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
4. Atlikti simuliaciją lyginančia Cassandra NoSQL praeinamumą ir vėlavimą su Hyperledger Fabric grandinių duomenų baze

# 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

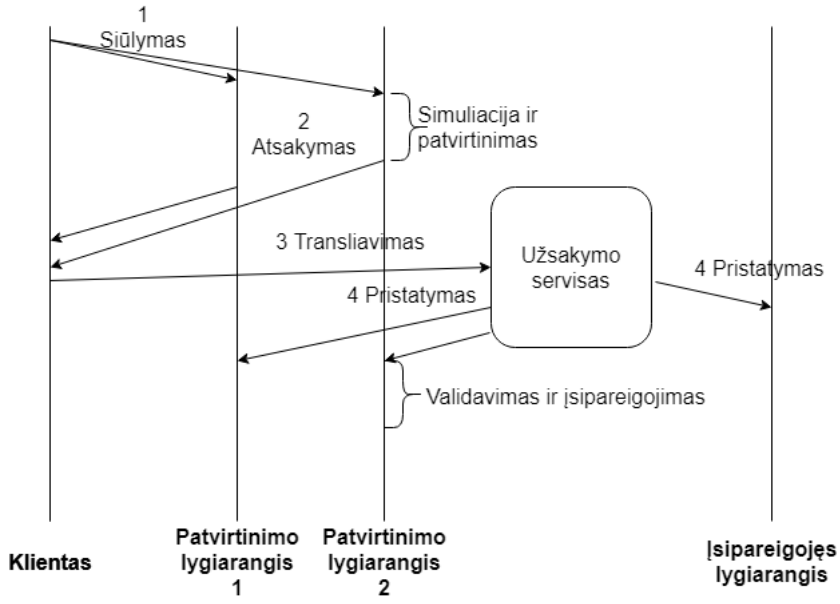
Hyperledger Fabric(toliau - Fabric) yra privati blokų grandinė skirta įmonių lygio aplikacijoms. Ši blokų grandinė gali vykdyti arbitrišką išmanujį kontraktą parašyta Java, Go arba NodeJS kalbomis (grandinės kodai). Fabric sudaro šios esybės:

- 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

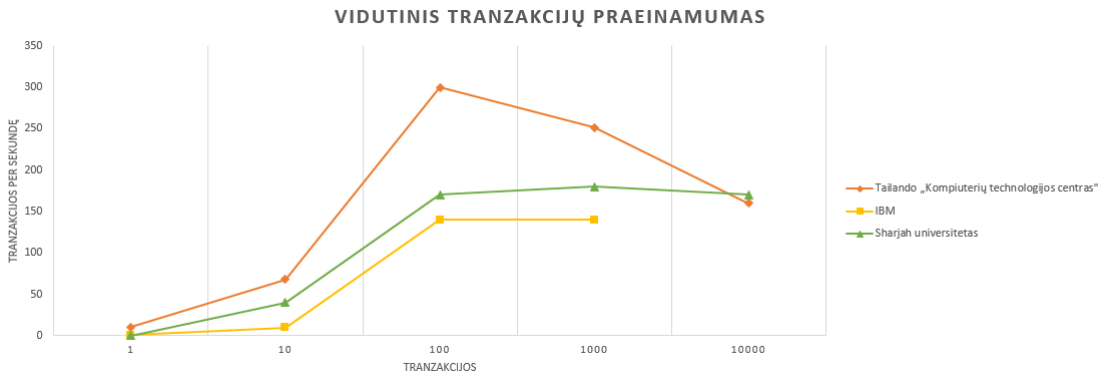
- Mazgų skaičius - vartotojų skaičius privačioje blokų grandinėje
- Tranzakcijų skaičius - keičiant tranzakcijų skaičiū keičiasi vėlavimas ir pralaidumas
- Bloko dydis - tranzakcijos yra sugrupuojamos į blokus. Blokai yra siunčiami visiems tinklo perams. Užsakymo parašas būna verifikuojamas kiekvienam 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

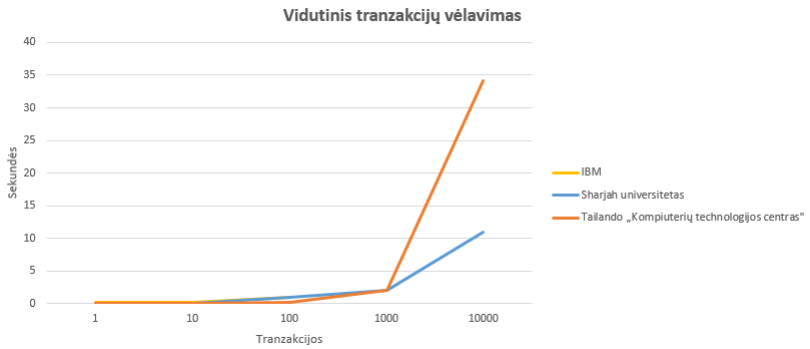
Šiame darbe apžvelgta trijų straipsnių duomenys, juose naudota tokia kompiuterinė įranga:

| | |
|---|---|
| [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.

### 1.1.6. Palyginimas pagal CAP teoremą

#### 1.1.6.1. CAP teorema

Eric A. Brewer 2000 [**CAP**] metais pristatė savo teoremą sakančia, kad tinklinis servisas negali užtikrinti trijų savybių vienu metu:

- Neprieštaringumas(Consistency) - kiekviena skaitymo užklausa gauna naujausią informaciją arba klaidos pranešimą.
- Pasiekiamumas(Availability) - kiekviena užklauso gauna atsakymą, be garantijos, kad atsakyme bus naujausias įrašas.
- Skaidinių toleravimas(Partition tolerance) - sistema nesustoja funkcionuoti jeigu būna prarandamas arbitrišką skaičių žinučių

Šios teoremos įrodymas pateikiamas Seth Gilbert ir Nancy Lynch straipsnyje [**CAPP**] Kuriant paskirstytasias sistemas, pagal panaudojimo atvejį, svarbu pasirinkti kuriuos du principus tenkins kuriama paskirstytoji sistema. Sekančiame skirsnyje bus apžvelgiama Hyperledger Fabric ir Cassandra atitikimas pagal CAP teoremą

#### 1.1.6.2. Hyperledger Fabric pagal CAP teoremą

Romos ir Southampton universitetų mokslininkų darbe [**BCCAP**] buvo tiriama kaip skirtingi susitarimo algoritmai įtakoja Etherium blokų grandinę pagal CAP teoremą. Šiame darbe CAP buvo pateiktas kitos, labiau blokų grandines atitinkantis CAP teoremos apibrėžimas:

- Neprieštaringumas(Consistency) - blokų grandinė yra neprieštaringa, jeigu yra išvengta išsišakojimų. Neprieštaringumas yra pasiekiamas susitarimo algoritmų. Jeigu neprieštaringumas nebūna pasiekiamas turime nurodyti, ar jis bus pasiektas vėliau(galiausiai neprieštaringas) ar nebus pasiektas niekada(prieštaringas)
- Pasiekiamumas(Availability) - blokų grandinė yra pasiekiama, jeigu klientų pateiktos tranzakcijos yra apdorojamos ir galiausiai patvirtinamos ir visam laikui pridedamos prie grandinės
- Skaidinių toleravimas(Partition tolerance) - kai įvyksta tinklo skaidymasis, valdžia yra paskirstoma į atskiras grupes taip, kad skirtingos grupės negali komunikuoti tarpusavyje. Blokų grandinė turi toleruoti 1. Periodus kai tinklas veikia asinchroniškai 2. Atitinkama skaičių Bizantinių valdžių siekiančių sutrigdyti pasiekiamumą ir neprieštaringumą.

Naudojantis šiais CAP apibrėžimais sekančiuose skyriuose bus įvertinta Hyperledger fabric blokų grandinių duomenų bazė.

#### 1.1.6.3. Hyperledger neprieštaringumas

#### 1.1.6.4. Hyperledger pasiekiamumas

#### 1.1.6.5. Hyperledger skaidinių toleravimas

#### 1.1.6.6. Cassandra neprieštaringumas

Kai duomenys yra rašomi į duomenų bazę užtrunka laiko duomenys pateks į visus tinklo mazgus. Kaikurie mazgai gali būtų nepasiekiami. Cassandra yra „galiausiai neprieštaringa” duo-

menų baze, nėra užtikrinta, kad duomenys kurie yra skaitomi yra naujausios versijos visame tinkle. Cassandra duomenų bazėje yra įgyvendintas pasirenkamasis pasiekiamumas kai klientas pats gali nurodyti kokios lygio neprieštaringumo jis nori skaitant duomenis(mazgų skaičius kurie turi identiškus duomenis) ir rašant duomenis(į kiek mazgų bus įrašyti duomenys). Cassandra duomenų bazėje nėra užtikrinamas neprieštaringumas, nes jeigu kažkuris mazgas yra nepasiekiamas vykdant skaitymą arba rašymą ir jį pasiekti būtina siekiant užtikrinti neprieštaringuma operacija neįvyks

#### 1.1.6.7. Cassandra pasiekiamumas

Rašant duomenis į Cassandra duomenų bazę yra padaromos kelios kopijos ir išsiunčiamos skirtingies klasterio mazgams. Tai yra užtikrinamas, kad jeigu mazgas tampa nebepasiekiamu duomenys nebūna prarandami.[**CasDesk**] Duomenų replikavimo lygį galima nurodyti kuriant duomenų bazę.

#### 1.1.6.8. Cassandra skaidinių toleravimas

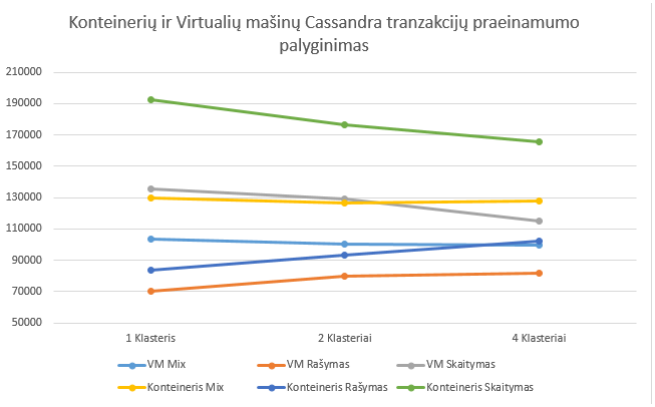
## 1.2. Apache Cassandra

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.

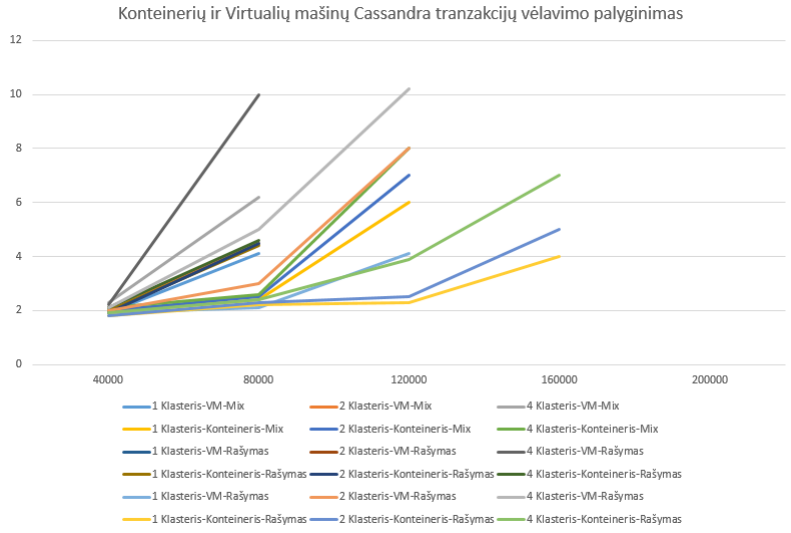
### 1.2.1. Testavimo metodologija

| | |
|---|---|
| [Sog19] | Testai buvo atlikti su trimis HP serveriais DL380 G7 iš viso su 16 branduolių (naudojant HyperThreading) ir 64 GB RAM ir HDD 400 GB. Red Hat Enterprise Linux Server 7.3 (Maipo) (Kernel Linux 3.10.0-514.e17.x86 64) ir Cassandra 3.11.0 yra įdiegta į visus kompiuterius, taip pat ir virtualias mašinas. Ta pati Cassandra versija naudojama ir apkrovos generavimui. Konteinerių testavimui naudota, Docker versija 1.12.6. Virtualioms mašinoms naudota VMware ESXi 6.0.0. |
| [Sog19] | Testai buvo leidžiami naudojant Ubuntu Server 12.04 32bit Virtual Machine leidžiant ant VMware Player. Virtuali mašina turėjo 2GB RAM ir priimančioji mašina turi vieno mazgo Core 2 Quad 2.40 GHz su 4GB RAM ir Windows 7 operacine sistema. Duombazių versijos: MongoDB version 2.4.3 ir Cassandra version 1.2.4. |

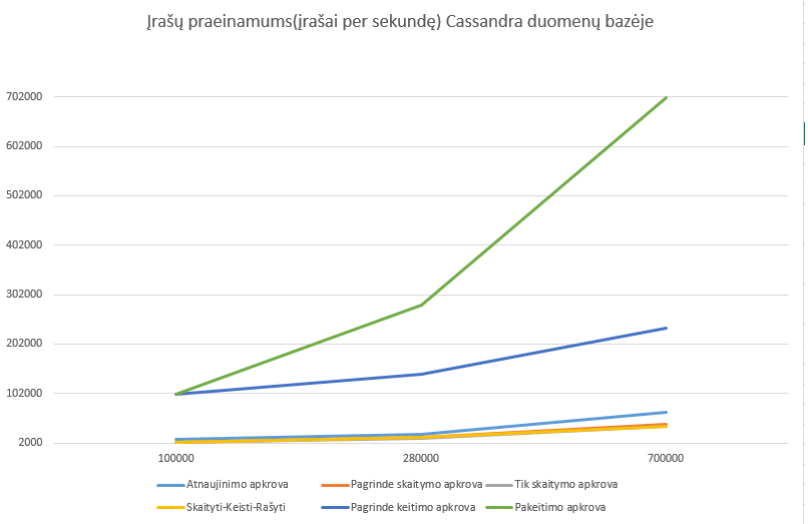
### 1.2.2. Tyrimų rezultatai




4 pav. Tranzakcijų vėlavimas




5 pav. Tranzakcijų vėlavimas




6 pav. Tranzakcijų vėlavimas

#### 1.2.2.1. Blekinge technologijų instituto tyrimas

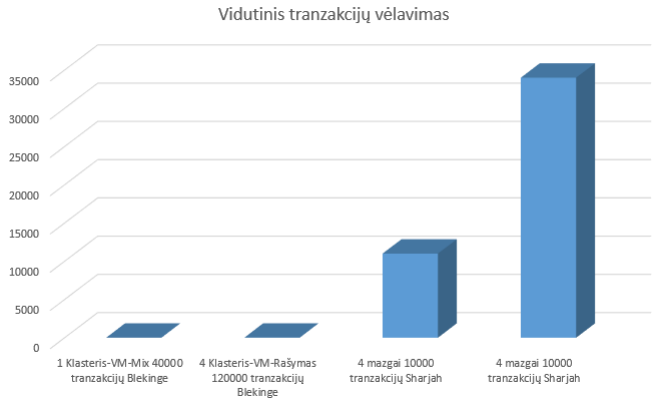
Blekinge technologjų instituto atliktame tyrime [Sog19] buvo atliktas Cassandra duomenų bazės našumo palyginimas lyginant duomenų bazę veikiančia konteineriuose ir virtualioje mašinoje. Buvo lyginama tranzakcijų praeinamumas ir vėlavimas keičiant klasterių skaičių. Įvairioje apkrovoje virtualios mašinos praeinamumas laikėsi apie 100000 tranzakcijų per sekundę, rašymo apkrovoje apie 75000 tranzakcijų per sekundę, skaitymo apkrovoje apie 120000 tranzakcijų per sekundę. Įvairioje apkrovoje konteinerių praeinamumas buvo apie 127000 tranzakcijų per sekundę, rašymo apkrovoje apie 95000 tranzakcijų per sekundę, o skaitymo aprovoje apie 180000 kranzakcijų per sekundę. Virtualiose mašinose didinant klasterių skaičių padidėjo praeinamumas didėjo su rašymo apkrova ir mažėjo su skaitymo apkrova. Tas pats buvo pastebėta ir konteineriuose.
Tyrime mažiausias vėlavimas buvo naudojant vieną Cassandrą klasterį su 40000 tranzakcijų apkrova, o didžiausias vėlavimas buvo pastebėtas naudojant keturis Cassandra klasterius veikiančius ant virtualių pašinų naudojant 120000 tranzakcijų apkrovą. Viso tyrimo metu buvo pastebėtas vėlavimas intervale nuo 1 iki 11 milisekundžių.

#### 1.2.2.2. Coimbra politechnikos instituto tyrimas

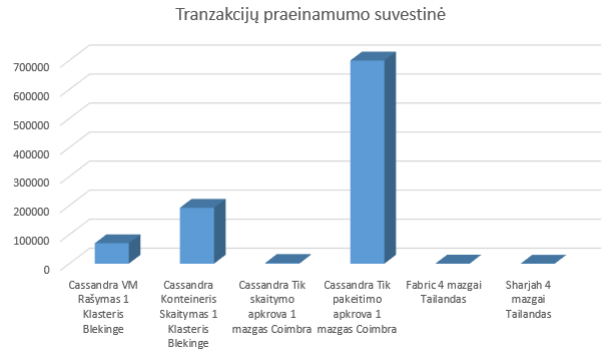
Coimbra politechnikos instituto atliktame tyrime [Ver13] buvo lyginamos Cassandra ir MongoDB duomenų bazės. Lyginimas buvo atliktas naudojant tik vieną mazgą todėl buvo lygintas tik įrašų(tranzakcijų) praeinamumas su skirtingomis apkrovomis. Šio darbo tikslui žiūrėsime tik į rezultatus gautus apie Cassandra duomenų bazę. Pastebėta, kad didinant apkrovą tranzakcijų praeinamumas padidėjo. Didžiausias praeinamumas buvo pasiektas kai buvo atlikta tik vieno tipo, o ne miksuotos duomenų operacijos. Aukšiausias duomenų pakeitimo praeinamumas buvo 700000 tranzakcijų per sekundę, o skaitymo praeinamumas buvo 35000 tranzakcijų per sekundę. Mažiausias duomenų praeinamumas buvo pastebėtas su mažesnia apkrova. Žemiausias duomenų pakeitimo praeinamumas buvo 100000 tranzakcijų per sekundę, o žemiausias skaitymo praeinamumas buvo 2325 tranzakcijų per sekundę.
Pagrindinis šių tyrimų palyginimo trūkumas yra tai, kad visi šiame darbe apžvelgti tyrimai buvo atlikti su skirtingo galimgumo sistemomis ir skirtingomis duomenų bazių konfigūracijomis. Ieškant litratūros šiam darbui kilo sunkumų surasti tyrimų kuriuose su ta pačia ar panašia kompiuterine įranga ir duomenų bazių konfigūracijomis tiesiogiai būtų lyginamos blokų grandinių duomenų bazės su NoSQL duomenų bazėmis. Sekančiame skirsnyje bus atliekamos Fabric ir Cassandra našumo tyrimas.

### 1.2.3. Palyginimas pagal CAP teoremą

## 1.3. Literatūros apibendrinimas




7 pav. Tranzakcijų vėlavimas




8 pav. Tranzakcijų vėlavimas

Apžvelgtuose darbuose buvo matuojamos Apache Cassandra ir Hyperledger Fabric duomenų bazių praeinamumas ir vėlavimas. Palyginus rezultatus buvo aiškiai matoma, kad Cassandra praeinamumas buvo didesnis ir vėlavimas mažesnis. Net ir palyginus prasčiausią Cassandra praeinamumo rezultatą (2325 tranzakcijų per sekundę) ir geriausią Fabric rezultatą (300 tranzakcijų per sekundę) Cassandra duomenų bazės praeinamumas buvo geresnis 775 procentais. Lygynant prasčiausią Cassandra ir geriausią Fabric vėlavimo rezultatą Cassandra vėlavavimas buvo 1078 procentų mažesnis negu Fabric.

# 2. Simuliacija

# 3. Išvados

# 4. Priedai

## 4.1. 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.

[Sog19] Emiliano Casalicchio Sogand Shirinbab Lars Lundberg. Performance comparision between scaling of virtual machines and containers using cassandra nosql database, 2019.

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

[Ver13] Jorge Bernardino Veronika Abramova. Nosql databases: mongodb vs cassandra, 2013.