VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMŲ STUDIJŲ PROGRAMA

Srautinio apdorojimo sistemų balansavimas taikant mašininį mokymąsi

Balancing stream processing systems using machine learning

Mokslo tiriamasis darbas III

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

Vilnius – 2021

TURINYS

Įvadas

Realaus laiko duomenų apdorojimas (angl. real–time data processing) yra jau senai nagrinėjamas kaip vienas iš būdų apdoroti didelių kiekių duomenis (angl. Big data). Viena iš didelių duomenų apdororojimo tipinių architektūrų yra srautinis apdorojimas. Srautinis duomenų apdorojimas (angl. stream processing) – lygiagrečių programų kūrimo modelis, pasireiškiantis sintaksiškai sujungiant nuoseklius skaičiavimo komponentus srautais, kad kiekvienas komponentas galėtų skaičiuoti savarankiškai [Bea15].

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

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

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

ir tai daro tik reaktyviai.

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

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

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

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

Magistro darbo uždaviniai:

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

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

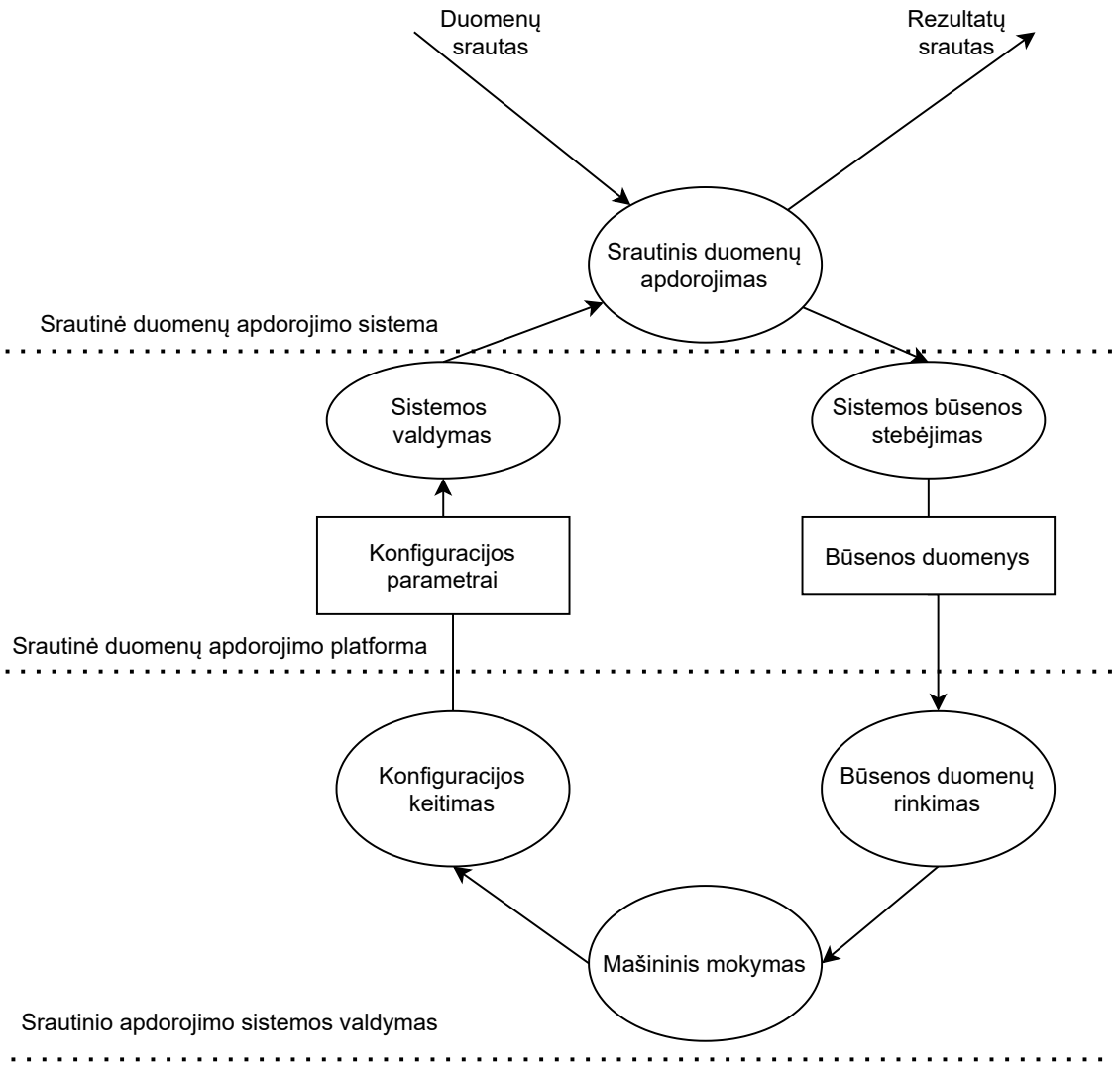
skatinamojo mokymosi algoritmai, o šiame mokslo tiriamajame darbe – trečioje magistro darbo dalyje – siekta tikslo: apibrėžti darbo metodą ir sukurti bei pagrįsti algoritmą, kuris bus naudojamas eksperimentui. Ir įgyvendinti šie uždaviniai:

1. Sukurti srautinio apdorojimo sistemos, balansuojamos skatinamojo mokymosi algortimais, modelį.
2. Apibrėžti eksperimento eigą ir reikalavimus eksperimentinei sistemai, kuri bus naudojama atlikti matavimus ir įvertinti sukurtą modelį.

1. Srautinės duomenų apdrojimo sistemos, valdomos mašininiu mokymusi, modelis

Tyrime nagrinėjamas srautinės sistemos, balansuojamos mašininių mokymų, modelis (1 pav.) susidaro iš trijų pagrindinių elementų: srautinio duomenų apdorojimo sistemos, srautinio duomenų apdorojimo platformos ir valdymo sistemos.

1.1. Modelis



1 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 konfiguracijos parametrų pokyčius ir mokosi iš anskčiau padarytų sprendimų.

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

Visumoje 1 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.

1.2. Keičiami konfigūracijos parametrai

Norint koreguoti konfigūracijos parametrus reikia siųsti atnaujinimo komandą į Heron komandinės eilutės įrankį (toliau Heron CLI). Peteikus 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 sudeties.

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

1 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

1.3. 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.

2 lentelėje aprašytos metrikos yra standartinės visiems Heron srautinio apdorojimo sistemos komponentams ir grąžinamos iš Heron per Heron API [Her20]. Šios metrikos ir bus perduodamos į mašininio mokymosi algoritmą kaip aplinkos būsenos apibūdinimas.

2 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
__jvm–process–cpu–load (toliau procesoriaus apkrova)	JVM apkrova procesoriui
__jvm–memory–used–mb (toliau naudojamas RAM)	JVM naudojama operatyvi atmintis
__jvm–memory–mb–total (toliau išskirtas RAM kiekis)	JVM turima operatyvi atmintis
__jvm–gc–collection–time–ms (toliau vidutinė GC trukme)	JWM šiušklių surinkimo trukmė
__time_spent_back_pressure_by_compid (toliau bendra priešslėgio trukmė)	Kiek laiko komponentui buvo įjungtas priešslėgio režimas

1.4. Tikslo funkcija

Srautinio apdorojimo sistemos valdymo tikslas – keičiant konfigūraciją, pasiekti didžiausią greitaveiką. Šiame tyrime greitaveika matuojama vėlinimu, tačiau tuo pačiu turi būti palaikomas sistemos stabilumas ir duomenų pralaidumas. Todėl pasirinkti konfigūracijos elementai su koeficientais, kurie nurodo tam tikros metrikos svarbą pasiekti greitaveikai.

3 lentelė. Metrikų tikslai

Metrika	Koeficientas	Tikslas
Išsiųstas kiekis	2	∞
Apdorotas kiekis	2	∞
Vidutinis apdorojimo vėlinimas	3	0
Naudojamas RAM	1	Išskirtas RAM – Naudojamas RAM = 0
Vidutinė GC trukmė	1	0
Bendra priešslėgio trukmė	2	0

3 lentelėje pateiktos metrikos, šių metrikų svarumo koeficientas ir jų tikslas. Metrikų koeficientas pasirinktas pagal tai, jog visų pirma yra optimizuojama turėti kuo mažesnį vėlinimą.

2. Balansavimo algoritmas

2.1. 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 Soft Actor Critic.

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

- Srautinio apdorojimo sistemos metrikos (2 lentelė)
- Srautinio apdorojimo sistemos pradinė konfigūracija (4 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 Soft Actor Critic [HZH+19] turi bendrus veikimo bruožus:

- Jie yra lasivo nuo modelio (angl. model–free) tipo skatinamojo mokymosi algoritmai, tai yra - jie daro sprendimus pagal tai, ką išmoko veikimo metu ir negeneruoja bendro modelio. 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 pasrinkti pagal dabartinę strategiją.

Šie algoritmai buvo pasirinkti, nes skiraiasi kaip apibrežiamos veiksmų aibės – Deep Q Network galimų veiksmų aibę priima kaip rinkinį, o Soft Actor Critic kaip intervalą, todėl naudojant šiuos tinklus galimų konfigūruojamų parametrų aibė apsirašo skirtingai.

2.2. 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 atpildą.

Srautinio apdorojimo sistemos dabartinė būsena – srautinio apdorojimo sistemos metrikos patalpintos masyve.

Srautinio apdorojimo sistemos balansavimo galimų veiksmų aibė – konfiguracijos parametrai, kuriuos algortimas gali koreguoti ir jų reikšmių aibė. Kadangi Deep Q Network reikalauja diskrečių reikšmių, o Soft Actor Critic testinių reikšmių aibės, jos skirtingai paduodamos į balansavimo algoritmą, tačiau reikšmių ribos vienodos.

4 lentelė. Galimų veiksmų aibė

Parametras	Naturaliųjų reikšmių aibė
component–parallelism=[skaičiavimo komponento pavadinimas]	$[1; 10] * $ 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	$[0; 20]$

Srautinio apdorojimo sistemos balansavimo žingsnio funkcija:

- Atliekamas veiksmas – naujos konfigūracijos įkėlimas ir nustatyto laiko laukimas, kad srautinio apdorojimo sistema pasiektų apkrovą.
- Atpildas – skaitinė reikšmė apskaičiuotą naudojant normalizuotas metrikas (apibrėžtas 2 pav.) su koeficientais ir 1 formulę.

$$\begin{aligned} \text{Atpildas} = 2 * \text{Išsiųstas kiekis } + 2 * \text{Apdorotas kiekis} - \\ 3 * \text{Vidutinis apdorojimo vėlinimas} - \text{Naudojamas RAM} - \\ \text{Vidutinė GC trukme} - 2 * \text{Bendra priešslėgio trukmė} \end{aligned} \tag{1}$$

2.3. 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 mašinio 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, atpildo ir naujos būsenos rinkinių. Pakartojimo buferio aibė saugoma viso mokymosi metu ir yra konfigūruojamas mokymosi metu pasirenkamos aibės dydis.

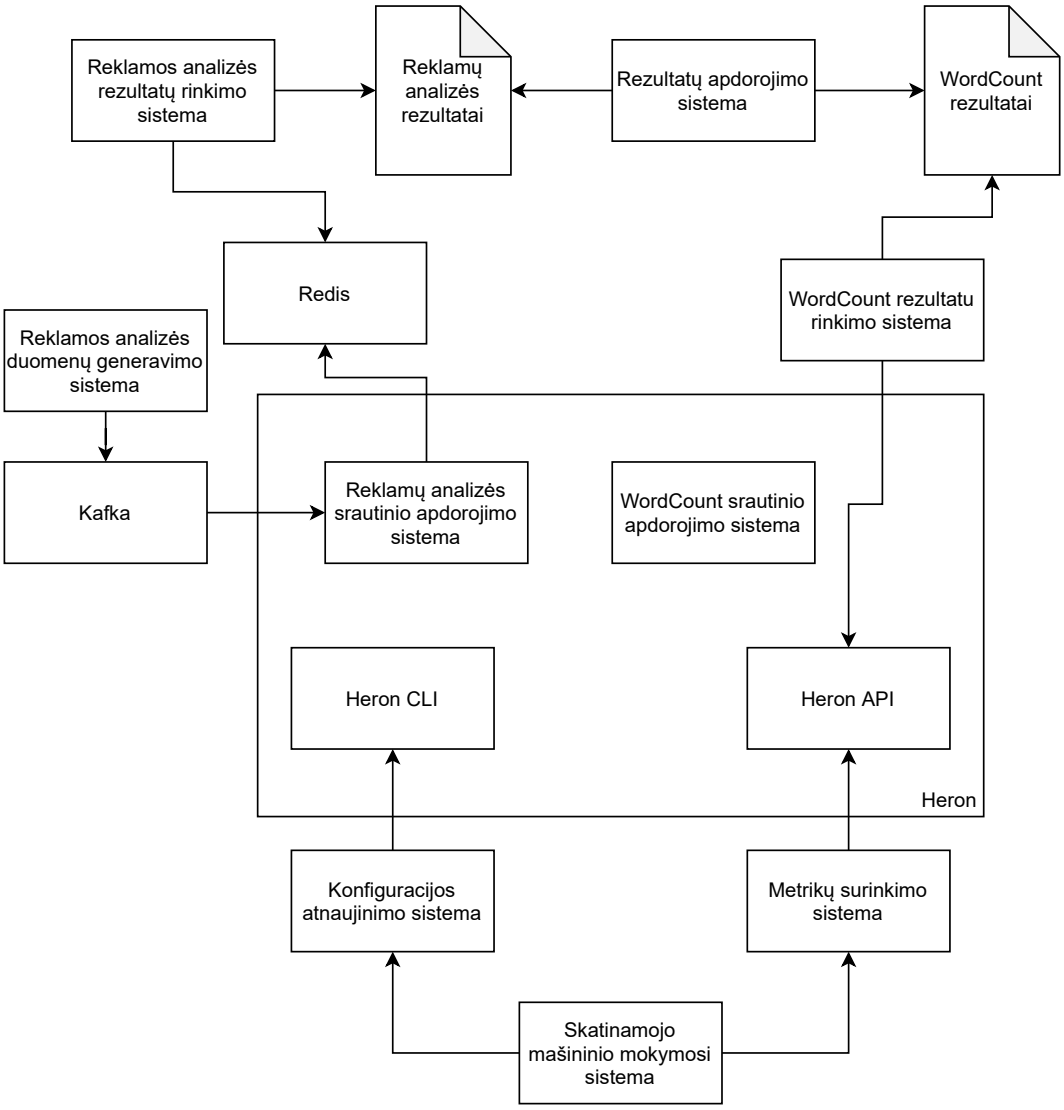
3. Eksperimento tyrimo planas

3.1. 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.

3.2. Eksperimentinė sistema

Eksperimentui atlikti reikia paruošti Heron aplinką ir sukurti reikiamus elementus. Pilna eksperimento sistema pavaizduota 2 paveikslėlyje.



2 pav. Eksperimento sistemos architektūra

Sistemos, kurios reikalingos ekeperimetui:

1. Skatinamojo mašininio mokymosi sistema. Kadangi bus atlikti bandymai su dviem skatinamojo mokymosi algoritmais: Deep Q Network ir Soft Actor Critc, bus parašyti du mašininio mokymosi agentai naudojant Python su PyTorch biblioteka. Taip pat bus sukurta aplinka aprašanti srautinio apdorojimo sistemos būseną, veiksmų aibę ir konfiguracijos žingnį.
2. Metrikų surinkimo sistema parašyta su Python naudojanti HTTP, kad gauti būsenos duomenis iš Heron API.
3. Konfigūracijos atnaujinimo sistema parašyta su Python, kuri per Heron CLI pateikia atnaujintą konfigūraciją.
4. Reklamų srautinio apdorojimo sistema parašyta su JAVA, gaunanti duomenis iš Kafka žinučių eilės ir sauganti rezultatus Redis duomenų bazėje.
5. WordCount srautinio apdorojimo sistema parašyta su JAVA, pati generuojanti duomenis ir neperduodanti duomenis į išorę.
6. WordCount rezultatų surinkimo sistema parašyta su Python skaitanti vėlinimo metrikas iš Heron API ir sauganti rezultatus tekstiniame faile.
7. Rezultatų apdorojimo sistema parašyta su Python, kuri surenka duomenis iš rezultatų failų, apdoroja juos ir grąžina skirtingų sprendimų diagramas.
8. Reklamos analizės rezultatų rinkimo sistema parašyta su Clojure ir pateikta kartu su Reklamos analizės greitaveikos testu.
9. Reklamos analizės duomenų generavimo sistema parašyta su Clojure ir pateikta kartu su Reklamos analizės greitaveikos testu.

Kompiuterinės įrangos su kuria bus 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

3.3. Planuojamų eksperimentų apimtis

Magistro darbe eksperimentai bus atliekami 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.
  - Soft Actor Critic algoritmą skatinamojo mokymosi posistemėje.

Kiekvienas sprendimas bus testuojamas su dviem srautinio apdorojimo sistemų implementacijomis:

- Reklamų analizės srautinio apdorojimo sistema.
- WordCount 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 duombazę. WordCount sistemos rezultatai bus skaitomi tiesiai iš Heron platformos naudojant Heron API ir saugomi į failą tokiu pačiu formatu kaip ir reklamų analizės sistema, naudojant absoliutų vėlinimą gauta iš Heron API.

Eksperimentai su visomis sistemomis bus vykdomi po 10 valandų, kadangi [VC18] straipsnio autoriai nustatė, kad jų balansavimo algoritmas konverguoja po maždaug 11 valandų. Gauti rezultatai iš failo bus skaitomi Python programa, kuri apdoroja duomenis į dvi grupes: visų sprendimų vėlinimą ir visų sprendimų vėlinimo 99–tą percentilę ir sugeneruoja linijines diagramas, kiekvienai vėlinimo grupei pagal sprendimą.

Taigi, iš viso magistro darbe planuojama atlikti 8 eksperimentus, kurių bendra trukmė – 80 valandų.

Rezultatai ir išvados

Rezultatai

1. Sukurtas srautinio apdorojimo sistemos valdymo modelis, naudojantis skatinamojo mašininio mokymosi algoritmą.
2. Apibrėžti tyrimui naudojami balansavimo algoritmai. Susietos srautinio apdorojimo metrikos su algoritmo įeitimis. Susieti konfigūracijos parametrai ir algoritmų galimų veiksmų aibė.
3. Aprašyti eksperimentai ir reikalavimai eksperimentinei sistemai, kuri įgyvendina sukurtą modelį.

Išvados

- Sukūrus modelį nustatyta, jog būsenos duomenys yra pateikiami atskiriems skaičiavimo komponentams, o ne visai srautinio apdorojimo sistemai, todėl galima balansuoti sistemą keičiant indvidualių skaičiavimo komponentų konfigūraciją pagal jų metrikas, nekeičiant visos sistemos konfigūracijos.
- Sukurtas srautinio apdorojimo sistemos, balansuojamos mašininiu mokymu, modelis yra įgyvendinamas naudojant skatinamojo mokymosi algoritmus Deep Q Network ir Soft Actor Critic. Sukurto modelio validumą ir efektyvumą įmanoma patikrinti su apibrėžta eksperimentine sistema.

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.

[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.

[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.

[HZH+19] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker ir k.t. Soft actor-critic algorithms and applications, 2019. arXiv: 1812.05905 [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.

[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].