš paskutinio skaičiavimo komponento į kitas sistemas.

Srautinio apdorojimo sistemų platforma turi šiuos elementus:

- Srautinį duomenų apdorojimą.
- Sistemos būsenos stebėjimą – srautinio apdorojimo platformos sistema renkanti srautinio apdorojimo sistemų veikimo metrikas ir šias metrikas atskleidžianti į išorę. Tyrime naudojama Heron platforma metrikas pateikia kiekvienam skaičiavimo komponentui individualiai per HTTP protokolą arba į naudotojo pateiktą metrikų surinkimo sistemą.
- Būsenos duomenys – tai metrikos vaizduojančios kiekvienos srautinio apdorojimo sistemos skaičiavimo komponentų veikimo rodiklius, tokius kaip vėlinimas, pralaidumas, apkrovos ir t.t.
- Konfigūracijos parametrai – tai konfigūracijos rinkinys kurį apibrėžia srautinio apdorojimo platforma. Šie konfigūracijos parametrai nurodo srautinės apdorojimo sistemos veikimą ir taip pat gali apibrėžti parametrus skirtus individualiems skaičiavimo komponentams. Tyrime naudojama Heron platforma apibrėžia ir valdo visus srautinės apdorojimo sistemos konfigūracijos parametrus.
- Sistemos valdymas – konfigūracijos parametrų pateikimas į srautinio apdorojimo platformą. Šie konfigūracijos parametrai nurodo srautinės apdorojimo sistemos ir jos skaičiavimo komponentų veikimą. Tyrime naudojama Heron platformą leidžia pateikti konfigūracijos parametrus per komandinę eilutę. Pateikus konfigūraciją platforma sustabdo srautinę apdorojimo sistemą, atnaujina jos konfigūraciją ir paleidžia sistemą iš naujo. Kai sistema sustabdoma, taip pat nustojama ir skaityti duomenis iš duomenų srauto, o paleidus sistemą duomenys skaitomi toliau.

Srautinio apdorojimo sistemos valdymo elementai susidaro iš:

- Būsenos duomenų rinkimo – tai sistema renkanti duomenis apie srautinės apdorojimo sistemos būseną iš srautinės apdorojimo platformos. Ši sistema atsakinga už aktualių metrikų surinkimą, kurios naudojamos suformuoti vaizdą apie srautinio duomenų apdorojimo sistemos būseną.
- Mašininio mokymosi – sistema, kurį gauna srautinės apdorojimo sistemos būsenos duomenis

ir pagal tai apskaičiuoja naujas konfigūracijos parametrų reikšmes. Tyrime naudojamas skatinamasis mašininis mokymasis, kuris nuolatos bando gerina sistemos būseną apskaičiuojant konfigūracijos parametrų pokyčius ir mokosi iš anksčiau padarytų sprendimų.

- Konfigūracijos keitimo – sistema priimanti atnaujintus konfigūracijos parametrus ir pateikianti juos į srautinio apdorojimo platformą.

Visumoje 4 paveikslėlis apibrėžia duomenų judėjimą srautinio duomenų apdorojimo sistemos valdymo modelyje. Srautinio apdorojimo sistema yra pagrindinis elementas atsakingas už patį duomenų apdorojimą ir veikia nepriklausomai nuo kitų elementų modelyje. Srautinio apdorojimo platforma talpina ir palaiko srautinio apdorojimo sistemas, taip pat suteikia prieiga gauti informaciją apie srautinio apdorojimo sistemų būseną bei suteikia galimybę valdyti srautinio apdorojimo sistemas. Srautinio apdorojimo sistemų valdymo elementai atsakingi už srautinės apdorojimo sistemos konfigūracijos keitimą pagal surinktus būsenos duomenis.

3.2. Keičiami konfigūracijos parametrai

Norint koreguoti konfigūracijos parametrus reikia siųsti atnaujinimo komandą į Heron komandinės eilutės įrankį (toliau Heron CLI). Pateikus konfigūracijos parametrus Heron platformą perkrauna srautinio apdorojimo sistemą su naujais parametrais. Vienas iš pagrindinių srautinės apdorojimo sistemos konfigūracijos parametrų – skaičiavimo komponentų lygiagretumas, kuris nurodo kiek paleidžiama tam tikro komponento instancijų. Taip pat tai yra vienintelis keičiamas konfigūracijos elementas, kuris priklauso nuo valdomos srautinio apdorojimo sistemos sudėties.

Visi kiti konfigūravimo parametrai[Her19], kurie taip pat yra keičiami balansavimo metu pateikti 4 lentelėje:

4 lentelė. Keičiami konfigūracijos parametrai

Parametras	Paaiškinimas
component–parallelism=[skaičiavimo komponento pavadinimas]	Tam tikro skaičiavimo komponento lygiagretumas
heron.instance.tuning.expected.bolt.read.queue.size	Numatomas skaitomos eilės dydis Bolt tipo komponentuose
heron.instance.tuning.expected.bolt.write.queue.size	Numatomas rašomos eilės dydis Bolt tipo komponentuose

heron.instance.tuning.expected.spout.read.queue.size	Numatomas skaitomos eilės dydis Spout tipo komponentuose
heron.instance.tuning.expected.spout.write.queue.size	Numatomas rašomos eilės dydis Spout tipo komponentuose
heron.instance.set.data.tuple.capacity	Didžiausias kiekis kortežų sugrupuotu vienoje žinutėje
heron.instance.emit.batch.time.ms	Didžiausias laikas Spout tipo komponentui išsiųsti gautą kortežą
heron.instance.emit.batch.size.bytes	Didžiausias partijos dydis Spout tipo komponentui išsiųsti gautą kortežą
heron.instance.execute.batch.time.ms	Didžiausias laikas Bolt tipo komponentui apdoroti gautą kortežą
heron.instance.execute.batch.size.bytes	Didžiausias partijos dydis Bolt tipo komponentui apdoroti gautą kortežą
heron.instance.internal.bolt.read.queue.capacity	Skaitomos eilės dydis Bolt komponentams
heron.instance.internal.bolt.write.queue.capacity	Rašomos eilės dydis Bolt komponentams
heron.instance.internal.spout.read.queue.capacity	Skaitomos eilės dydis Spout komponentams
heron.instance.internal.spout.write.queue.capacity	Rašomos eilės dydis Spout komponentams
heron.api.config.topology_container_max_ram_hint	Daugiausiai operatyvios atminties kiekio konteineriui išskyrimo užuomina
heron.api.config.topology_container_max_cpu_hint	Daugiausiai procesoriaus pajėgumo konteineriui išskyrimo užuomina
heron.api.config.topology_container_max_disk_hint	Daugiausiai kietojo disko atminties kiekio konteineriui išskyrimo užuomina
heron.api.config.topology_container_padding_percentage	Užuominų galimą paklaidą

3.3. Naudojamos metrikos

Metrikos iš Heron srautinio apdorojimo sistemų gali būti pasiektos keliais skirtingais būdais: darant užklausą į Heron API, skaitant iš tekstinio failo, kurį pildo Heron platformą arba naudojant savo sukurtą metrikų skaitymo priedą, kuris pateikiamas į Heron platformą prieš ją paleidžiant. Visos metrikos yra saugomos srautinio apdorojimo sistemos kiekvienam skaičiavimo komponentui.

5 lentelėje aprašytos metrikos yra standartinės visiems Heron srautinio apdorojimo sistemos komponentams ir grąžinamos iš Heron per Heron API [Her20]. Šios metrikos perduodamos į mašininio mokymosi algoritmą ir naudojamos atlygio apskaičiavimui.

5 lentelė. Naudojamos metrikos

Metriką	Paaiškinimas
__emit–count (toliau išsiųstas kiekis)	Išsiųstų kortežų kiekis
__execute–count (toliau apdorotas kiekis)	Apdorotų kortežų kiekis Bolt komponentuose
__execute–latency (toliau vidutinis apdorojimo vėlinimas)	Vidutinė trukmė Bolt komponentui apdoroti kortežą

3.4. Tikslo funkcija

Srautinio apdorojimo sistemos valdymo tikslas – keičiant konfigūraciją, pasiekti didžiausią greitaveiką. Šiame tyrime greitaveika matuojama vėlinimu, todėl pasirinkta tikslo funkcija – tiesiogiai imama pagal vėlinimą (__execute–latency) ir duodamas teigiamas atlygis, jeigu pasiektas mažiausias vėlinimas, neigiamas – jeigu vėlinimas nėra, mažiausias, tokiu atveju kaip atlygis gražinamas skirtumas mažiausio gauto vėlinimo ir esamo vėlinimo.

4. Balansavimo algoritmas

4.1. Srautinės architektūros sistemos konfigūracijos valdymo algoritmas

Algoritmo tikslas – pagal esamą būseną ir esamą konfigūraciją apskaičiuoti naują konfigūraciją, kuri pagerintų srautinės apdorojimo sistemos greitaveiką. Algoritmo pagrindas yra pasirinkti skatinamojo mašininio mokymosi algoritmai Deep Q Network ir Actor–Critic with Experience Replay.

Kad algoritmas galėtų apskaičiuoti naują konfigūraciją, jam yra paduodami šie duomenys:

- Srautinio apdorojimo sistemos metrikos (5 lentelė)
- Srautinio apdorojimo sistemos pradinė konfigūracija (6 lentelė).
- Konfigūruojamų parametrų sąrašas ir jų galimos reikšmės, aprašytos intervalu (pavyzdžiui $[512,4096]$) arba tiksliais variantais (pavyzdžiui $[512, 1024, 2048, 4096]$).

Duomenys, kurie paduodami balansavimo algoritmui, yra renkami pastoviai, neatsižvelgiant į pačio algoritmo veikimą.

Balansavimo algoritmas veikia ciklais visą srautinės apdorojimo sistemos veikimo laiką ir tarpas tarp ciklų turi būti pakankamai ilgas srautinio apdorojimo sistemai atsinaujinti su naujais konfigūracijos parametrais bei, kad surinktų duomenų kiekis būtų pakankamai svarūs pateikti mašininio mokymosi algoritmui. Balansavimo algoritmas, atlikęs skaičiavimus, grąžina naują konfigūracijos parametrų rinkinį ir laukia naujo ciklo pradžios.

Balansavimui naudojami skatinamojo mašininio mokymosi algoritmai Deep Q network [FWX+20] ir Actor–Critic with Experience Replay [WBH+16] turi bendrus veikimo bruožus:

- Jie yra laisvo nuo modelio (angl. model–free) tipo skatinamojo mokymosi algoritmai. Tai aktualu mūsų atveju kadangi balansavimo algoritmas turi galėti prisitaikyti prie kintančių duomenų kiekio ir greičio bei kintančios aplinkos.
- Jie yra be strategijos (angl. off–policy) tipo, tai reiškia, kad jie mokosi atliekant skirtingus veiksmus, nebūtinai tuos, kurie buvo pasirinkti pagal dabartinę strategiją.

4.2. Srautinio duomenų apdorojimo aplinkos apibrėžimas balansavimui

Balansavimui atlikti reikalingas tikslus apibrėžimas srautinio apdorojimo aplinkos, šis apibrėžimas susidaro iš: dabartinės srautinio apdorojimo sistemos būsenos, galimų veiksmų aibės ir žingsnio funkcijos, kuri atlieka veiksmą ir apskaičiuoja atlygį.

Srautinio apdorojimo sistemos balansavimo galimų veiksmų aibė – konfigūracijos paramet-

rai, kuriuos algoritmas gali koreguoti ir jų reikšmių aibė ir komponentų lygiagretumas. Pasirinkti algoritmai naudoja diskrečią veiksmų aibę. Todėl buvo sugeneruoti konfigūracijos parametrų rinkiniai pagal 6 pateiktas aibes.

6 lentelė. Konfigūracijos elementų aibė

Parametras	Natūraliųjų reikšmių aibė
component–parallelism=[skaičiavimo komponento pavadinimas]	$[1; 10] * \text{Komponentų kiekis}$
heron.instance.tuning.expected.bolt.read.queue.size	$[2; 20]$
heron.instance.tuning.expected.bolt.write.queue.size	$[2; 20]$
heron.instance.tuning.expected.spout.read.queue.size	$[128; 512]$
heron.instance.tuning.expected.spout.write.queue.size	$[2; 20]$
heron.instance.set.data.tuple.capacity	$[64; 512]$
heron.instance.emit.batch.time.ms	$[8; 128]$
heron.instance.emit.batch.size.bytes	$[8192; 65536]$
heron.instance.execute.batch.time.ms	$[8; 128]$
heron.instance.execute.batch.size.bytes	$[8192; 65536]$
heron.instance.internal.bolt.read.queue.capacity	$[64; 512]$
heron.instance.internal.bolt.write.queue.capacity	$[64; 512]$
heron.instance.internal.spout.read.queue.capacity	$[512; 4096]$
heron.instance.internal.spout.write.queue.capacity	$[64; 512]$
heron.api.config.topology_container_max_ram_hint	$[256; 2048]$
heron.api.config.topology_container_max_cpu_hint	$[20; 80]$
heron.api.config.topology_container_max_disk_hint	$[8192; 65536]$
heron.api.config.topology_container_padding_percentage	$[0; 20]$

Sujungus konfigūracijos atnaujinimą ir komponentų lygiagretumo valdymo sudarytas galimų veiksmų rinkinys (7 pav.). Pasirinkta išdalinti konfigūracijos rinkinius į 10 dalių siekiant gauti didesnius pokyčius atnaujinant konfigūracija ir taip pagreitinti modelio apmokymą. Taip pat modeliui pasirinkus veiksmą, kuris išeitų iš ribų, atliekamas 0 veiksmas – nedaryti nieko.

7 lentelė. Galimų aplinkos veiksmų aibė

Veiksmo numeris	Veiksmas	Riba
0	Nedaryti nieko	–
1	Atnaujinti konfigūraciją, kurios numeris vienetų žemesnis	0
2	Atnaujinti konfigūraciją, kurios numeris vienetų aukštesnis	10
1 + 2n	Atnaujinti n–tojo komponento konfigūracija, sumažinant jo lygiagretumą	1
2 + 2n	Atnaujinti n–tojo komponento konfigūracija, padidinant jo lygiagretumą	9

Srautinio apdorojimo sistemos dabartinė būsena, kuri yra gražinama – apibrėžiama kaip masyvas kurio ilgis yra komponentų kiekis + 1, o masyvo elementai rodo komponento lygiagretumo skaičių ir vienas elementas rodo dabartinės konfigūracijos numerį.

Skatinamojo mokymosi algoritmo veikimui reikalingas atlygis, kuris darbe naudojamos aplinkos skaičiuojamas pagal for formulę:

$$Atlygis(n) = \begin{cases} \text{min(Vėlinimas)}– \text{ Vėlinimas}_n & \text{Vėlinimas}_n \geq \text{min(Vėlinimas)} \\ C– \text{ Vėlinimas}_n & \text{Vėlinimas}_n < \text{min(Vėlinimas)} \end{cases}$$

C – konstanta apskaičiuojama per pirmą paleidimą, suapvalinus gautą vėlinimą su pradinę konfigūracija.

Srautinio apdorojimo sistemos balansavimo žingsnio funkcija susidaro iš:

- Įeities – atliekamo veiksmo skaičiaus, kuris nurodo aplinkai ką ji turi atlikti.
- Išeities – kuri susidaro iš reikšmių kortežo:
  - Stebėjimo būsena – srautinio apdorojimo sistemos dabartinė būsena.
  - Atlygio – apskaičiuojamas naudojant metrikas.
  - Epizodo baigtumo rodiklio – nurodo ar aplinka baigė epizodą ir turi būti paleista iš naujo.
  - Papildoma informacija – naudojama derinimo pagalbai, neprivaloma.

4.3. Balansavimo algoritmo apmokymas

Balansavimo algoritmo apmokymas vyks srautinio apdorojimo sistemos veikimo metu. Paleidus srautinę apdorojimo sistemą bus paleidžiamas ir balansavimo algoritmas. Tam, kad algoritmas turėtų pagrindą veiksmų pasirinkimui, algoritmo veikimo pradžioje konfigūracija bus pasirenkama atsitiktinai apibrėžtą ciklų kiekį ir renkami rezultatai mašininio tinklo apmokymui. Po šio ciklo algoritmas atliks sprendimus pagal patirtį. Deep Q Network ir Soft Actor Critic algoritmai naudoja pakartojimo buferio mokymosi optimizaciją, kai algoritmas mokosi ne iš paskutinio atlikto veiksmo, o iš atsitiktinai pasirinktos aibės, kurios elementai susidaro iš veiksmo, būsenos, atlygio ir naujos būsenos rinkinių. Pakartojimo buferio aibė saugoma viso mokymosi metu ir yra konfigūruojamas mokymosi metu pasirenkamos aibės dydis.

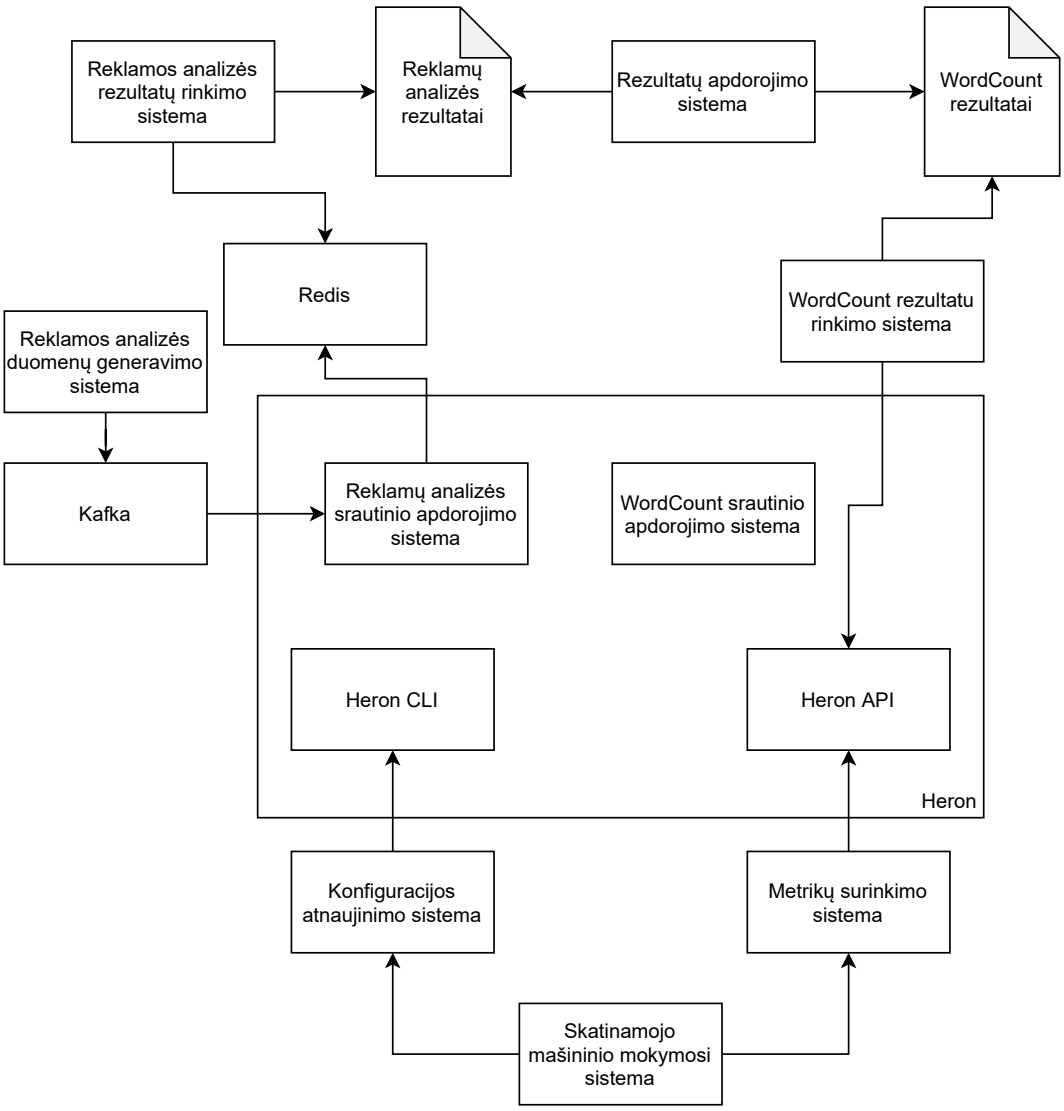
5. Eksperimento tyrimo planas

5.1. Tyrimo tikslas

Šio tyrimo tikslas – įvertinti siūlomo balansavimo modelio ir pasirinkto optimizavimo algoritmo validumą. Tam reikia atlikti bandymus su eksperimentine sistema naudojančia aprašyta optimizavimo algoritmą, su eksperimentine sistema naudojančia REINFORCE algoritmą ir su aplinka naudojančią standartinę konfigūracija be jokių pakeitimų. Gautus bandymo duomenis palyginti ir nustatyti, ar pasiūlytas sprendimas tinka srautinio apdorojimo sistemų balansavimui.

5.2. Eksperimentinė sistema

Eksperimentui atlikti naudojama lokali Heron aplinka ir papildomos posistemės skirtos duomenų generavimui, rezultatų rinkimui ir t.t. Pilna eksperimento sistema pavaizduota 5 paveikslėlyje.



5 pav. Eksperimento sistemos architektūra

Sistemos, kurios naudojamos eksperimente:

1. Skatinamojo mašininio mokymosi sistema. Kadangi atliekami bandymai su skirtingais skatinamojo mokymosi algoritmais, kiekvienam jų sukurta atskira sistema:
    - Skatinamojo mokymosi sistema naudojanti REINFORCE algoritmą, įgyvendinta naudojant tensorflow biblioteką ir oficialiai pateiktą dokumentaciją šio algoritmo įgyvendinimui. Taip pat šio algoritmo hiperparametrai sudėti pagal pateiktus [VC18] straipsnyje.
    - Giliojo skatinamojo mokymosi sistema naudojanti Deep Q Network algoritmą, įgyvendinta naudojant tensorflow ir stable–baselines bibliotekas ir naudojant rekomenduojamus hiperparametrus.
    - Giliojo skatinamojo mokymosi sistema naudojanti Actor–Critic with Experience Replay algoritmą, įgyvendinta naudojant tensorflow ir stable–baselines bibliotekas ir nau-

dojant rekomenduojamus hiperparametrus.

2. Skatinamojo mokymosi aplinka – aprašanti srautinio apdorojimo sistemos būseną, veiksmų aibę ir konfigūracijos žingsnį, kuris atnaujina srautinio apdorojimo sistemą, surenką būsenos duomenis ir apskaičiuoja atlygio reikšmę.
3. Aplinkai reikalingos papildomos posistemės atlikti atnaujinimus ir surinkti duomenis:
    - Metrikų surinkimo sistema parašyta Python kalba, naudojanti HTTP protokolą, būsenos duomenų surinkimui iš Heron API.
    - Konfigūracijos atnaujinimo sistema parašyta su Python, skirta konfigūracijos atnaujinimui ir pateikimui į Heron sistemą per Heron CLI.
4. Reklamų srautinio apdorojimo sistema parašyta su JAVA, gaunanti duomenis iš Kafka žinučių eilės ir sauganti rezultatus Redis duomenų bazėje.
5. Rezultatų apdorojimo sistema parašyta su Python, kuri surenka duomenis iš rezultatų failų, apdoroja juos ir grąžina skirtingų sprendimų diagramas.
6. Reklamos analizės rezultatų rinkimo sistema parašyta su Clojure ir pateikta kartu su Reklamos analizės greitaveikos testu.
7. Reklamos analizės duomenų generavimo sistema parašyta su Clojure ir pateikta kartu su Reklamos analizės greitaveikos testu.

Kompiuterinės įrangos su kuria atliekamas eksperimentas parametrai:

- Procesorius: Intel Core i7–5930k (6 branduoliai/12 gijų)
- Operatyvi atmintis: 64 GB (2666 MHz)
- Vaizdo plokštė: Nvidia GTX 1080Ti
- Operacinė sistema: Windows 10 Education

Kadangi Heron nepalaiko Windows sistemos visi tyrimai ir visos reikiamos posistemės leidžiamos naudojant Windows Subsystem Linux (toliau WSL) su Ubuntu 18.04. WSL yra suderinamumo sluoksnis naudojantis Linux branduolį per Hyper–V, kieno pagalba galima leisti programas skirtas Linux be emuliacijos neprarandant greitaveikos.

Pagrindinės programinės įrangos versijos:

- Apache Heron: 0.20.3
- Kafka: 2.13
- Redis: 4.0.9
- Python: 3.6.9
- Java: 8

5.3. Eksperimentų apimtis

Eksperimentai atlikti su keturiais skirtingais sprendimais:

- Srautinio apdorojimo sistemos veikimas su standartine konfigūracija.
- Srautinio apdorojimo sistemos veikimas balansuojant ją naudojant sukurtą eksperimentinį sprendimą pagal apibrėžtą balansavimo algoritmą naudojant:
  - REINFORCE algoritmą skatinamojo mokymosi posistemėje.
  - Deep Q Network algoritmą skatinamojo mokymosi posistemėje.
  - Actor–Critic with Experience Replay algoritmą skatinamojo mokymosi posistemėje.

Kiekvienas skatinamojo mokymosi sprendimas yra testuojamas su Reklamų analizės srautinio apdorojimo sistema.

Reklamų analizės sistema rezultatus talpina tekstini failą, kur kiekvienas įrašas yra vėlinimas milisekundėmis nuo paskutinio išsiųsto įrašo į žinučių eilę tam specifiniam kampanijos langui iki kol jis yra įrašomas į Redis duomenų bazę. Pasinaudojant šiais duomenimis, kiekvieną iteracija apskaičiuojamas vėlinimo vidurkis.

Eksperimentai buvo atliekami su visais algoritmais apmokius juos 50 iteracijų, 100 iteracijų ir 250 iteracijų, tuo siekiant palyginti rezultatus tarp algoritmų ir bendra veikimą. Algoritmo efektyvumui patikrinti apmokytas modelis atlieka konfigūracijos keitimą 25 iteracijas ir matuojamas vėlinamas, kiekvieno po kiekvieno pakeitimo praėjus 3 minutėm, laikas kurio reikia srautinio apdorojimo sistemai pasileisti ir pasiekti pilną apkrovos stadiją.

5.4. Eksperimento rezultatai

Rezultatai po 50 mokymosi iteracijų:

Rezultatai po 100 mokymosi iteracijų:

Rezultatai po 250 mokymosi iteracijų:

Rezultatai ir išvados

Literatūra

[BBD+02] Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani ir Jennifer Widom. Models and issues in data stream systems. *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, p. 1–16, 2002.

[BC17] Muhammad Bilal ir Marco Canini. Towards automatic parameter tuning of stream processing systems. P. 189–200, 2017-09. DOI: 10.1145/3127479.3127492.

[BCB+18] Manu Bansal, Eyal Cidon, Arjun Balasingam, Aditya Gudipati, Christos Kozyrakis ir Sachin Katti. Trevor: automatic configuration and scaling of stream processing pipelines, 2018. arXiv: 1812.09442 [cs.DC].

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

[Bon00] André B Bondi. Characteristics of scalability and their impact on performance. *Proceedings of the 2nd international workshop on Software and performance*, p. 195–203, 2000.

[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.

[DP18] Miyuru Dayarathna ir Srinath Perera. Recent advancements in event processing. *ACM Comput. Surv.*, 51(2), 2018-02. ISSN: 0360-0300. DOI: 10.1145/3170432.

[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.

[FY11] Zbynek Falt ir Jakub Yaghob. Task scheduling in data stream processing. *DATESO*, p. 85–96, 2011.

[FWX+20] Jianqing Fan, Zhaoran Wang, Yuchen Xie ir Zhuoran Yang. A theoretical analysis of deep q-learning. *Learning for Dynamics and Control*, p. 486–489. PMLR, 2020.

[HCL20] Herodotos Herodotou, Yuxing Chen ir Jiaheng Lu. A survey on automatic parameter tuning for big data processing systems. *ACM Computing Surveys (CSUR)*, 53(2):1–37, 2020.

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

[Her20] Apache Heron. Heron tracker rest api. `https://heron.incubator.apache.org/docs/user-manuals-tracker-rest`, 2020.

[HHD+10] Shengsheng Huang, Jie Huang, Jinquan Dai, Tao Xie ir Bo Huang. The hibench benchmark suite: characterization of the mapreduce-based data analysis. *2010 IEEE 26th International Conference on Data Engineering Workshops (ICDEW 2010)*, p. 41–51. IEEE, 2010.

[HLL+11] Herodotos Herodotou, Harold Lim, Gang Luo, Nedyalko Borisov, Liang Dong, Fatma Bilgen Cetin ir Shivnath Babu. Starfish: a self-tuning system for big data analytics. *Cidr*, tom. 11 numeris 2011, p. 261–272, 2011.

[HSS+14] Martin Hirzel, Robert Soulé, Scott Schneider, Buğra Gedik ir Robert Grimm. A catalog of stream processing optimizations. *ACM Computing Surveys (CSUR)*, 46(4):1–34, 2014.

[YLL+12] Hailong Yang, Zhongzhi Luan, Wenjun Li, Depei Qian ir Gang Guan. Statistics-based workload modeling for mapreduce. *2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum*, p. 2043–2051. IEEE, 2012.

[IS15] Muhammad Hussain Iqbal ir Tariq Rahim Soomro. Big data analysis: apache storm perspective. *International journal of computer trends and technology*, 19(1):9–14, 2015.

[JC16] Pooyan Jamshidi ir Giuliano Casale. An uncertainty-aware approach to optimal configuration of stream processing systems. *CoRR*, abs/1606.06543, 2016. eprint: `1606.06543`.

[KB14] Diederik P. Kingma ir Jimmy Ba. Adam: a method for stochastic optimization, 2014. arXiv: 1412.6980 [cs.LG].

[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.

[KF16] Supun Kamburugamuve ir Geoffrey Fox. Survey of distributed stream processing. *Bloomington: Indiana University*, 2016.

[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.

[KRK+18] Jeyhun Karimov, Tilmann Rabl, Asterios Katsifodimos, Roman Samarev, Henri Heiskanen ir Volker Markl. Benchmarking distributed stream processing engines. *ArXiv*, abs/1802.08496, 2018.

[LCH+19] Jiaheng Lu, Yuxing Chen, Herodotos Herodotou ir Shivnath Babu. Speedup your analytics: automatic parameter tuning for databases and big data systems. *Proceedings of the VLDB Endowment*, 12(12):1970–1973, 2019.

[LHP+15] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver ir Daan Wierstra. Continuous control with deep reinforcement learning, 2015. arXiv: 1509.02971 [cs.LG].

[LTX16] Teng Li, Jian Tang ir Jielong Xu. Performance modeling and predictive scheduling for distributed stream data processing. *IEEE Transactions on Big Data*, 2(4):353–364, 2016.

[LXT+18] Teng Li, Zhiyuan Xu, Jian Tang ir Yanzhi Wang. Model-free control for distributed stream data processing using deep reinforcement learning. *Proc. VLDB Endow.*, 11(6):705–718, 2018-02. ISSN: 2150-8097. DOI: 10.14778/3199517.3199521. URL: https://doi.org/10.14778/3199517.3199521.

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

[MKS+15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu ir k.t. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[NLY+19] Xiang Ni, Jing Li, Mo Yu, Wang Zhou ir Kun-Lung Wu. Generalizable resource allocation in stream processing via deep reinforcement learning, 2019. arXiv: 1911.08517 [cs.LG].

[NPP+17] Shadi A Noghabi, Kartik Paramasivam, Yi Pan, Navina Ramesh, Jon Bringhurst, Indranil Gupta ir Roy H Campbell. Samza: stateful scalable stream processing at linkedin. *Proceedings of the VLDB Endowment*, 10(12):1634–1645, 2017.

[PGT16] Panagiotis Petridis, Anastasios Gounaris ir Jordi Torres. Spark parameter tuning via trial-and-error, 2016. arXiv: 1607.07348 [cs.DC].

[QWH+16] S. Qian, G. Wu, J. Huang ir T. Das. Benchmarking modern distributed streaming platforms. *2016 IEEE International Conference on Industrial Technology (ICIT)*, p. 592–598, 2016.

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

[RCP19] Gabriele Russo Russo, Valeria Cardellini ir Francesco Lo Presti. Reinforcement learning based policies for elastic stream processing on heterogeneous resources. *Proceedings of the 13th ACM International Conference on Distributed and Event-Based Systems*, DEBS '19, p. 31–42, Darmstadt, Germany. Association for Computing Machinery, 2019. ISBN: 9781450367943. DOI: 10.1145/3328905.3329506. URL: https://doi.org/10.1145/3328905.3329506.

[RM19] Henriette Röger ir Ruben Mayer. A comprehensive survey on parallelization and elasticity in stream processing. *ACM Computing Surveys (CSUR)*, 52(2):1–37, 2019.

[SÇZ05] Michael Stonebraker, Uğur Çetintemel ir Stan Zdonik. The 8 requirements of real-time stream processing. *ACM Sigmod Record*, 34(4):42–47, 2005.

[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.

[Ste97] Robert Stephens. A survey of stream processing. *Acta Informatica*, 34(7):491–541, 1997.

[TLW17] M. Trotter, G. Liu ir T. Wood. Into the storm: descrying optimal configurations using genetic algorithms and bayesian optimization. *2017 IEEE 2nd International Workshops on Foundations and Applications of Self* Systems (FAS*W)*, p. 175–180, 2017.

[TV07] Andrew S Tanenbaum ir Maarten Van Steen. *Distributed systems: principles and paradigms*. Prentice-Hall, 2007.

[TWH19] Michael Trotter, Timothy Wood ir Jinho Hwang. Forecasting a storm: divining optimal configurations using genetic algorithms and supervised learning. *2019 IEEE International Conference on Autonomic Computing (ICAC)*, p. 136–146. IEEE, 2019.

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

[WBH+16] Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu ir Nando de Freitas. Sample efficient actor-critic with experience replay. *arXiv preprint arXiv:1611.01224*, 2016.

[Wil92] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[WMG+17] Chunkai Wang, Xiaofeng Meng, Qi Guo, Zujian Weng ir Chen Yang. Automating characterization deployment in distributed data stream management systems. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2669–2681, 2017.

[WXH16] Guolu Wang, Jungang Xu ir Ben He. A novel method for tuning configuration parameters of spark based on machine learning. *2016 IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, p. 586–593. IEEE, 2016.

[ZGQ+17] Xinwei Zhao, Saurabh Garg, Carlos Queiroz ir Rajkumar Buyya. A taxonomy and survey of stream processing systems. *Software Architecture for Big Data and the Cloud*, p. 183–206. Elsevier, 2017.

[ZYL+20] Yitian Zhang, Jiong Yu, Liang Lu, Ziyang Li ir Zhao Meng. L-heron: an open-source load-aware online scheduler for apache heron. *Journal of Systems Architecture*, 106:101727, 2020.

[ZKZ+15] Nikos Zacheilas, Vana Kalogeraki, Nikolas Zygouras, Nikolaos Panagiotou ir Dimitrios Gunopulos. Elastic complex event processing exploiting prediction. *2015 IEEE International Conference on Big Data (Big Data)*, p. 213–222. IEEE, 2015.