VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMŲ KATEDRA

Blokų grandinių duomenų bazės finansinių duomenų kaupimui

Blockchain databases for financial data

Kursinis darbas

Atliko: 3 kurso 3 grupės studentas
Matas Savickis (parašas)

Darbo vadovas: dr. Vytautas Valaitis (parašas)

Vilnius – 2019

TURINYS

Įvadas

Per pastaruosius keletą metų blokų grandinių technologija susilaukė didelio žmonių susidomėjimo. Šis susidomėjimas daugiausiai kilo dėl išpopulerėjusių kriptovaliutų, tokių kaip Bitcoin, Etherium, Litecoin ir daugeliu kitų kurios ir yra paremtos blokų grandinių technologija. Šią technologiją 2008 metais sukūrė Satošis Nakamoto [Nak08]. 2009 metais Nakamoto implementavo blokų grandinių technologiją sukurdamas Bitcoin kriptovaliutą [Nak09]. Nors, šiuo metu, žmonių susidomėjimas kripto valiutomis ir yra sumažėjęs [Goo19], tačiau informacinių technologijų industrija mato daugiau blokų grandinių panaudojimo atveju negu tik kripto valiutos. Vienas iš blokų grandinių panaudojimo atvėjų yra blokų grandinių duomenų bazės. Reliacinės ir dokumentų duomenų bazės ilgą laiką buvo pagrindinis duomenų saugojimo būdas. Tačiau šios duomenų bazės turi ir savo trūkumų, saugant duomenis tradicinėsė duomenų bazėse kyla duomenų integralumo problemos [EW81] . Naudojant duomenų bazes financinėms transakcijoms sekti kyla dvigumo pinigų išleidimo problema[Hoe08] . Naudojantis tradicinėmis duomenų bazėmis taip pat kyla pasitikėjimo problema, visa duomenų prieeiga yra trečiosios šalies valdžioje, ir vartotojas turi pasitikėti, kad duomenys nebus pakeisti be jo žinios. Per pastaruosius kelis metus šias problemas buvo stengtasi išspręsti kuriant duomenų bazes paremtas blokų grandinių technologija. Privačios blokų grandinių duomenų bazės užtikrina pasitikėjimą, nes kiekvienas vartotojas turi visą duomenų bazės kopiją. Darant pakeitimus tokioje duomenų bazėje kiekvienas vartotojas turi sutikti su daromais pakeitimais ir saugo visų pakeitimų istoriją. Blokų grandinių duomenų bazės išsprendžia duomenų integralumo ir dvigumo pinigų išleidimo problemą, nes kiekvienas mazgas blokų grandinėje tinkle gali palyginti savo turimus duomenis su kitais mazgais. Nors blokų grandinės išsprendžia autoriaus išvardytas problemas, tačiau finansinėms transakcijos svarbus ir greitis. Šiuo darbu siekia atlikti blokų grandinių duomenų bazių analizę greičio aspektu.

Uždaviniai

1. Palyginti Cassandra NoSQL transakcijų praeinamumo greičius su Hyperledger Fabric blokų grandinių duomenų baze
2. Palyginti Cassandra NoSQL transakcijų vėlavimą su Hyperledger Fabric blokų grandinių duomenų bazėmis
3. Išskirti esamų tyrimų trūkumus ir galimas sritis ateities darbams

1. Literatūros analizė

1.1. Hyperledger

Hyperleadger yra atviro kodo blokų grandinės pradėtos pradėtos kurti Linux fondo. Šio projekto tikslas yra gerinti tarpindustrini bendradarbiavimą kuriant blokų grandines kurios užtikrintų patikimą, greitą ir saugų finansinių duomenų perdavimą pagrindinėse technologijų, finansų ir produktų tiekimo kompanijose [Hyp16]. Šiame skirsnyje bus apžvelgti Hyperledger Fabric, populiariausios Hyperledger implementacijos, architektūra, greičio tyrimai, kaip šie tyrimai buvo atlikti, kokie parametrai įtakoja Hyperledger greitį ir pateikta tolimesnės analizės pasiūlymai.

1.1.1. Architektūra

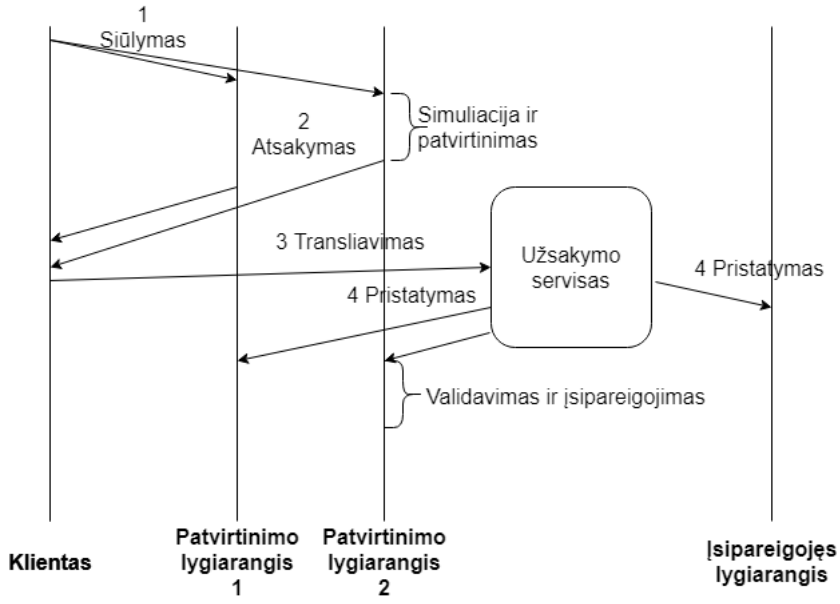
Hyperledger Fabric(toliau - Fabric) yra privati blokų grandinė skirta įmonių lygio aplikacijoms. Ši blokų grandinė gali vykdyti arbitrišką išmanujį kontraktą parašyta Java, Go arba NodeJS kalbomis (grandinės kodai). Fabric sudaro šios esybės:

- Lygiarangis - šis mazgas yra atsakingas už grandinės kodą kurį vykdant yra įgyvendinamas išmanusis kontraktas. Lygiarangis savyje turi visą tinklo informaciją(angl. Ledger).
- Užsakymo servisas (angl. Ordering Service) - Užsakymo serviso mazgas dalyvauja susitarimo(angl. consensus) protokole ir padalina bloką taip, kad jį būtų galima naudoti tranzakcijom. Padalintas blokas būna persiunčiamas kitiems lygiarangiams.
- Klientas - atsakingas už transakcijos siūlymo sukūrimą, ir išsiuntimą tinklo lygiarangiams. Gavus patvirtinimą iš lygiarangių vartotojas siunčia prašymą tvarkytojui, kad jis informaciją įtrauktų į bloką ir išsiūstų ją visiems tinklo perams.

1.1.2. Tranzakcijų vykdymas

Tranzakcijos vyksta trejomis fazėmis(1 pav.):

1. Patvirtinimo fazė - simuliuojama tranzakcija su pasirinktais vienarangiais ir renkami būsenos pokyčiai
2. Užsakymo fazė - užsakomos tranzakcijos per susitarimo protokolą
3. Patvirtinimo fazė - patvirtinimas ir informacijos įdėjimas į buhalterinę knygą



1 pav. Tranzakcijų sekų diagrama

1.1.3. Parametrai

Yra grupė parametrų [Par18] kurie įtakoja Fabric greitį

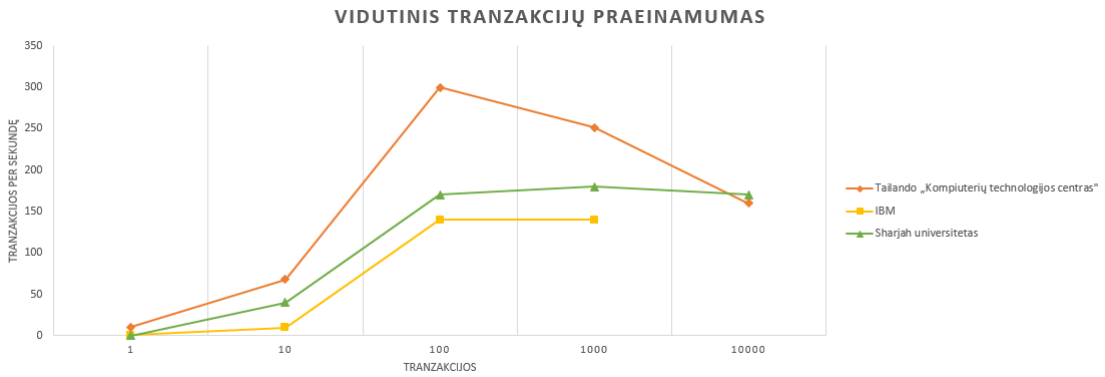
- Mazgų skaičius - vartotojų skaičius privačioje blokų grandinėje
- Tranzakcijų skaičius - keičiant tranzakcijų skaičiū keičiasi vėlavimas ir pralaidumas
- Bloko dydis - tranzakcijos yra sugrupuojamos į blokus. Blokai yra siunčiami visiems tinklo perams. Užsakymo parašas būna verifikuojamas kiekvienam blokui, o perdavimo patvirtinimo parašas verifikuojamas kiekvienai transakcijai, todėl keičiant bloko dydį atsiranda kompropisas tarp palaidumo ir vėlavim
- Patvirtinimo politika - diktuoja kiek tranzakcijų ir pasirašymų turi būti įvykdyta prieš siunčiant tranzakcijas užsakytoją. Didinant politikos sudėtingumą didės resursų sunaudojimas ir įvertinimo laikas.
- Kanalai - izoliuoja tranzakcijas viena nuo kitos ir norint persiųsti tranzakcijas iš vieno kanalo į kitą turi būti patvirtintos, surikiuotos ir apdorotos nepriklausomai viena nuo kitos.
- Resursų paskirstymas - kiekvienas lygiarangis vykdo grandinės kodą skirtą parašo skaičiavimams ir verifikavimo rutinoms. Keičiant procesoriaus branduolių skaičių keičiasi vykdymo greitis.
- Būsenos duomenų bazė - Fabric naudoją dvi duomenų bazes, CouchDB ir GoLevelDB, kuriuose galima saugoti ledgerio buseną

1.1.4. Testavimo metodologijos

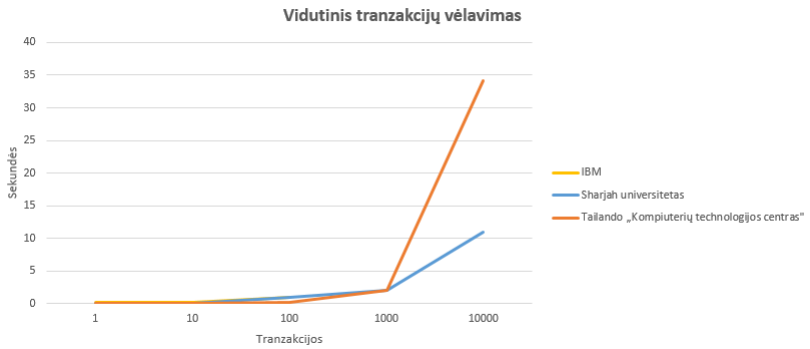
Šiame darbe apžvelgta trijų straipsnių duomenys, juose naudota tokia kompiuterinė įranga:

[Par18]	x86 64 virtuali mašina IBM SoftLayer duomenų centre. Kiekvienai virtualiai mašinai yra alokuota 32 vCPUs Intel(R) Xeon(R) CPU E5-2683 v3 @ 2.00GHz ir 32 GB atminties. Trys kliento mašinos skirtos generuoti apkrovą buvo alokuota 56 vCPU ir 128 GB RAM. Mazgai prijungti prie 3 Gbps duomenų centro tinklo
[ST17]	Amazon AWS EC2 su Intel E5-1650 8 branduolių CPU, 15GB RAM, 128GB SSD
[ShaFabPerf]	HPC serveris su Intel(R) Xeon(R) CPU E5-2690, 2.60 GHz, 24 core CPU, 64 GB RAM, and running Ubuntu 16.04

1.1.5. Tyrimų rezultatai



2 pav. Tranzakcijų praeinamumas



3 pav. Tranzakcijų vėlavimas

1.1.5.1. IBM tyrimas

IBM atliktame tyrime [Par18] buvo tyriama kaip keičiant Fabric parametrus keičiasi tranzakcijų praeinamumas ir vėlavimas. Didinant tranzakcijų atvykimo kiekį nuo 20 tranzakcijų per

sekundę iki 100 praeinamumas dideja tiesiškai, o nuo 100 transakcijų per sekundę praeinamumas sustojo ir nebekilo. Bloko dydis praeinamumui įtakos neturėjo.
Mažiausias vėlavimas būna pasirinkus mažiausia bloko dydį. Keičiant tranzakcijų atvykimo kiekį nuo 25 iki 125 vėlavimas išlieka apie 0,3 sekundės ir tuomet smarkiai kyla iki 10 sekundžių tranzakcijų atvykimo kiekį pakėlus iki 150 transakcijų.
Iš šio tyrimo imsime geriausius rezultatus(2 pav. ir 3 pav.) ir vėliau lyginsime juos su MySQL ir PostgreSQL duomenų bazėmis.

1.1.5.2. Tailando „Kompiuterių technologijos centro" tyrimas

Tailando „Kompiuterių technologijos centro" atliktame darbe buvo simuliuojamos pinigų siuntimas, pinigų išdavimas ir vartotojų sukūrimas. Keliant tranzakcijų skaičių iki 100 praeinamumas kilo iki 299.85 tranzakcijų per sekundę. Didinant tranzakcijų kaičių iki 1000 praeinamumas pakito nežymiai, kaip ir IBM [Par18] atliktame darbe, ir didinant tranzakcijų skaičių iki 10000 praeinamumas krito iki 159.76 tranzakcijų per sekundę. Didinant tranzakcijas nuo 1 iki 100 vėlavimas kilo nežymiai, nuo 0.09 iki 0.17. Padidinuos tranzakcijas nuo 100 iki 10000 staiga pakilo vėlavimas net iki 34.08 sekundžių.

1.1.5.3. Sharjah tyrimas

Sharjah universiteto mokslininkų tyrime [**ShaFabPerf**] Fabric 0.6 ir Fabric 1.0 tranzakcijų praeinamumas ir vėlavimas. Nuo 10 iki 100 tranzakcijų praeinamumas kilo nuo 40 tranzakcijų per sekundę iki 165 tranzakcijų per sekundę, toliau didinant tranzakcijų skaičių praeinamumas nebekito. Toks rezultatas gautas naudojant Fabric 1.0 versija ir praeinamumas geresnis negu Fabric 0.6 versijos. Didinant tranzakcijų skaičių nuo 10 iki 1000 vėlavimas pakilo nežymiai, nuo 0.1 sekundės iki 1 sekundės, tačiau padidinus tranzakcijų skaičių iki 10000 pastebėtas didelis vėlavimo pašokimas iki 10 sekundžių. Šio darbo metodologija buvo paremta jau minėtu Tailando mokslininkų darbu [ST17]

1.1.5.4. Rezultatu apibendrinimas

Iš aukščiau aptartų darbų([Par18], [ST17], [**ShaFabPerf**]) pastebima tendencija, kad didinant tranzakcijų skaičių iki 100 praeinamumas kilo tiesiškai, o didinant transakcijų skaičių toliau praeinamumas nebekilo arba net krito([ST17]).
Didinant tranzakcijas nuo 1 iki 1000 vėlavimas praktiškai nekyla, ir tada nuo 1000 iki 10000 vėlavimas smarkiai šoktelėja.

1.2. Apache Cassandra

Apache Cassandra yra atviro kodo, paskirstyta NoSQL duomenų bazė kuri palaiko klasterizaciją bei asinchroninį be valdančiojo kompiuterio duomenų replikavimą. Šios Cassandra galimybės yra panašios į blokinių grandinių duomenų bazių galimybes, todėl šiame skyriuje apžvelgsime jau padarytus Cassandra duomenų bazės našumo tyrimus.

1.2.1. Testavimo metodologija

1.2.2. Tyrimų rezultatai

1.2.2.1. Blekinge technologijų instituto tyrimas

1.2.2.2. Coimbra politechnikos instituto tyrimas

2. Išvados

3. Priedai

3.1. Žodynas

- Grandinės kodas(angl. chaincode) - programa parašyta Go, Java arba NodeJS kalbomis kuri vykdo verslo logiką sutartą tarp tinklo narių.
- Lygiarangis(angl. peer) - tinklo dalyvis gaunantis ir siunčiantis informaciją.
- Išmanusis kontraktas(smart contract)

Literatūra

[EW81] R.C. Summers E.B. Fernandez ir C. Wood. Database security and integrity, 1981.

[Goo19] Google. Https://trends.google.com/trends/explore?date=all&geo=us&q=bitcoin,etherium,litecoi 2019.

[Hyp16] Hyperledger. Linux foundation's hyperledger project announces 30 founding members and code proposals to advance blockchain technology, 2016.

[Hoe08] Jaap-Henk Hoepman. Distributed double spending prevention, 2008.

[Nak08] S. Nakamoto. A peer-to-peer electronic cash system, 2008.

[Nak09] S. Nakamoto. Bitcoin: a peer-to-peer electronic cash system, 2009.

[Par18] Balaji Viswanathan Parth Thakkar Senthil Nathan N. Performance benchmarking and optimizing hyperledger fabric blockchain platform, 2018.

[ST17] Chaiyaphum Siripanpornchana Suporn Pongnumkul ir Suttipong Thajchayapong. Performance analysis of private blockchainplatforms in varying workloads, 2017.