VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
STUDIJŲ PROGRAMA: INFORMATIKA

Mokslo duomenų bazių neprieštaringumo, prieinamumo bei ilgaamžiškumo tyrimas naudojant atvirą kristalografijos duomenų bazę (COD)

The study of the data consistency, availability and longevity of the scientific databases based on Crystallography Open Database (COD)

Baigiamasis magistro darbas

Atliko: Darius Žvirblis
VU el. p.: darius.zvirblis@mif.stud.vu.lt

Vadovas: prof. dr. Saulius Gražulis
Recenzentas: asist. dr. Vytautas Valaitis

Vilnius
2023

Santrauka

Tiriamajame darbe nagrinėjamas mokslo duomenų išsaugojimas. Numatomos procedūrinės, technologinės, organizacinės ir natūraliosios priežastys, dėl kurių duomenys gali būti prarasti. Tiriamasis darbas atliekamas naudojant atvirą kristalografijos duomenų bazę (angl. *Crystallography Open Database*). Bus ieškomi sprendimai, kaip užtikrinti mokslo duomenų tvarumą ir ilgaamžiškumą, bei įgyvendinti sėkmingą realizaciją, kuri užtikrintų mažesnę duomenų praradimo tikimybę. Numatoma serverių spiečiaus realizacija galima vadinti išskirstyta sistema, todėl bus atliekamas tyrimas, kaip apeiti CAP teoremos apribojimus remiantis mokslo duomenų bazės COD ypatumais.

Raktiniai žodžiai: Mokslo duomenų išsaugojimas, CAP ir CAL teoremos, FAIR, atkartojamumo problema, išskirstytos sistemos

Summary

The preservation of scientific data is examined in this research work. Procedural, technological, organizational and natural reasons for the loss of data are foreseen. The research work is based on the Crystallography Open Database (COD). The goal of the research work will be to ensure the sustainability and longevity of scientific data and to build a successful implementation that will reduce the data loss possibility. The expected implementation of the server hive can be called a distributed system, so a study will be consider on how to avoid the limitations of the CAP theorem based on the features of the scientific database COD.

Keywords: Scientific data, CAP and CAL theorems, FAIR, reproducibility crisis, distributed systems

Turinys

Įvadas

Mokslo duomenų išsaugojimas ir galimybė juos panaudoti ateities tyrimams yra be galo svarbus. Būtina užtikrinti, kad duomenys būtų saugomi teisingai ir juos būtų galima lengvai rasti taip kaip tai apibrėžiama FAIR [WDA+16] principuose. Ne mažiau svarbu užtikrinti, kad iš pateiktų duomenų būtų įmanoma atkartoti tyrimo rezultatus [Pen11] ir duomenys neišnyktų susiklosčius nenumatytoms aplinkybėms, tokioms kaip stichinės nelaimės, programinės ar aparatinės įrangos gedimas ar pasikeitus teisinei aplinkai. Pateikiame keli duomenų praradimo pavyzdžiai, kaip net tokia organizacija kaip „NASA“ neužtikrino mokslo duomenų išsaugojimo ir buvo prarasti aukštos kokybės „Apolo-11“ misijos įrašai [Pri10] ir tik atsitiktinumo dėka išsaugoti „NASA“ palydovo „Nimbus II“ [Pri10] duomenys. 2020 m. liepos mėn. liūtys Vilniaus mieste užliejo VĮ „Registrų centras“ serverinę, ko pasekoje ≈savaitei sutriko svarbių valstybinių informacinių sistemų veikla [Žyg20]. Kadangi sistemos buvo apsaugotos tik atsarginėmis kopijomis, buvo prarasti svarbūs duomenys, tačiau tikslus prarastų duomenų kiekis nėra skelbiamas. Darbo metu paaiškėjo, kad išorinių veiksnių įtaka galimam duomenų išnykimui yra didesnė, nei buvo manoma iki šiol ir gali nuolatos kisti. Globalūs stichinių nelaimių padariniai, nestabili ir nuolat kintanti geopolitinė situacija tik patvirtina, kad duomenų saugojimas vienos valstybės teritorijoje, net ir išskirstant po skirtingus miestus, nebėra pakankama priemonė duomenų ilgaamžiškumui ir prieinamumui užtikrinti.

Išskiriamos keturios galimos duomenų praradimo priežastys: procedūrinės, technologinės, organizacinės, natūraliosios. Toks grupavimas padeda geriau suprasti, kokiomis aplinkybėmis galimi duomenų praradimai ir kokias priemones galima taikyti, kad to išvengti.

Šis darbas atliekamas remiantis atvira kristalografijos duomenų baze (COD) [GDM+11]. Tai unikali mokslo duomenų bazė, kurioje kaupiama kristalografinė informacija CIF formato failuose. Kristalografinės informacijos failas yra standartinis teksto failo formatas, skirtas kristalografinei informacijai pateikti, paskelbtas Tarptautinės kristalografijos sąjungos. COD duomenų bazė taip pat unikali tuo, kad joje sėkmingai integruoti kitos kristalografijos duomenų bazės duomenys „CrystalEye“ [DDA+12]. Šie duomenys buvo sėkmingai perkelti ir išsaugota svarbi mokslinė informacija laimingo atsitiktinumo ir mokslininkų bendradarbiavimo dėka. Deja, šiai perkėlimo procedūrai nebuvo iš anksto numatytos informacinės priemonės ir procedūros. Šio darbo sprendiniai, bus taikomi konkrečiai mokslo duomenų bazei COD, tačiau jie galėtų būti perpanaudoti ir kitoms mokslo sritims ir jų duomenų išsaugojimui.

Šiame mokslo tiriamajame darbe bandoma rasti standartinius arba nestandartinius informatikos metodus ir priemones, kuriomis būtų pasiekiamas duomenų tvarumas ir ilgaamžiškumas. Daroma prielaida, kad geografiškai išskirstyta, daugelio mazgų autonominė sistema, šiame darbe vadinama serverių spiečius, padės pasiekti keliamus tikslus.

Darbo tikslas

Nustatyti kokie informaciniai sprendimai gali užtikrinti, kad mokslo duomenys, konkrečiai COD, bus išsaugoti bei prieinami ateities kartoms tol kol jie bus reikalingi (pvz. ≈100 metų į priekį) ir bus užtikrinamas duomenų vientisumas su mažiausiomis sąnaudomis, bei realizuoti galimą sistemos architektūrinį sprendinį.

Darbo uždaviniai

1. Atlikti COD duomenų bazės tyrimą, aprašyti jos ypatumus, bei nustatyti, kaip mokslo duomenų bazės COD ypatumai apribojami CAP ir CAL teoremomis.
2. Atlikti COD duomenų bazės tyrimą, remiantis FAIR principais.
3. Atlikti atkartojamumo problemos (angl. *Reproducibility crisis*) tyrimą ir rasti taikomus sprendimus.
4. Nustatyti kokios duomenų saugyklų architektūros turi geriausią kainos ir ilgaamžiškumo santykį.
5. Įgyvendinti sėkmingą realizaciją, kuri užtikrintų duomenų tvarumą ir ilgaamžiškumą, bei sumažintų duomenų praradimo tikimybę. Sėkminga realizacija apibrėžiama, kaip veikiantis ir ištestuotas sistemos prototipas, įgyvendintas pagal pasirinktą architektūrinį modelį.

1. Atvira kristalografijos duomenų bazė (COD)

2003 m. buvo pradėtas Atviros kristalografijos duomenų bazės (COD) projektas. Pagrindinis projekto tikslas buvo ir vis dar yra – surinkti visas žinomas mažo ir vidutinio dydžio kristalografines struktūras vienoje patikimoje, atviros prieigos duomenų bazėje. Šiuo duomenų bazėje saugomi daugiau kaip 480 000 įrašų sukauptų per daugiau kaip 15 m. laikotarpį. COD sukūrimo metu, buvo daromos 4 veidrodinės duomenų kopijos į 4 skirtingas valstybes, tačiau šiuo metu aktyviai palaikomos tik dvi duomenų bazės, pagrindinė – Vilniuje ir veidrodinė kopija esanti Ispanijoje [GDM+11]. Šiuo metu aktyviai prižiūrimos tik dvi duomenų bazės Vilniuje ir Ispanijoje. Tai tik įrodo, kad laikui bėgant didėja tikimybė prarasti svarbius tyrimų duomenis. Šioje duomenų bazėje sėkmingai integruoti kitos kristalografinės duomenų bazės „CrystalEye“ informacija, įskaitant programinę įrangą.

1.1. COD tyrimas

Norint priimti teisingus technologinius sprendimus ir sprendinius, būtina suprasti, kas yra COD. Analizuojant esamą sistemą, bus atsižvelgiama į šiuos aspektus:

1. Duomenų saugojimo ypatybės, duomenų formatas ir jų apdorojimas. Taikomi sprendimai duomenų teisingumo užtikrinimui ir dublikatų vengimas, unikalių reikšmių generavimas ir priskyrimas.

2. Skaitymo ir rašymo operacijų skaičius.

3. Taikomi technologiniai sprendimai duomenims saugoti.

Sistemos pagrindas - kristalų struktūros saugojimas CIF formato failuose [GDM+11]. CIF failai yra pirminis informacijos šaltinis. Kiekviena įkeliama struktūra, atlikus failo tikrinimo procedūras, patalpinama į jiems skirtą failų struktūros vietą ir atitinkama informacija įrašoma į duomenų bazę. Failų struktūra saugojama versijavimo sistemos pagrindu sukurtoje medžio struktūroje – repozitorijoje. Tokiu būdu yra galimybė matyti kiekvieną pakeitimą – reviziją. Kiekviena nauja struktūra įkelta į COD gauna unikalų septynių skaitmenų numerį, vadinamą COD numerį. Šis numeris identifikuoja konkrečią struktūrą. COD naudojami ir papildomi struktūrų patikrinimo metodai, siekiant išvengti dublikatų. Numatyti tokie sistemos apribojimai – duomenys įrašomi tik vieną kartą, saugomas kiekvienas pakeitimas (revizija), kiekvienai struktūrai priskiriamas unikalus identifikavimo numeris.

Remiantis sistemos analize, šiuo metu:

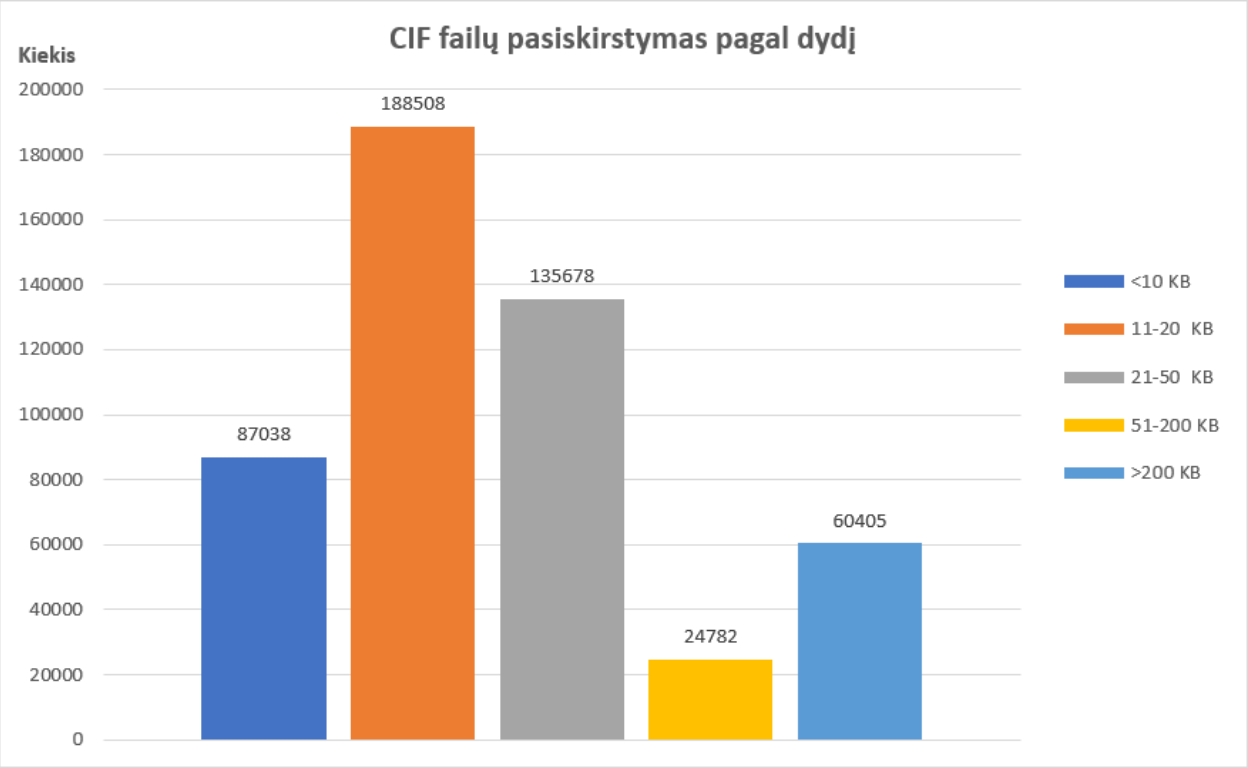
1. Per parą nuskaitoma struktūrų ≈ 450.

2. Per parą įkeliama ≈ 30 naujų struktūrų.

3. Viso saugoma ≈ 480000 struktūrų.

4. CIF failai užima ≈ 80 GB disko vietos.

Žemiau pateikiamas CIF failų pasiskirstymas Pav. 1 pagal failų dydį. Visi failai suskirstyti į penkis ruožus. Didžiosios dalies failų dydis yra tarp 11-50 kilobaitų.



1 pav. Failų pasiskirstymas pagal dydį

2. COD analizė remiantis FAIR principais

2016 m. buvo paskelbti FAIR principai. Jais siekiama pagerinti skaitmeninių tyrimų duomenų kokybę, bei prieinamumą tiek žmonėms, tiek mašinoms. Tik turint teisingus ir tvarkingus duomenis, taip kaip juos apibrėžia FAIR, juos bus galima išsaugoti ir įgalinti ateities tyrimams. FAIR yra pagrindiniai mokslinių duomenų valdymo ir tvarkymo principai [WDA+16], kurie apibrėžia:

1. Randamumą (angl. *Findability*) – duomenys privalo būti randami tiek žmonėms tiek ir automatizuotoms duomenų apdorojimo sistemoms.

2. Prieinamumą (angl. *Accessibility*) – duomenys privalo būti prieinami, tada kada jų reikia.

3. Sąveikumą (angl. *Interoperability*) – duomenys turi būti integralūs analizei, apdorojimui skirtingomis programomis.

4. Perpanaudojamumą (angl. *Reusability*) – saugojami duomenys privalo būti prieinami pakartotiniam pakartojimui.

FAIR principuose ypatingas dėmesys skiriamas mašinų gebėjimo automatiškai rasti ir naudoti duomenis, bei įgalinti juos pakartotiniam panaudojimui. Atkreipiamas dėmesys, kad skaitmeniniais duomenis laikomi ne tik patys duomenys, tačiau ir programinė įranga [LGK+20]. Pilnas duomenų rinkinio pateikiamas su programiniu kodu yra svarbus pakartotiniam duomenų panaudojimui.

Šiuo metu naudojami technologiniai ir organizaciniai sprendiniai, kurie yra panaudoti kuriant COD, užtikrina visus FAIR principus. Teisingą mokslinių duomenų saugojimo principų laikymąsi esamoje sistemoje įrodo:

1. Duomenys yra lengvai randami tiek žmonėms tiek automatizuotoms duomenų apdorojimo sistemoms. Paieškos palengvinimui ir pagreitinimui naudojam SQL tipo duomenų bazė, kurioje saugoma kristalų struktūrų meta duomenys [GDM+11].

2. Duomenys yra laisvai prienami. Instrukcijos kaip ir kokia apimtimi duomenys prieinami, laisvai pateikiama COD sistemos interneto tinklapyje.

3. Duomenys saugomi CIF (angl. *Crystallographic Information File*) [HAB91] formato failuose. Nuo 1990 m. CIF formato failai yra kaip standartas archyvuojant ar dalinantis kristalografine informacija. Standartų naudojimas yra svarbi FAIR principų užtikrinimo priemonė.

4. Duomenys pateikiami pilna apimtimi, kokia jie buvo pateikti autoriaus, taip užtikrinant galimybė pakartotiniam panaudojimui ar tyrimų atkuriamumui.

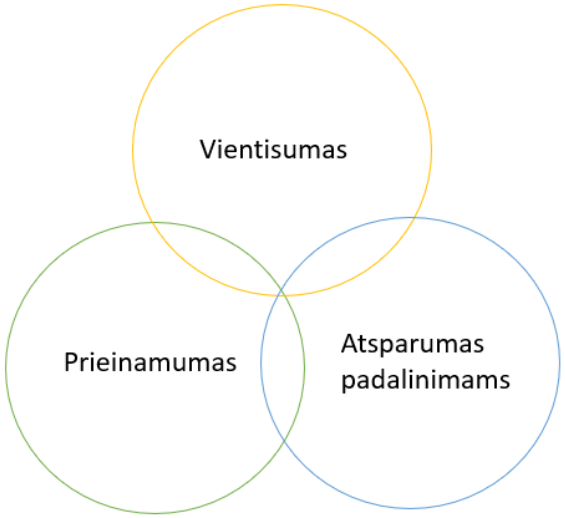
Taip pat duomenys licencijuojami pagal CC0 [Com09] licencijos tipą, kuris parodo, kad COD esantys duomenys nėra ribojami licencinėmis taisyklėmis. Privaloma užtikrinti, kad FAIR principų atitiktis bus išlaikyta projektuojamoje serverių spiečiau architektūroje.

Tai kaip rašoma [WDA+16] straipsnyje, kad teisingas duomenų tvarkymas turi perspektyvą ateityje, naujiems atradimams ir duomenų pakartotiniam panaudojimui. Teisingas duomenų tvarkymas nėra griežtai apibrėžtas ir tai yra duomenų valdytojo sprendimas.

Pačiuose FAIR principuose nėra kalbama apie konkrečius technologinius sprendimus. Jie veikia kaip vadovas duomenų skelbėjams ir prižiūrėtojams, padėti jiems įvertinti, ar dėl konkretaus įgyvendinimo pasirinkimo jie tampa randami, prieinami, sąveikūs ir daugkartinio naudojimo.

3. Serverių spiečius

Keliama hipotezė, kad serverių spiečius turėtų sumažinti duomenų praradimo tikimybę, nei esamoje sistemoje. Šiame darbe apžvelgiamos technologijos, tokios kaip atsarginis kopijavimas, blokų grandinės, ACID tranzakcinių duomenų bazių savybės, klientas-klientas sistemos, NoSQL duomenų bazės, kurios galėtų būtų naudojamos, kaip realizacijos pagrindas. Kadangi projektuojamas serverių spiečius yra išskirstyta sistema, tai sukelia papildomų sunkumų, kurie išdėstyti CAP teoremoje [Bre12]. CAP teoremoje teigiama, kad vienu galima pasiekti dvi iš trijų norimų charakteristikų (Pav. 2). Taip pat apžvelgiami išskirstytų sistemų algoritmai.



2 pav. CAP teoremos apribojimai

Projektuojant išskirstytą sistemą reikia pasirinkti tarp galimų alternatyvų, kurios gali būti:

1. Nepriešstaringumas arba vientisumas (angl. *Consistency*) – kreipiantis į bet kurį spiečiaus narį, turime gauti naujausią galimą informaciją

2. Prieinamumas (angl. *Availability*) – kiekviena užklausa gauna atsakymą, negarantuojant, kad joje yra naujausias įrašas.

3. Atsparumas padalinimams (angl. *Partition tolerance*) – sistema ir toliau veikia, nepaisant to, kad tinklas tarp mazgų atmetė (arba uždelsė) pranešimo pristatymą. Kai įvyksta tinklo padalinimas, reikia nuspręsti, ar tai padaryti atšaukiant operaciją ir taip sumažinti pasiekiamumą, bet užtikrinti nuoseklumą ar tęsti operaciją ir tokiu būdu užtikrinti prieinamumą, bet rizikuoti, kad duomenys bus nesuderinami.

Taip pat reikia atkreipti dėmesį į tai, kad serverių spiečiaus nariai galėtų būti bet kurioje pasaulio šalyje. Tai sukelia taip pat papildomų sistemos iššūkių, tokių kaip tinklo vėlinimas (angl. *Network latency*). Šiai problematikai taip pat priskiriamas vėlinimas vartotojui (angl. *User latency*) – laikas per kurį vartotojas dar matys neatnaujintus duomenis [LBL+21]. Numatoma, kad kiekvienas serverių spiečiaus narys turės jam priskirtą unikalių identifikatorių ruožą, kuriame atitinkama tvarka bus saugomos naujai įkeliamos kristalų struktūros. Serverių spiečius turėtų spręsti šias duomenų praradimo priežastis:

1. Procedūrines ir organizacines – nariai priimami ir išleidžiami pagal griežtas iš anksto suderintas taisykles. Taip pasibaigus projektui ar mokslininkui nutraukus veiklą, duomenys nebus prarasti ir tą bus galima suplanuoti ir kontroliuoti.

2. Technologines – duomenys nebus prarasti esant vieno ar kelių mazgų gedimui, priklausomai nuo to ko, kiek bus aktyvių narių. Atsparumas duomenų praradimui iš anksto suplanuotas ir kaina pamatuota pagal saugomų duomenų vertę.

3. Natūraliąsias – vieno ar kelių duomenų centrų sugadinimas, esant stichinėms nelaimėms, karui ar kitoms globalioms katastrofoms neturės įtakos duomenų praradimui, atitinkamai narių skaičiui ir priklausomai nuo geografinio paveikimo regiono.

Projektuojama, kad sistemoje nebus valdančio nario, todėl viskas bus paremta lygiais serverių duomenų mainais. Visi nariai dirbs vienodomis rolėmis. Taip bus stengiamasi išvengti priklausomybės nuo vieno nario gedimo, tačiau toks sprendimas tikėtina, kad sukurs papildomų architektūrinių ypatumų [CGH+18].

Atsižvelgiant į saugomų duomenų tipą ir sistemos paskirtį, mūsų pasirinkimas yra sistemos prieinamumas prieš vientisumą. Net jei ir yra versijų skirtumas, t.y. einamuoju momentu nėra išlaikomas vientisumas, jis visuomet artėja link 0, tai reiškia, kad sistema stengsis pasiekti vientisumą (angl. *Eventually consistent*) [Vog09]. Svarbu pabrėžti, kad saugomų įrašų versijų skirtumai nereiškia duomenų nekorektiškumo. Atsiradęs sistemos padalinimas, neturi turėti įtakos vartotojo prieigai ir jam turi būti nematomas.

Serverių spiečiui numatomi tokie apribojimai (invariantai):

1. Kai faktas apie gamtą užregistruojamas serverių spiečiuje, tai yra duomenys įrašomi duomenų bazėje, nėra svarbu kuriam ruože jis bus buvo priimtas.

2. Jei vieno ruožo spiečiaus narys išnyksta, šio ruožo duomenys neprarandami.

3. Pradinis informacijos šaltinis yra CIF failai.

4. Saugomi duomenys visuomet yra teisingai, net ir tuo atveju, kai yra naujesnė duomenų versija.

Apibendrinant, sudaromas reikalavimų sąrašas, kuris būtinas duomenų tvarumui ir ilgaamžiškumui užtikrinti:

1. Informacija griežtai įrašoma ir atnaujinama tik per CIF failus.
2. Kiekvienas spiečiaus narys turi priskirtą unikalių identifikacinių numerių ruožą.
3. Spiečiaus narys gali būti bet kurioje pasaulio vietoje.
4. Komunikacija tarp spiečiaus narių turi būti kriptografiškai apsaugota.
5. Vienu metu skaitant informaciją iš dviejų skirtingų mazgų leidžiamas revizijos skirtumas.
6. Spiečius visuomet bando pasiekti galutinį nuoseklumą (angl. *Eventually consistency*).
7. Išlaikytas COD tęstinumas ir revizijų monotoniškas didėjimas.
8. Spiečiaus nariai prijungiami ir atjungiami griežtai pagal numatytas taisykles.
9. Sprendimas turi užtikrinti ir išlaikyti FAIR.
10. Esant neaktyviam vienam ar keliems spiečiaus nariams, užklausas apdoroja bet kuris aktyvus serveris.

3.1. Priėmimo ir išleidimo taisyklės

Kiekvienas mokslininkas ar organizacija, norėdami prisijungti prie bendros serverių spiečiaus sistemos, privalo laikytis šių taisyklių:

1. Spiečiaus nariai turi būti identifikuoti ir autorizuoti.
2. Spiečiaus narys turi pritarti, pasižadėti ir sutarti laikytis mokslo duomenų kokybės standartų, sutartų COD tarybos (angl. *Advisory board*).
3. Naudoti CIF failų struktūros kokybės patikrinimo programinę įrangą ir laikytis sutartų kokybinių reikalavimų pvz.: atviras kodas.
4. Sutikti publikuoti duomenis CC0 licencija.
5. Naudoti tuos pačius minimalius kokybės kriterijus.
6. Naudoti F/LOSS licencijos programinę įrangą.

4. Metodai

4.1. Ilgaamžiškumo užtikrinimas atsarginėmis kopijomis

Pasiekti norimą duomenų ilgaamžiškumą ir atkuriamumą būtų galima naudojant atsargines kopijas, jas išdėstant skirtingose pasaulio valstybėse ar žemynuose ir atliekant atitinkamais laiko intervalais [LXD15] atsižvelgiant į leistinas sistemos prastovas. Kaip teigiama [LXD15] šis metodas orientuotas į vieno komponento-serverio, realaus laiko sistemas, siekiant užtikrinti maksimalų sistemos patikimumą, bei optimizuoti išlaikymo ar galimo atkūrimo po kritinių situacijų kaštus. Optimalus kaštų panaudojimas taip pat svarbus ir mokslo duomenų bazėms.

Kaip nurodoma [AM13] straipsnyje, kad natūraliosios priežastys sudaro didelę dalį iš visų įvykusių informacinių sistemų avarijų (1 lentelė). Kita didelė sistemų veiklos sutrikimų dalis priklausoma nuo žmogiškųjų veiksmų.

1 lentelė. Nelaimės ir jų poveikis

Priežastis	**Mastas**
Sistemų atnaujinimai	72%
Energijos tiekimo sutrikimai	70%
Gaisrai	69%
Konfigūracijos keitimai	64%
Kibernetinės atakos	63%
Piktavaliai darbuotojai	63%
Duomenų nutekėjimai	63%
Potvyniai	48%
Uraganai	46%
Žemės drebėjimai	46%
Tornadai	46%
Terorizmo aktai	45%
Cunamiai	44%
Ugnikalniai	42%
Karas	42%
Kitos priežastys	1%

Nors atsarginės kopijos galėtų būti kaip optimalus sprendimas COD, tačiau negali išpildyti keliamų sistemos reikalavimų:

1. Duomenų prieinamumas (angl. *Availability*) – duomenys nėra tiesiogiai prieinami iš atsarginių kopijų. Atstatymas iš atsarginių kopijų užima tam tikro laiko tarpą, per kurį sistema bus neprieina. COD „serverių spiečius“ turi būti prieinamas esant vieno ar kelių mazgų gedimui, atitinkamai nuo esančių aktyvių narių skaičiaus.

2. Duomenų praradimas – sudarinėjant avarinio atstatymo planus (angl. *Disaster Recovery*

Plan), kuriuose kaip priemonė gali būti ir dažnai yra atsarginės kopijos, kaip sprendimas avariniam atstatymui, visuomet vertinami keli kriterijai, kurių vienas yra atkūrimo taškas (angl. *Recovery Point Objective*) [AM13], kuris apibrėžia, už kokį periodą toleruojamas duomenų praradimas. Projektuojamos serverių spiečiaus sistemos atveju siekiamas atkūrimo taškas RPO yra 0.

3. Neprieštaringumas – darant atsargines kopijas skirtingais laiko intervalais, skirtingiems spiečiaus nariams, atsiranda duomenų prieštaringumo tikimybė, t. y. kiekviena atsarginė kopija gali skirtis viena nuo kitos.

Atsarginės kopijos gali būti naudojamos kaip papildoma sistemos apsaugos priemonė, tačiau nėra tinkamoa, kaip pagrindinis architektūrinis sprendimas. Kaip aprašoma Gregory Levitin [LXD15] straipsnyje, toks sprendinys būtų tinkamas esamai COD architektūrai, kuomet turimas vienas pagrindinis serveris.

4.2. Blokų grandinės – vienetiniai žetonai

Vienetiniai arba nepakeičiamieji žetonai (angl. *Non-Fungible Tokens*) – vis labiau populiarėja skaitmeninių meno kūrinių saugojimui ir prekyba jais [VBT+21]. Skaitmeninė informacija saugoma viešose blokų grandinėse (angl. *Blockchain*).

Nepakeičiamas žetonas (NFT) apibrėžiamas kaip kriptografiškai unikalus, nedalomas, nepakeičiamas ir patikrinamas prieigos raktas, vaizduojantis tam tikrą turtą, nesvarbu ar jis būtų skaitmeninis ar fizinis. Šiuo metu didžioji dauguma nepakeičiamųjų žetonų yra sukurti „Ethereum“ blokų grandinių tinkle. Verta paminėti, kad naujai atsirandančios blokų grandinių platformos, tokios kaip „Flow“, „Tezos“ ir „Algorand“, taip pat yra įtrauktas NFT palaikymas. Blokų grandinių koncepcija gali būti naudojama daug plačiau, nei tik pinigams – kriptovaliutoms. Esminis skirtumas tarp „Ethereum“ ir „Bitcoin“ blokų grandinės tinklo yra „Ethereum“ prieigos raktas (t. y. žetonas saugomas ir juo prekiaujama blokų grandinėse) kuris sukurtas ir valdomas pagal vadinamąją „išmaniąją sutartį“ [WOY+19]. Išmanioji sutartis gali būti apibūdinama kaip savarankiškai vykdoma sutartis tarp dviejų šalių, kurių susitarimo sąlygos yra įrašytos į kodo eilutes ir kurių vykdymą bei susijusias operacijas galima atsekti.

NFT išpopuliarėjo kriptovaliutų bendruomenėje per žaidimus. Ypač per virtualų žaidimą „CryptoKitties“, kuris yra skaitmeninių kolekcionuojamų daiktų žaidimas, kuriame kiekvienas „CryptoKitties“ yra unikalus. Be žaidimų, NFT taip pat pritaikė įvairios pramonės šakos, nuo finansų, įsipareigojimų ir paskolų iki tiekimo grandinių.

4.2.1. Kristalo struktūros saugojimas vienetiniame žetone

Kristalo struktūra saugoma CIF formato failuose [GDM+11] taip pat galėtų būti saugomi kaip vienetiniai žetonai, taip kaip saugomi meno kūriniai ar bet kuris kitas skaitmeninis turtas.

Taip kaip [VBT+21] straipsnyje keliama hipotezė, kad ši technologija, gali padėti muziejams, galerijoms ar bibliotekoms pritraukti papildomų lėšų, būtų galima kelti hipotezę, kad kaip vieneti-

niai žetonai galėtų būti naudojami mokslo tyrimų ar mokslo duomenų saugojimo ar tyrimo atlikimo finansavimui, tačiau šios hipotezės tolimesnis plėtojimas šiame tiriamajame darbe neatliekamas.

Esminis „Ethereum“ blokų grandinės trūkumas – didelis energijos suvartojimas. To pasekoje mokamas trasakcijos mokestis (angl. *Gas fees*) [VBT+21]. Preliminariai, 2020 m. liepos mėn., kai mokesčiai už transakcijas buvo žymiai mažesni nei 2021 m., mokestis už 1 megabaito vaizdo saugojimą (t. y. registravimą) „Ethereum“ blokų grandinė kainavo daugiau nei 13 000 Amerikos dolerių. Sujungti du blokų grandinės pagrindu sukurti žetonai su skaitmeniniu turtu, kurį jie atstovauja yra saugomi išorinėse saugojimo sistemose, NFT platformos naudoja skirtingus susiejimo būdus žetonas su turtu – paprasčiausiomis formomis viena tokia nuoroda gali būti žiniatinklio nuoroda kartu su prieigos raktu jo metaduomenyse. Pritaikant šią technologiją kristalų struktūroms, būtina turėti patikimą išorinę saugyklą su kuria būtų susiejami žetonai, norint išvengti didelių mokesčių.

4.2.2. Hibridinis sutarimas

BFT (angl. *Byzantine fault-tolerant*) protokolas leidžia grupei mazgų pasiekti bendrą sutarimą, net jei kai kai kurios kopijos yra neteisingos. Ši technologija naudojama blokų grandinių technologijoje vadinamam hibridiniam sutarimui (angl. *Hybrid consensus*) [SWK+20]. Šios architektūros idėja yra naudoti ilgiausios grandinės protokolą (angl. *Longest-chain*), kad atsitiktinai pasirinkti komitetą, kuris vykdo BFT protokolą blokų patvirtinimui. Mazgas, kuris išrinktas nariu, privalo išlikti aktyvus iki jo komiteto laikotarpio pabaigos, o tai kenkia prisitaikymo savybėms. Dėl šios savybės atsiranda tikimybės, kad mazgai negalės pasiekti bendro sutarimo dėl to, protokolas praranda patikimumą.

Panašaus principo CIF struktūrų validavimas ir tikrinimas galėtų būti taikomas ir COD, siekiant nustatyti ar visi mazgai veikia tinkamai ir ar nėra nedraugiško spiečiaus mazgo.

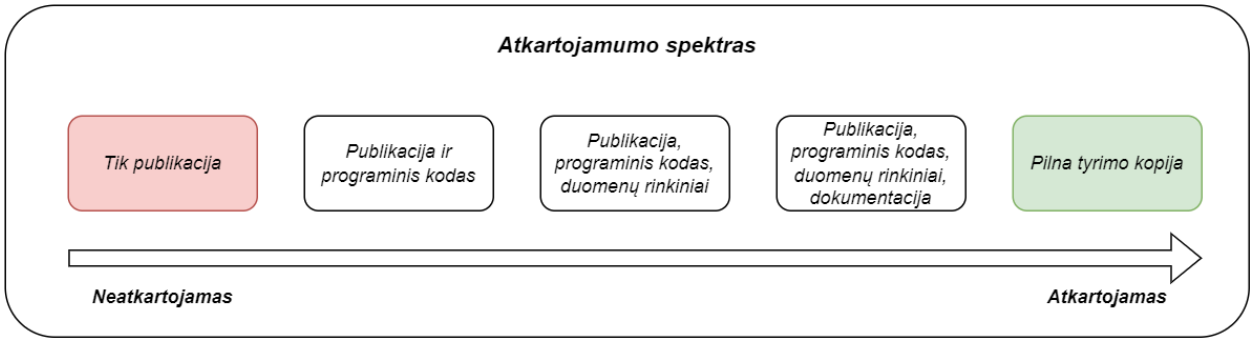
4.3. Tyrimų atkuriamumas

Kaip teigiama daugelyje straipsnių, kad efektyviausias tyrimo rezultatų patikrinimo metodas – tyrimo atlikimas iš naujo. Siekiant atkartoti tyrimo rezultatus, būtina žinoti, kaip tyrimas buvo atliktas. Ši informacija būna pateikiama publikacijoje. Taip pat būtina turėti pradinius duomenis, tyrimui atlikti naudojamas programinis kodas, skriptai, papildomai naudoti duomenų šaltiniai ir kiti duomenys, kurie buvo panaudoti tyrimo rezultatams gauti. Deja, pastaroji informacija ne visuomet yra prieinama arba pateikiama ne pilna apimtimi. Atkreiptinas dėmesys, kad neretai tyrėjai susiduria su problemomis, bandydami atkurti savo pačių tyrimus [SKC00].

Siekiant išspręsti atkuriamumo problematiką, tyrimo rezultatai privalo būti pateikiami pilna apimti [Pen11]:

1. Programinis kodas.
2. Įvesties duomenys.
3. Tarpiniai rezultatai.
4. Galutiniai rezultatai.

Tai gerai atspinti Pav. 3 iliustracija.



3 pav. Atkuriamumo spektras

Analizuojant mokslinius straipsnius susijusius su tyrimų atkuriamumo problematika, tokius kaip [SKN+16], [Bar10] ar [BCH+20] minima programinės įrangos arba programinio kodo, kuris buvo naudojamas tyrimo atlikimui pateikimas. Programinis kodas sulyginimas su bet kuria tyrimo medžiaga ir pabrėžiama jo svarba. Ne mažiau svarbu pateikti visą informaciją, kuri svarbi programinio kodo paleidimui, naudojamos bibliotekos ar kodo paleidimo eiliškumas. Deja, bet ši informacijos dalis būna dažnai praleista ir nedokumentuota [SKC00].

Atkuriamumo problemos (angl. *Reproducibility crisis*) sprendimui, kuriamos pagalbinės priemonės, kurios padėtų efektyviai dokumentuoti tyrimo eigą ir padėtų atkurti tyrimo rezultatus. Taip pat sprendimų ieškoma apeliuojant į tyrėjų sąžiningumą, bei procedūrines priemones, kuriuos leistų publikuoti tik pilna apimtimi pateikiamus mokslinius darbus. Serverių spiečiaus priėmimo ir išleidimo taisyklės gali būti priskiriamos prie procedūrinių priemonių tyrimų atkuriamumui užtikrinti.

4.4. ACID

Kaip teigiama [FPF+14] straipsnyje, kad yra įmanoma optimizuoti CAP teoremos atributus ir taikymo sritis. Pasiekti santykinai didelį prieinamumą ir pakankamai nuoseklius duomenis, nepaisant padalinimų. Pateikiamas elektroninių sveikatos priežiūros įrenginių (angl. *Electronic health records*) pavyzdys. Šie įrenginiai vienu metu gali būti daugybėje ligoninės vietų todėl yra didelė padalinimo ar atsijungimo nuo bendros sitemos tikimybė.

1. Atomiškumas (angl. *Atomicity*) – atomiškumas garantuoja, kad kiekviena operacija būtų traktuojama kaip viena trasakcija, kuri arba visiškai sėkminga, arba visiškai nesėkminga. Jeigu nepavyksta įvykdyti bet kurios iš operaciją sudarančių trasakcijų, visa operacija atšaukiama ir duomenų bazė paliekama nepakeista.

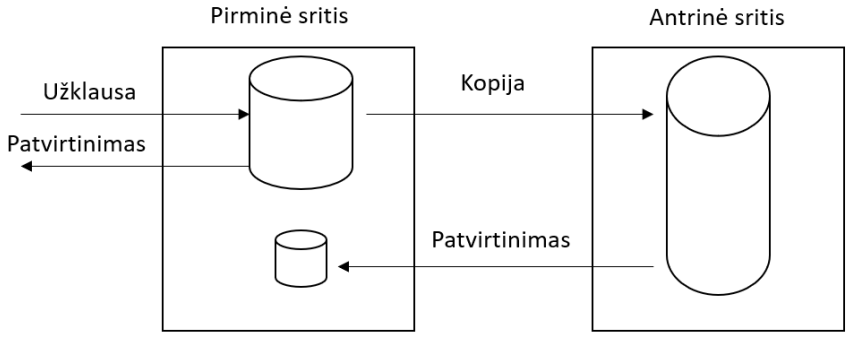
2. Vientisumas (angl. *Consistency*) – duomenų bazė yra nuosekli, jei jos duomenys atitinka visas nuoseklumo taisykles.

3. Izoliavimas (angl. *Isolation*) – izoliavimas užtikrina, kad tuo pačiu metu vykdant operacijas duomenų bazė liktų toje pačioje būsenoje, kuri būtų gauta, jei operacijos būtų vykdomos nuosekliai.

4. Patvarumas(angl. *Durability*) – patvarumas garantuoja, kad atlikus transakcija, ji išliks net ir sistemos gedimo atveju. Paprastai tai reiškia, kad užbaigtos operacijos įrašomos į pastovią, apsaugotą atmintį.

[FPF+14] straipsnyje aprašoma, kaip naudojant skirtingo lygio n-saugių (angl. *n-safe*) replikavimo modelių tipus, automatiniams particijų atkūrimui. Pagrindinis replikacijos dizainas nurodo kiek reikia saugoti duomenų kopijų n-saugus, 2-saugus, 1-saugus (Pav. 4) arba 0-saugus, atitinkamai, kai n, 2, 1 arba 0 iš n kopijų yra nuoseklios ir atnaujintos esant įprastiniam darbo rėžimui. N-saugus ir kvorumui saugus replikacijos dizainas netinka sistemoms, kurios turėtų veikti atjungtu (angl. *Disconnected*) režimu. Šie metodai tinka tik asinchroniniam replikavimui.

Sprendžiam serverių spiečiau iššūkius, toks metodas būtų tinkamas, atsižvelgiant į sistemos apribojimus ir į tai, kad galutinis nuoseklumas (angl. *Eventual consistency*) projektuojamoje sistemoje yra leidžiamas.



4 pav. Principinė “1-safe“ schema

4.5. Peer-to-Peer tinklai

Klientas-klientas sistemos (P2P) (angl. *Peer-to-Peer*) yra paskirstytos sistemos susidedančios iš tarpusavyje sujungtų mazgų, galinčių savarankiškai organizuotis į tinklo topologijas su tikslu dalintis ištekliais, tokiais kaip turinys (dokumentai, vaizdo ir audio medžiaga ir pns.), procesorius, saugykla ir tinklo pralaidumas, galinčios prisitaikyti prie gedimų, laikinų mazgų sutrikimų ar išsijungimų, išlaikant priimtiną patikimumą ir našumą, nereikalaujant centralizuoto valdančio serverio. P2P sistemos skirstomos į kelias kategorijas, pagal tai ar yra centralizuotas valdantis serveris ar ne [AS04]:

1. Visiškai decentralizuota architektūra (angl. *Purely decentralized architectures*) – tokio tipo sistemos sukuria virtualų tinklą, su savitu maršrutizavimo protokolu, kuris leidžia vartotojams dalintis failais. Nėra centrinio koordinuojančio mazgo. Vartotojai prisijungia vienas prie kito tiesiogiai per programinę įrangą, kuri veikia ir kaip klientas ir kaip serveris.

2. Iš dalies centralizuota architektūra (angl. *Partially centralized architectures*) – šis tipas panašus į visiškai decentralizuotą architektūrą, tačiau turi valdantį narį. Valdantis narys išrenkamas automatiškai, jei tenkina numatyti reikalavimai tokie kaip tinklo pralaidumas, skaičiavimo pajėgumai ir pns..

3. Hibridinė decentralizuota architektūra (angl. *Hybrid decentralized architectures*) – šiose sistemose yra centrinis serveris, palengvinantis narių sąveikavimą, metaduomenų, katalogų tvarkymą, tačiau duomenų manai vyksta tiesiogiai (Pav. 5).

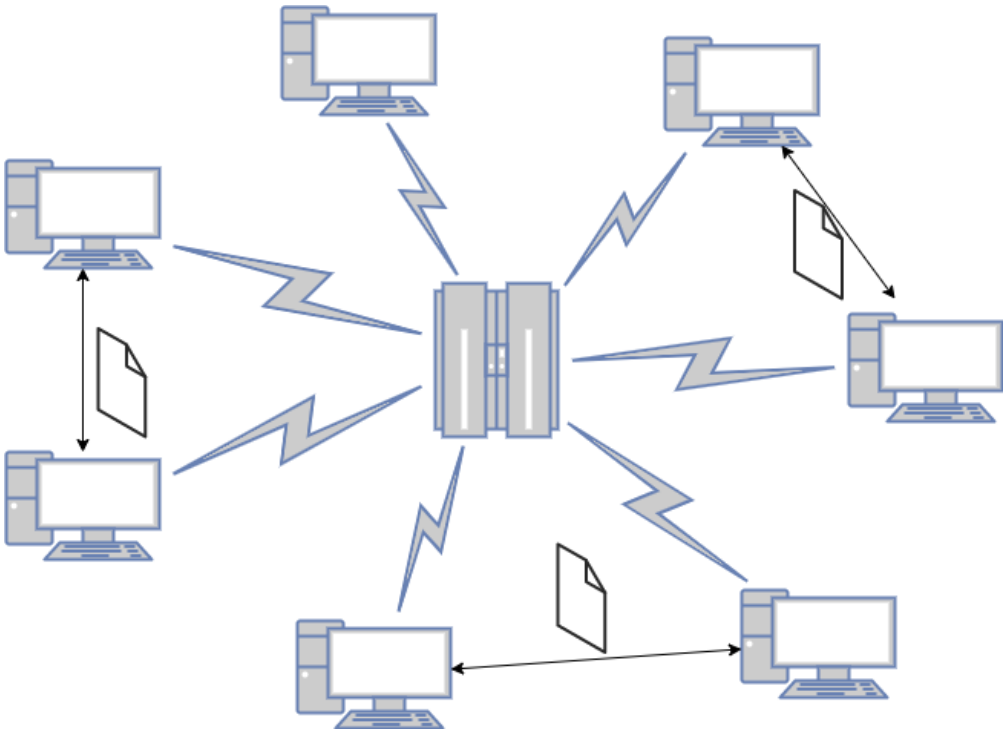
Šio tipo sistemos geba prisitaikyti prie tinklo pokyčių, greitai adaptuotis atsirandant ar pradingstant mazgams, užtikrinant reikiamą našumą [AS04]. Taip pat P2P tinklai leidžia atskiriems kompiuteriams tiesiogiai bendrauti tarpusavyje ir dalintis informacija bei ištekliais.

Numatoma serverių spiečiau architektūra turi panašumų į klientas-klientas (angl. *Peer-to-Peer*) architektūrą. Tikrojoje P2P architektūroje nėra centrinio valdančio serverio. Komunikacija vyksta tiesiogiai tarp dviejų klientų. Hibridinėje P2P architektūroje yra centrinis serveris, kuris nesaugo jokios informacijos. Centrinis narys tik nurodo klientams, kur saugomi duomenys.

Mūsų siekis kuriant patikimą ir išskirstytą COD duomenų bazę yra panašus, kaip P2P tinkluose. Duomenų manai vyksta CIF failais tarp spiečiaus narių. Nėra centrinio serverio, kuris valdytų duomenų mainus. Komunikacija tarp spiečiau narių privalo būti saugi. Tokios P2P sistemos, kaip „Gnutella“ [Rip01] daugiausia naudojamos apsikeitimui failais, kas iš dalies tenkintų keliamus reikalavimus naujai sistemai.

Atlikto „Gnutella“ tyrimo metu, nustatyti šios topologijos pažeidžiamumai, todėl norint šią technologiją panaudoti COD duomenų bazei, būtina įdiegti papildomus saugos mechanizmus.

Esminis skirtumas tarp siekiamos architektūros ir P2P, kad vartotojai, kurie norės pasiekti informaciją, nebus serverių spiečiaus nariai. P2P technologija galėtų būti naudojama, kaip komunikacijos protokolas, tačiau ją reikėtų adaptuoti prie “serverių spiečiaus“ poreikių.



5 pav. Hibridinė decentralizuota architektūra

4.5.1. P2P taikymo sritys

P2P aplikacijų kategorijai priklauso turinio platinimo sistemos kurios yra pagrįstos šia technologija. Išskiriamos dvi kategorijos:

1. Failų apsikeitimo sistemos – šios sistemos skirtos paprastiems, vienkartiniams failų mainams. Paprastai tai yra paprastos programos, kurios užtikrina numatytas funkcijas, neatsižvelgiant į saugumą, prieinamumą ir patikimumą.

2. Turinio publikavimo ir saugojimo sistemos – šios sistemos yra skirtos sukurti paskirstytą saugyklą, kurioje ir per kurią vartotojai galėtų saugiai ir nuolat skelbti, saugoti ir platinti turinį. Toks turinys skirtas tam, kad jį kontroliuojamai galėtų pasiekti atitinkamas teises turintys nariai. Pagrindinis tokių sistemų akcentas yra saugumas ir patvarumas, taip pat turinio valdymo – atnaujinimo, pašalinimo, versijų kontrolės priemonės.

Sprendžiant pagal turinio publikavimo ir saugojimo sistemų apibrėžimo, šio tipo sistema galimai tenkintų naujos COD sistemos reikalavimus.

4.6. NoSQL duomenų bazės

Reliacinė duomenų bazių sistema duomenims saugoti naudoja dvimatę lentelę su tokiomis savybėmis kaip operacijos, sudėtingos SQL užklausos ir su keliomis lentelėmis susijusios užklausos. Tačiau tai nėra veiksminga, kuomet dirbama su dideliais duomenų kiekiais. Reliacinių duomenų bazių mastelio keitimas reikalauja galingų serverių, kurie yra brangūs ir sunkiai valdomi. NoSQL duomenų bazės suteikia galimybę saugoti visus duomenis kaip dokumentus, o ne įprastą lentelės eilutės-stulpelio metodą. NoSQL yra labai naudinga, kai mums reikia pasiekti ir analizuoti didžiulius kiekius nestruktūrizuotų duomenų arba duomenis, kurie nuotoliniu būdu saugomi keliuose virtualiuose serveriuose [TPP13].

Išskiriami keli NoSQL duomenų bazių tipai:

1. Raktų-reikšmių saugyklos: raktų-reikšmių sistemos saugo indeksuotas reikšmes, kurias galima pasiekti pagal indeksus. Naudojant šio tipo sistemas galima saugoti struktūrizuotus arba nestruktūruotus duomenis. Labai svarbu, kad tokio tipo sistemą nesudėtinga realizuoti kaip išskirstytą.

2. Į stulpelius orientuotos duomenų bazės: Šio tipo duomenų bazės duomenis saugo stulpelyje.

3. Dokumentinės saugyklos: Šiose tipo duomenų bazėse duomenys ir tvarkomi kaip dokumentų rinkiniai. Naudojant šią duomenų bazę, galima pridėti bet kokio ilgio ir bet kokio laukų kiekio duomenis.

4. Grafų duomenų bazės: Šios duomenų bazės duomenų saugojimui ir apdorojimui grafų pavidalu naudoja mazgus.

Palyginti skirtingų tipų NoSQL duomenų bazes galima šiais aspektais – duomenų modelis, duomenų replikavimo galimybės, duomenų vientisumas. Duomenų replikavimo ir vientisumo užtikrinimas yra labai svarbus projektuojant ir parenkant COD sprendimus.

CAP teoremos apribojimai [Bre12] taip galioja ir NoSQL duomenų bazėms [HJ11]. Sistemos mazgų išskirstymas užtikrina didesnį prieinamumą ir užtikrina didesnę greitaveiką, tačiau svarbu atkreipti dėmesį, kad tokie sprendimai galimi tik tada kai sistemoje galimoje galimas galutinis nuoseklumas (angl. *Eventually Consistent*). Tokios duomenų bazės remiasi BASE principais [Gan15]:

1. Iš esmės pasiekiama (angl. *Basically Available*) – duomenys prieinami didžiąją laiko dalį, negarantuojant nuoseklumo.

2. Minkšta būsena (angl. *Soft State*) – dėl nuoseklumo stokos duomenų reikšmės laikui bėgant gali keistis.

3. Galutinis nuoseklumas (angl. *Eventually Consistent*) –- sistema stengiasi pasiekti nuoseklumą, nors šiuo metu jo ir nėra. Užklausa gražina rezultatą, tačiau tai nebūtinai bus naujausia duomenų versija.

5. Išskirstytų sistemų algoritmai

Paskirstytas susitarimas yra pagrindinė išskirstytos sistemos problema. Jis reikalauja, kad visi procesai ar mazgai susitartų dėl bendros reikšmės. Atviros kristalografinės duomenų bazės atveju bendra reikšmė reikštų susitarimą dėl CIF struktūros atitikimo reikalavimams. Bendras sutarimas yra būtinas, kad paskirstytos sistemos būtų atsparios gedimams, nes teisingai veikiantys mazgai (angl. *Non-faulty replicas*) privalo sutarti dėl reikšmių. Deja, sutarimas yra sudėtingas kai tiek mazgai, tiek ryšio kanalas yra nepatikimi [CLS16]. Kad būtų pasiektas susitarimas dėl tam tikros reikšmės, kiekvienas procesas gali pateikti savo pasiūlymą. Galiausiai, naudojant paskirstytą konsensuso algoritmą, visi tinkamai veikiantys procesai nusprendžia tą pačią reikšmę. Kuriant išskirstytas sistemas, dažnai daromos klaidingos prielaidos į kurias būtina atsižvelgti:

1. Tinklas yra patikimas.
2. Nėra tinklo vėlinimo.
3. Pralaidumas yra begalinis.
4. Tinklas yra saugus.
5. Topologija nesikeičia.
6. Egzistuoja vienas administratorius.
7. Perdavimas nieko nekainuoja.
8. Tinklas yra homogeninis.

COD labiausiai įtakoja tinklo vėlinimas ir pralaidumas. Dažniausiai CIF failai yra palyginus maži, tačiau pasitaiko ir ženkliai didesnių failų, kurie esant geografiškai išskirstytai sistemai, gali sukelti papildomų iššūkių.

5.1. Paxos

Paxos algoritmas leidžia pasiekti sutarimą išskirstytoje asinchroninėje sistemoje, kur pranešimai gali būti dubliuojami, prarasti arba pertvarkyti [Lam01]. Yra daug šio algoritmo variacijų, kaip *Multi-paxos*, *Pig-paxos*, *Cheap-paxos* ir kt.

Paxos algoritmą sudaro trys rolės:

- Siūlytojai (angl. *Proposers*) – apdoroja vartotojo įvestį.
- Akceptoriai (angl. *Acceptors*) – balsuoja už pasiūlytas reikšmes.
- Besimokantieji (angl. *Learners*) – išsaugo pasiūlytas reikšmes.

Multi-Paxos algoritmas neriboja, kad pasiūlymus gali siūlyti tik lyderis (silpnas lyderis). Reikšmė priimama, jei už ją balsuoja kvorumas akceptorių.

5.2. Raft

Kaip alternatyva Paxos algoritmui, buvo sukurtas Raft [LAF21], kuris turėtų būti suprantamesnis lyginant su Paxos ir užtikrinama panaši greitaveika kaip Multi-paxos [Lam19].

Algoritmas komunikacijai naudoja RPC protokolą ir naudoja du pranešimų tipus:

- *AppendEntries* – pranešimas siunčiamas lyderio logų replikavimui ir patikrinti narių gyvybingumą (angl. *Heartbeat*).
- *RequestVote* – siunčiamas kandidato pradėti balsavimą.

Gyvybingumo patikrinimas naudojamas lyderio, kad jį atpažintų laikotarpiais, kai nėra siunčiamos žinutės. Gyvybingumo patikrinimas yra *AppendEntries* užklausa be įrašų. Ne visi procesai gali būti išrinkti lyderiais. Lyderis turi turėti išsaugotas visas reikšmes, kad būtų išrinktas lyderiu, priešingu atveju tai gali priversti kitus procesus perrašyti anksčiau išsaugotas reikšmes. Šiame algoritme bet kuriuo metu yra daugiausia vienas lyderis ir reikšmes gali siųsti tik lyderis (stiprus lyderis).

Lyginant Multi-paxos ir Raft šie algoritmai yra labai panašūs:

- Raft lyderis yra Multi-Paxos siūlytojas.
- Raft terminas iš esmės yra Multi-Paxos pasiūlymo ID.
- Raft žurnalo įrašas yra Multi-Paxos pasiūlymas.
- Raft žurnalo indeksas yra Multi-Paxos egzemplioriaus ID.
- Raft lyderio rinkimai iš esmės yra Multi-Paxos pasirengimo etapas.
- Raft žurnalo kopijavimas yra Multi-Paxos priėmimo fazė.

Nors šie algoritmai turi labai daug panašumų, tačiau vienas iš labai svarbių skirtumų, kad Raft algoritme naujas narys gali prisijungti kaip nebalsuojantis narys, tam kad sinchronizuoti žurnalo informaciją (angl. *Log replication*), COD atveju – sinchronizuoti duomenis. Taip pat stipraus lyderio savybė, leidžia lengviau suvaldyti sistemos elgseną ir keičiantis lyderiui, keisti sistemos konfigūraciją. Dėl šių savybių, COD sistemos atveju, Raft turi pranašumą prieš Paxos algoritmą.

6. Baltymų duomenų bankas

Baltymų duomenų bankas (angl. *Protein Data Bank*) – pasaulinė baltymų duomenų banko organizacija (angl. *Worldwide Protein Data Bank*) veikia pagal oficialią sutartį, kuria visi partneriai įsipareigoja standartizuoti, rinkti, patvirtinti, komentuoti ir saugoti makromolekulinės struktūros duomenis kaip vieną pasaulinį duomenų archyvą ir platina duomenis vartotojams nemokamai ir be jokių duomenų naudojimo apribojimų. Šią organizaciją sudaro RCSB baltymų duomenų bankas, Japonijos baltymų duomenų bankas, baltymų duomenų bankas Europoje ir BioMagResBank. Baltymų duomenų bankas orientuojasi į šias paslaugas:

- Įkėlimas, patikrinimas ir apdorojimas – kiekviena struktūra yra patvirtinama naudojant bendruomenės nustatytus standartus ir kokybės rodiklius.
- Duomenų archyvavimas ir prieigos užtikrinimas – Už duomenų saugojimą ir tvarkymą atsakingi visi duomenų banko partneriai. Duomenys saugomi ir archyvuojami vadovaujantis FAIR principais. Duomenų saugumui užtikrinti daromos nuolatinės atsarginės kopijos. Duomenys gali būti prieinami per FTP arba taikomųjų programų sąsają (API).
- Duomenų tyrinėjimas – laisvai prieinami duomenų paieškos, naršymo, vizualizavimo, ataskaitų generavimo ir analizės įrankiai. Visos funkcijos realizuotos interneto naršyklės pagrindu.
- Informavimas ir švietimas – plėtoja informavimo ir švietimo išteklius, orientuotus į struktūrinę biologiją ir jos poveikį visuose moksluose. Funkcijos reguliariai atnaujinamos ir yra laisvai prieinamos pedagogams ir jų mokiniams.

Po patikrinimo proceso, kiekvienai struktūrai suteikiamas unikalus 4 simbolių identifikacinis numeris sudarytas iš raidžių ir skaičių. Prieš kelis metus duomenų bankas buvo visiškai pertvarkytas, leidžiantis patobulinti paiešką ir lengvesnę prieigą prie PDB duomenų, integruotų su daugiau kaip 40 išorinių biologinių duomenų šaltinių. Visi duomenų banko nariai naudoja OneDep sistemą duomenų struktūrų apdorojimui. OneDep veikia kaip decentralizuota sistema, tačiau tuo atveju, jei OneDep prieigos svetainės, esančios už JAV ribų, taptų nepasiekiamos, įrašai būtų nukreipiami, tam kad būtų užtikrinta nenutrūkstama prieiga visame pasaulyje [BBB+20].

Baltymų duomenų banko duomenis plačiai naudoja daugybė tyrėjų, mokslininkų ir studentų visame pasaulyje. Per 2019 m. tiesiogiai iš archyvo buvo atsisiųsta $\approx$ 840 mln. duomenų struktūrų.

7. Literatūros apžvalgos apibendrinimas

1. Esama COD realizacija neužtikrina duomenų ilgaamžiškumo, nes išnykus pagrindiniam serveriui, pradingtų svarbūs kristalografiniai duomenys.

2. Atvira kristalografijos duomenų bazė pasižymi tokiomis ypatybėmis:

   (a) Pagrindiniai duomenys saugomi CIF formato failuose, kurie yra pagrindinis ir pradinis duomenų šaltinis.

   (b) COD duomenų bazėje atliekama sąlyginai mažai rašymo operacijų, tačiau daug skaitymo operacijų. Esant šioms aplinkybėms galimas yra galimas galutinis vientisumas (angl. *Eventually Consistent*).

   (c) COD sistemoje rašymo operacija atliekama vieną kartą. Sekančios operacijos yra atnaujinimo.

   (d) Sistemoje saugoma kiekviena revizija.

   (e) Nėra atliekamos duomenų trynimo operacijos.

   (f) COD duomenų bazė visiškai atitinka FAIR principams.

   (g) Senesnė duomenų revizija nereiškia duomenų nekorektiškumo.

3. Projektuojamas serverių spiečius yra išskirstyta sistema, todėl ji yra apribojama CAP teoremos.

4. Numatomos priėmimo ir išleidimo taisyklės, kurių privaloma laikytis visiems spiečiau nariams ir norinčiais jais tapti.

5. Atsarginio kopijavimo sprendimas užtikrina duomenų išsaugojimą ir greitą sistemos atkūrimą esant net visiškam sistemos praradimui, tačiau naudojant šį sprendimą būtina atsižvelgti į sistemos charakteristikas ir galimus duomenų praradimo laikus (RPO). Atsižvelgiant į COD sistemai keliamus reikalavimus, šis sprendimas gali būti naudojama, kad pagalbinė priemonė, bet ne kaip pagrindinis architektūrinis sprendimas.

6. Kristalų struktūros CIF failuose gali būti saugomos blokų grandinės technologijos pagrindu, kaip vienetiniai žetonai. Daroma prielaida, kad NFT technologija galėtų būti pritaikoma serverių spiečiaus ar kitų mokslo duomenų bazių finansavimui, tačiau ši hipotezė šiame tiriamajame darbe toliau plėtojama nebus.

7. Sprendžiant mokslinių tyrimų atkuriamumo iššūkius, esminis kriterijus – tyrimų pateikimas pilna apimtimi. Atkreiptinas dėmesys, kad duomenis laikomi ne tik tyrimo duomenys, bet ir programinis kodas, tarpinių skaičiavimų failai ir kita informacija naudota tyrimo rezultatms gauti.

8. Turinio publikavimo ir saugojimo sistemos iš dalies tenkina serverių spiečiaus reikalavimus.

9. P2P technologija galimai tinkama serverių spiečiaus realizacijai, papildant technologinius sprendinius atitinkamomis semantinio patikrinimo procedūromis, bei užtikrinant saugią komunikaciją tarp spiečiaus narių, panaudojant kriptografines priemones.

10. NoSQL duomenų bazių BASE principai atitinka serverių spiečiaus invariantus ir gali būti naudojami, kaip technologinis sprendinys.

11. Išskirstytų sistemų algoritmas gali būti panaudojamas projektuojant naują COD sistemą.

Siekiant teisingai įvertinti naujos architektūros teisingumą ir atitikimą reikalavimams, bus sudaromas modelis, kuris remiantis atitinkamais kriterijais (parametrais), kaip spiečiaus narių skaičius, mazgo gedimo tikimybė ir pan., bus vertinama duomenų praradimo tikimybė atsižvelgiant į laiką.

8. Hipotezės patikrinimas

Keliama hipotezė, kad serverių spiečius sudarytas iš N mažiau patikimų mazgų, leis išsaugoti duomenis su mažesne praradimo tikimybe, nei turint vieną labai patikimą mazgą. Šiuo metu yra žinoma, kad per 15 metų dėl organizacinių priežasčių išnyko pusė (du iš keturių) COD serverių. Remiantis šiais faktais, daroma prielaida, kad COD narių nykimo greitis arba pusėjimo trukmė yra lygi 15 m. Kiekvienais metais tikimybė, kad serveris išnyks yra vienoda. Galimi fizinės įrangos gedimai nėra vertinami.

Pusėjimo trukmė šiuo atveju apibrėžia laiko tarpą, per kurį išnyksta pusė mazgų. Pagal turimus duomenis ir planuojamą sistemos gyvavimo trukmę ($\approx$ 100 m.) sistema turėtų išgyventi 7 pusėjimo trukmes. Pateikiamas programinis modelio kodas su komentarais, kuriame modeliuojamas eksponentinis nykimas angl. *Exponential decay*). Modeliuojami keli atvejai be naujų narių atsiradimo ir įvedant tikimybę, kad gali atsirasti nauji nariai. Taip pat simuliuoja stacionarinė sistemos būsena. Visais atvejais modeliuojami procesai yra stochastiniai. Naujų narių atsiradimas ir nykimas priklauso nuo atsitiktinių įvykių (modelyje generuojami atsitiktiniai skaičiai.).

```
import math
import random
import csv
import numpy as np
import matplotlib.pyplot as plt

# Kintamieji saugoti reikšmėms, iš kurių generuojami grafikai
plot = []
plot2 = []
# Kintamasis saugoti sistemos gyvavimo trukmę, kai N pasiekia 0
cluster_age = []
# Kintamasis saugoti sistemos gyvavimo trukmę, kai N nepasiekia 0
cluster_survive = []
# Mazgų skaičius
cluster_count = []
# Pradinis mazgų skaičius
M = 50
# Naujų mazgų atsiradimas per pusėjimo trukmę
R = 10
# Pusėjimo trukmė. Laiko vienetais
t = 1000
# Modeliavimo kartojimo skaičius. (eksperimentų skaičius)
experiments = 20
# Modeliavimo trukmė. Laiko vienetais
interval = 50000
# p - tikimybė, kad spiečiaus narys išnyks
```

```
p = 1 - 1 / pow(2, 1 / t)
# p1 - tikimybė, kad spiečiaus narys išnyks.
# Kitokia formulė papildomam patikrinimui.
p1 = 1 - math.exp(-(1 / t) * math.log(2))
# z - tikimybė, kad naujas narys atsiras.
z = (R/t) * math.log(2)
print(p, p1, z)

# Funkcija duomenų išsaugojimui faile
def write_to_file(data, mode="a"):
    with open('data/model4.dat', mode, newline='') as f:
        # create the csv writer
        writer = csv.writer(f)
        # write a row to the csv file
        writer.writerow(data)

# Ciklas eksperimentų kartojimui
for i in range(experiments):
    # Pradinis narių čskaiius N. Prilyginama reikšmė M
    N = M
    for n in range(1, interval):
        # Tikrininamas kiekvienas narys ar neišnyko
        for j in range(N):
            # Generuojamas atsitiktinis skaičius. Jei atsitiktinis
            # skaičius mažesnis už išnykimo tikimybę,
            # mažinamas mazgų skaičius. Pasiekus 0 ciklas stabdomas.
            r = random.uniform(0, 1)
            if r <= p:
                N = N - 1
                if N == 0:
                    break
        r = random.uniform(0, 1)
        # Generuojamas atsitikstinis skaičius r. Jei z - tikimybė, kad
        # atsiras naujas mazgas didesnė, už atsitiktinį skaičių
        # N šreikmė didinama vienetu.
        if r < z:
            N = N + 1
        plot.append(n)
        plot2.append(N)
        if N == 0:
```

```
                cluster_age.append(n)
                break
        if N > 0:
            cluster_survive.append(1)
            cluster_count.append(N)
        plot.append(None)
        plot2.append(None)
# Generuojama diagrama.
plt.plot(plot, plot2, linewidth='0.7')
plt.grid()
plt.ylabel('Mazgų skaičius')
plt.xlabel('Trukmė, laiko vienetais')
plt.title('Stacionarinė būsena')
plt.savefig("img\model4")
plt.show()

print("Vidutinis spiečiaus amžius: ", np.median(cluster_age),
"Santykis: ", np.median(cluster_age)/t,
" Bandymų skaičius: ", experiments)
print(cluster_age)
print("Tikimybė, kad serverių spiečius išnyks per ",
interval, ": ", len(cluster_age)/experiments)
print("Tikimybė, kad serverių spiečius išgyvens ",
interval, ": ", len(cluster_survive)/experiments)
print("Vidutinis mazgų skaičius: ", np.mean(cluster_count))
```

8.1. Modeliavimo įrankiai ir bibliotekos

Šio modelio sudarymui buvo naudojami šie įrankiai ir bibliotekos:

1. Python – populiari programavimo kalba, darbui su duomenis.
2. IntelliJ IDEA – kodo redaktorius.
3. Numpy – duomenų analitikos biblioteka, turinti reikalingas funkcijas modelio sudarymui.
4. Pandas – duomenų analizavimo biblioteka.
5. Matplotlib – grafikų braižymo biblioteka.

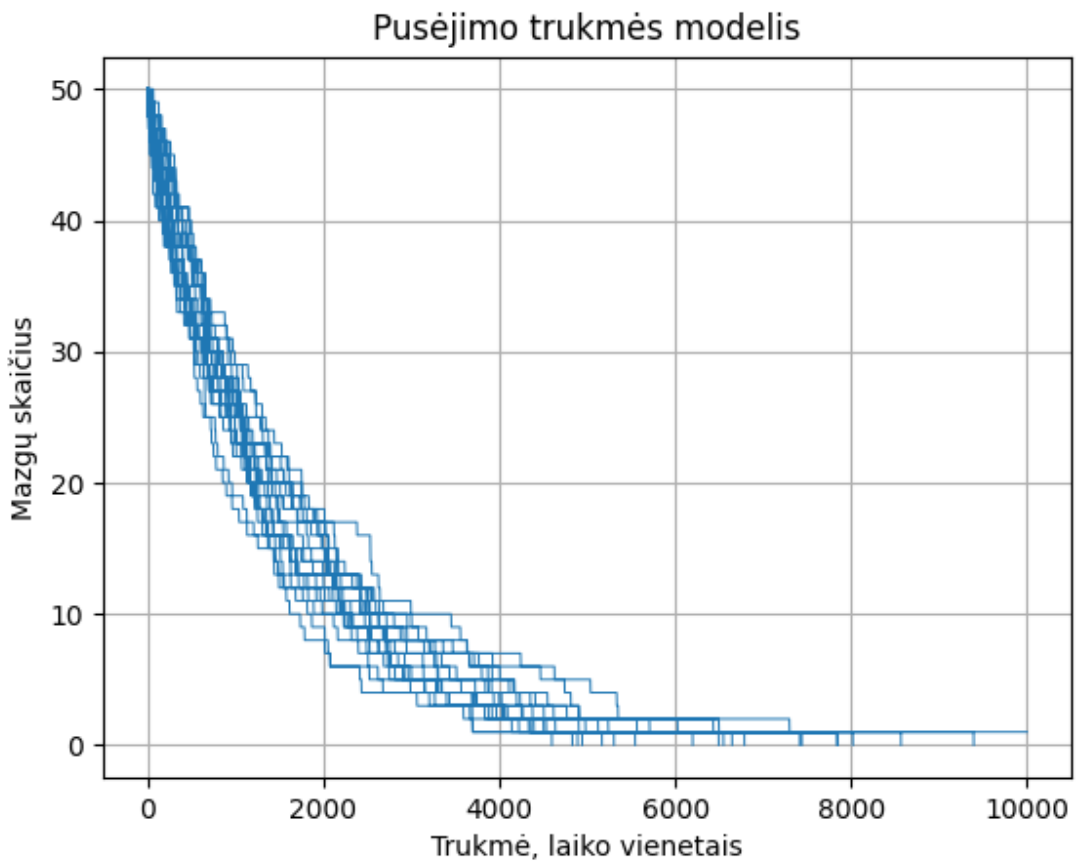
8.2. Pusėjimo trukmė be papildomų narių atsiradimo

Atliekamas eksponentinio nykimo modeliavimas laikant prielaidą, kad naujų narių atsiradimo tikimybė lygi 0. Kaip ir numatyta esant dideliam pradiniam mazgų skaičiui, nykimo greitis pradžio-

je yra didžiausias. Parenkamas sąlyginai didelis pradinis mazgų skaičius, bei ilgesnė nei numatyta modeliavimo trukmė. Atliekant šį modeliavimą sistema išnyko greičiau nei siekiama trukmė.

Nustatomi tokie modelio parametrai:

1. Pradinis mazgų skaičius – 50.
2. Pusėjimo trukmė – 1000 laiko vienetų.
3. Bandymo intervalas – 10000 laiko vienetų.
4. Naujo mazgo atsiradimo tikimybė – 0.
5. Bandymų skaičius viename eksperimente – 20.
6. Siekiamas rezultatas – vidutinė gyvavimo trukmė nemažiau nei 7000 laiko vienetų.



6 pav. Modeliavimas be naujų mazgų atsiradimo

Gautas rezultatas:

1. Vidutinis amžius – 5452 laiko vienetų.
2. Tikimybė, kad numirs per 10000 laiko vienetų – 0.9
3. Trumpiausia gyvavimo trukmė – 3637 laiko vienetų.

4. Ilgiausia gyvavimo trukmė – >10000 laiko vienetų.

5. Numatyta konfigūracija nepasiekė minimalios gyvavimo trukmės.

8.3. Pusėjimo trukmės modeliavimas su naujo nario atsiradimu

Daroma prielaida, kad bet kuriuo laiko momentu gali atsirasti papildomas spiečiaus narys. Modelio parametro R reikšmė nustatoma į 1. Grafikas pasikeičia ir pastebimos situacijos, kai priaugimo greitis didesnis nei nykimo greitis. Vidutinis sistemos amžius prailgėjo beveik 800 laiko vienetų. Nustatomi tokie modelio parametrai:

1. Pradinis mazgų skaičius – 50.

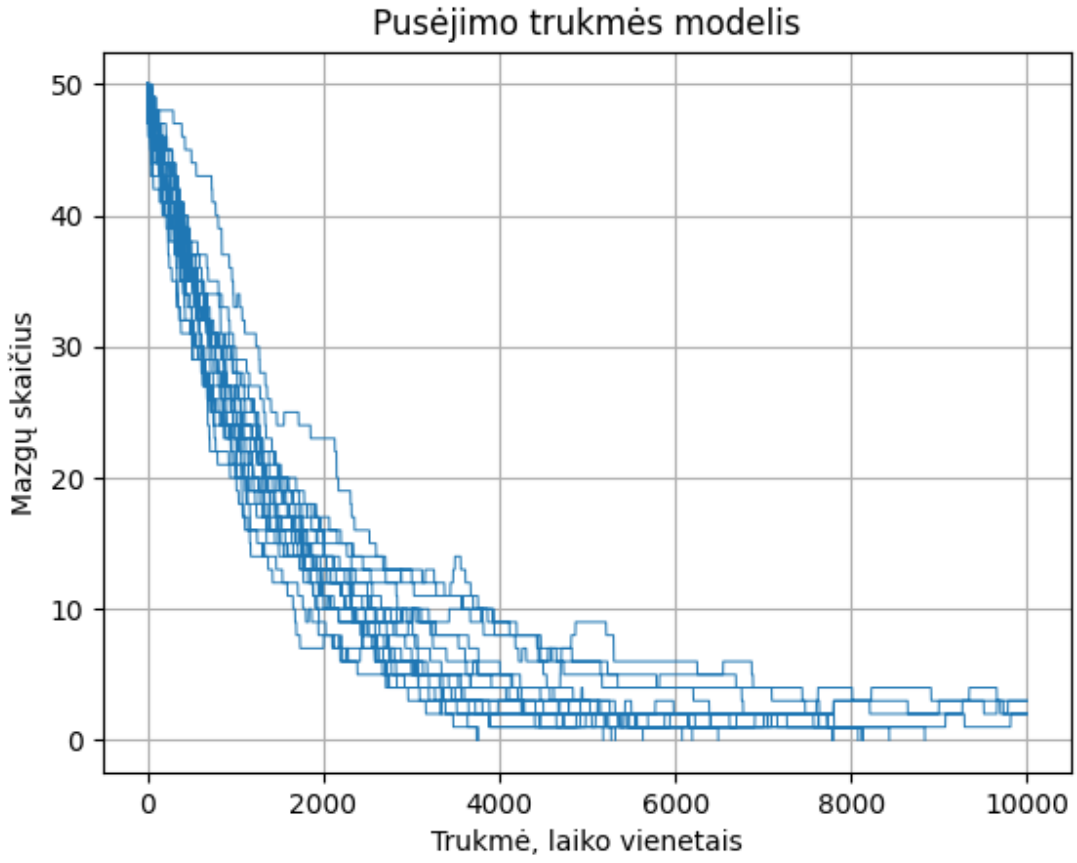
2. Pusėjimo trukmė – 1000 laiko vienetų.

3. Bandymo intervalas – 10000 laiko vienetų.

4. Naujo mazgo atsiradimo tikimybė – 0.0007.

5. Bandymų skaičius viename eksperimente – 20.

6. Siekiamas rezultatas – vidutinė gyvavimo trukmė nemažiau nei 7000 laiko vienetų.

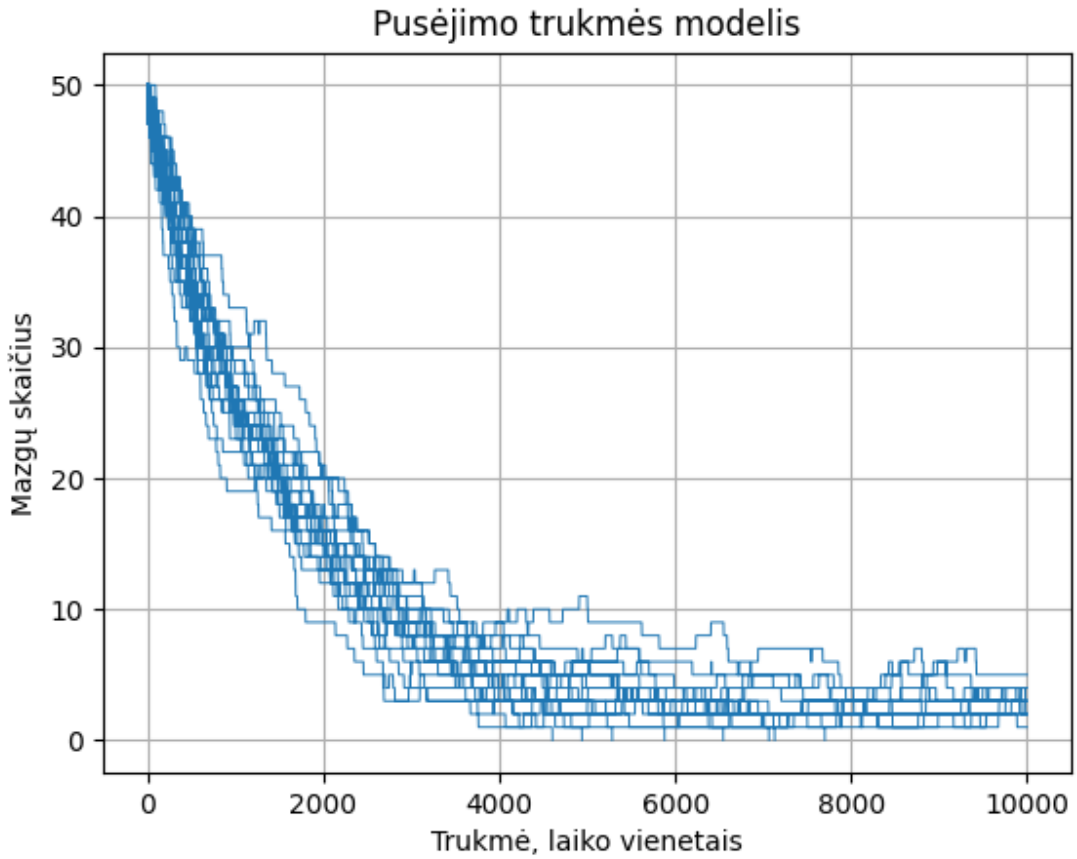


7 pav. Pusėjimo trukmės modeliavimas su naujo mazgo atsiradimu.

Gautas rezultatas:

1. Vidutinis amžius – 6184 laiko vienetų.
2. Tikimybė, kad numirs per 10000 laiko vienetų – 0.75
3. Trumpiausia gyvavimo trukmė – 4282 laiko vienetų.
4. Ilgiausia gyvavimo trukmė – >10000 laiko vienetų.
5. Numatyta konfigūracija artima minimaliai gyvavimo trukmei.

Naujo mazgo atsiradimo parametras R nustatomas į 2. Tai lygu naujo mazgo atsiradimo tikimybei – 0.001. Matomas ženklus grafiko pasikeitimas su aiškiais ruožais, kai mazgų skaičiaus augimas didesnis nei nykimas.



8 pav. Pusėjimo trukmės modeliavimas su naujo mazgo atsiradimu.

Gautas rezultatas:

1. Vidutinis amžius – 7209 laiko vienetų.
2. Tikimybė, kad numirs per 10000 laiko vienetų – 0.35
3. Trumpiausia gyvavimo trukmė – 4746 laiko vienetų.

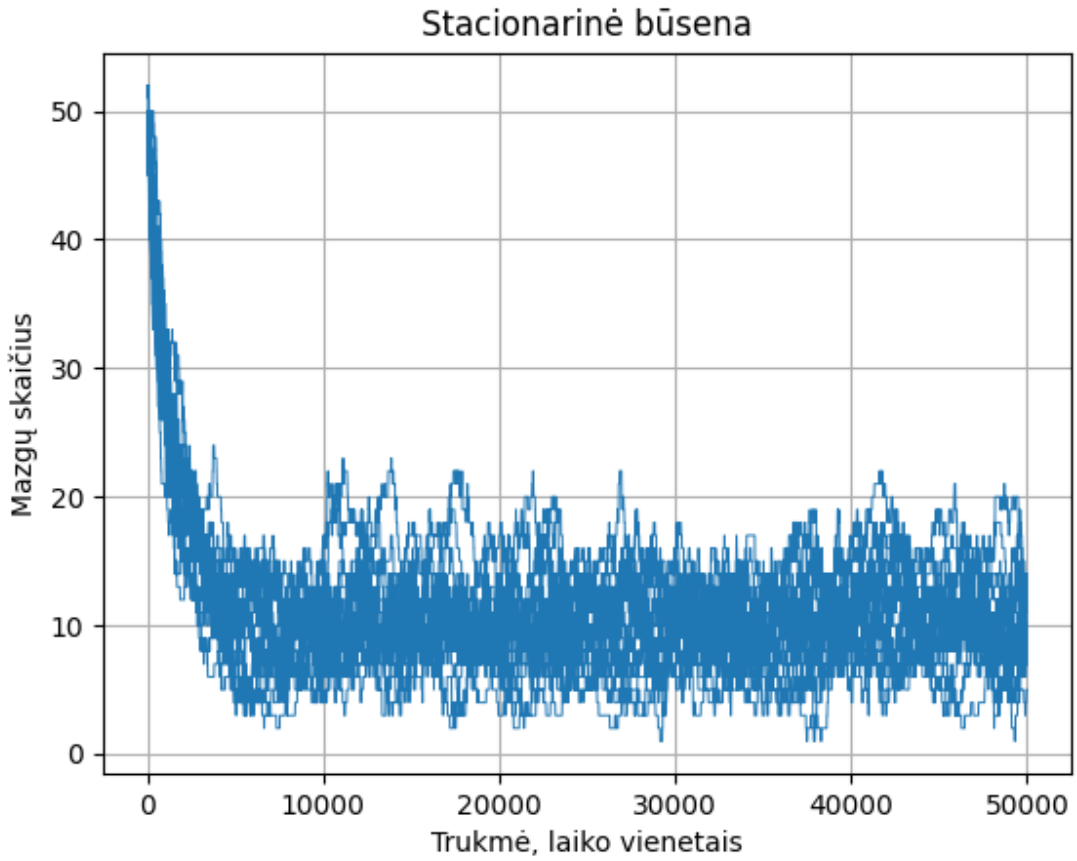
4. Ilgiausia gyvavimo trukmė – >10000 laiko vienetų.

5. Numatyta konfigūracija atitinka minimalią gyvavimo trukmę.

Atliekant šį modeliavimą, su parinktais parametrais, pasiekta minimali gyvavimo trukmė, todėl daugiau panašių simuliacijų neatliekama.

8.4. Stacionarinė sistemos būsena

Siekiamybė, kad sistema dirbtų stacionarinėje būsenoje – t. y. per laiko momentą, kaip pusėjimo trukmė, atsirastų ir pradingtų vienodas mazgų skaičius. Atliekant šią simuliaciją parenkamas ženkliai ilgesnis laiko periodas nei siekiama minimali vidutinė gyvavimo trukmė. Sistemos gyvavimo ciklui esant stacionarinėje būsena, turime nuolatinį atsinaujinimo procesą, t. y. per laiko vienetą tiek ir atsiranda ir pranyksta mazgai, balansuojant aplink numatytą naujų mazgų atsiradimo skaičių – šioje simuliacijoje 10. Atitinkamai naudojant mažesnį skaičių, atsiranda tikimybė išeiti iš stacionarinės būsenos.



9 pav. Stacionarinės sistemos būsenos modelis.

9. Architektūrinis sprendimo aprašas

Mokslo tiriamajame darbe toliau pateikiamos galimos serverių spiečiaus architektūros. Esminis sistemos patikimumo kriterijus - veikimas privalo būti užtikrinamas be žmogaus įsikišimo, esant vieno ar kelių mazgų gedimui, priklausomai nuo pradinio mazgų skaičiaus. Nei vienoje architektūroje nėra griežtai ribojami technologiniai ir programiniai sprendiniai kokybės ir sintaksės patikrinimui. Kiekvienas spiečiaus narys naudodamas jam pateiktą sistemos karkasą (angl. *Framework*) gali naudoti savo įgyvendintus CIF sintaksės ir kokybės tikrinimo sprendimus. Sprendimo architektūriniai sprendiniai sudaromi remiantis COD ir CIF failų tikrinimo eiga, tačiau technologiniai sprendimai ir bendras sistemos karkasas gali būti pritaikomas ir kitų mokslinių duomenų saugojimui ir dalinimuisi, kuriose galimi panašūs apribojimai kaip ir atviroje kristalografijos duomenų bazėje. Sprendimai, kurie apribotų naudoti kitus įrankius, kaip TPM (angl. *Trusted Platform Module*) nėra priimtini. Numatoma, kad turės būti galimybė palaikyti skirtingus failų formatus vienu metu, kaip pvz.: CIF v1.0, CIF v2.0. Skirtingų formatų palaikymas galimas neribotą laiką. Nepriklausomai nuo pasirinkto architektūrinio modelio, visos komunikacijos tarp spiečiaus mazgų privalo būti vykdomos saugiais protokolais. Tai būtina sąlyga siekiant užtikrinti sistemos saugumą.

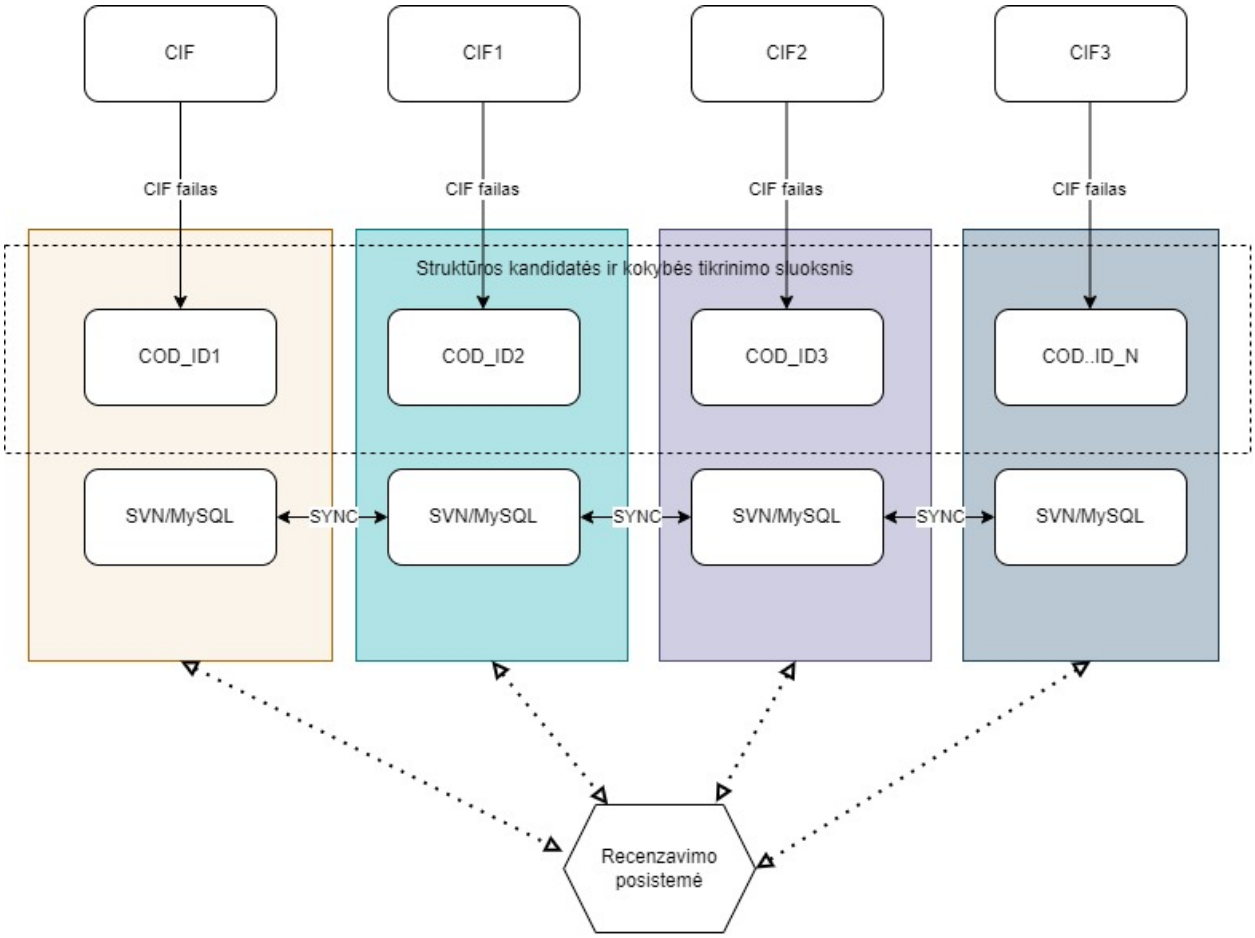
Kiekviename spiečiaus naryje numatomas papildomas privatus ruožas, kuris rezervuojamas privačiam struktūrų saugojimui ir nėra sinchronizuojamas tarp spiečiaus narių. Tokiu sprendimu, COD bus galima naudoti kaip asmeninį archyvą ar tarpiniam tyrimų rezultatų saugojimui.

9.1. Išskirstytos sistemos modelis nenaudojant automatizuoto konsensuso algoritmo

Daroma prielaida, kad tikimybė, jog ta pati struktūra bus įkeliama į kitą spiečiaus narį tuo pačiu metu yra artima nuliui. Daroma prielaida, kad kiekvienas spiečiaus narys atitinka minimalius CIF kokybinius reikalavimus. Šioje architektūroje, nebus naudojami automatizuotų konsensuso mechanizmai. Kiekvienu nariu yra visiškas pasitikėjimas. Pateikiama neformali algoritmo specifikacija, struktūros įkėlimui Algoritmas 1. Principinė architektūrinė schema pateikiama Pav. 10

Algoritmas 1 Neformali I architektūros specifikacija

1. Įkėlimo metu rankiniu būdu pasirenkamas aktyvus spiečiaus narys/ruožas į kurį norima deponuoti struktūrą.
2. Spiečiaus narys priima struktūrą.
3. Patikrina ar tokia struktūra jau nėra įkelta. Jei struktūra jau egzistuoja, ji atmetama.
4. Jei struktūra neegzistuoja, jei suteikiamas struktūros kandidatės identifikacijos numeris.
5. Atliekamas struktūros validavimas įskaitant kokybinį patikrinimą.
6. Jei struktūra atitinka minimalius kokybinius reikalavimus, jai suteikiamas unikalus identifikacijos numeris (*COD ID*) ir struktūra patalpinama į numatytą SVN repozitorijos šaką.
7. Išsiunčiamas pranešimas visiems spiečiaus nariams apie naujai įkeltą struktūrą.
8. Repozitorijos sinchronizuojamos tarp spiečiaus narių naudojant SVN funkcionalumą, užtikrinant vientisumą.
9. Papildomas kokybinis patikrinimas sinchronizavimo metu nėra atliekamas. Spiečiaus nariai pasitiki vienas kito priimtais sprendimais.
10. Visi spiečiaus nariai per atitinkamą laiką gauna naujai įkeltą struktūrą.
11. Jei buvo įvykęs tinklo padalinimas ar mazgas buvo neaktyvus, jam sugrįžus vykdomas papildomas repozitorijų sinchronizavimas, su bet kuriuo aktyviu mazgu. Galų gale visi mazgai turės vienodą informaciją (angl. *Eventually consistent*).
12. Esant neigiamam automatizuotam patikrinimui, struktūra siunčiama recenzavimui į recenzavimo posistemę.



10 pav. Architektūra be automatizuoto konsensuso algoritmo

Kiekvienas spiečiaus narys turi visų ruožų informaciją. Vartotojas turi galimybę gauti visas struktūras iš bet kurio spiečiaus nario, negarantuojant, kad visos struktūros yra naujausios versijos.

Privalumai:

1. Greitesnis struktūrų deponavimas. Struktūra nėra siunčiama kiekvienam nariui papildomam patikrinimui.
2. Vieno ar kelių narių sutrikimas neturi įtakos viso spiečiaus veikimui. Galimas struktūrų deponavimas esant ir vienam aktyviam nariui.
3. Galų gale pasiekiamas vientisumas (angl. *Eventually consistent*).
4. Paprastesnė realizacija, lyginant su architektūromis, kuriose naudojami konsensuso algoritmai.

Trūkumai:

1. Struktūros kokybės atitiktis priklauso nuo vieno nario. Kiti mazgai visiškai privalo pasitikėti vieni kitų sprendimais.
2. Didelė dublikatų tikimybė naudojant didelės spartos automatizuotas struktūrų surinkimo priemones.

9.2. Architektūrinis modelis panaudojant konsensuso algoritmą

Daroma prielaida, kad ta pati struktūra gali būti keliama vienu metu į skirtingus spiečiaus narius. Struktūrų surinkimui naudojami automatizuoti robotai. Nario gyvavimo laikas daug ilgesnis nei patvirtinimo laikas. Semantinis CIF patikrinimas gali skirtis. Pradinis narių skaičius priklauso nuo pasirinkto konsensuso algoritmo. Planuojama, kad sistema turi užtikrinti funkcionalumą esant ir vienam nariui, su galimybe nesudėtingai prijungti naujus mazgus.

Įvykus serverių padalinimui ir likus atskirtam spiečiaus nariui, kol bus atkurtas ryšys su kitais spiečiaus nariais, naujos struktūros į atskirtą narį gali būti nepriimamos, tačiau leidžia skaityti. Principinė architektūrinė schema pateikiama Pav. 11. Struktūros priėmimo eiga, esant naujai struktūrai Algoritmas 2:

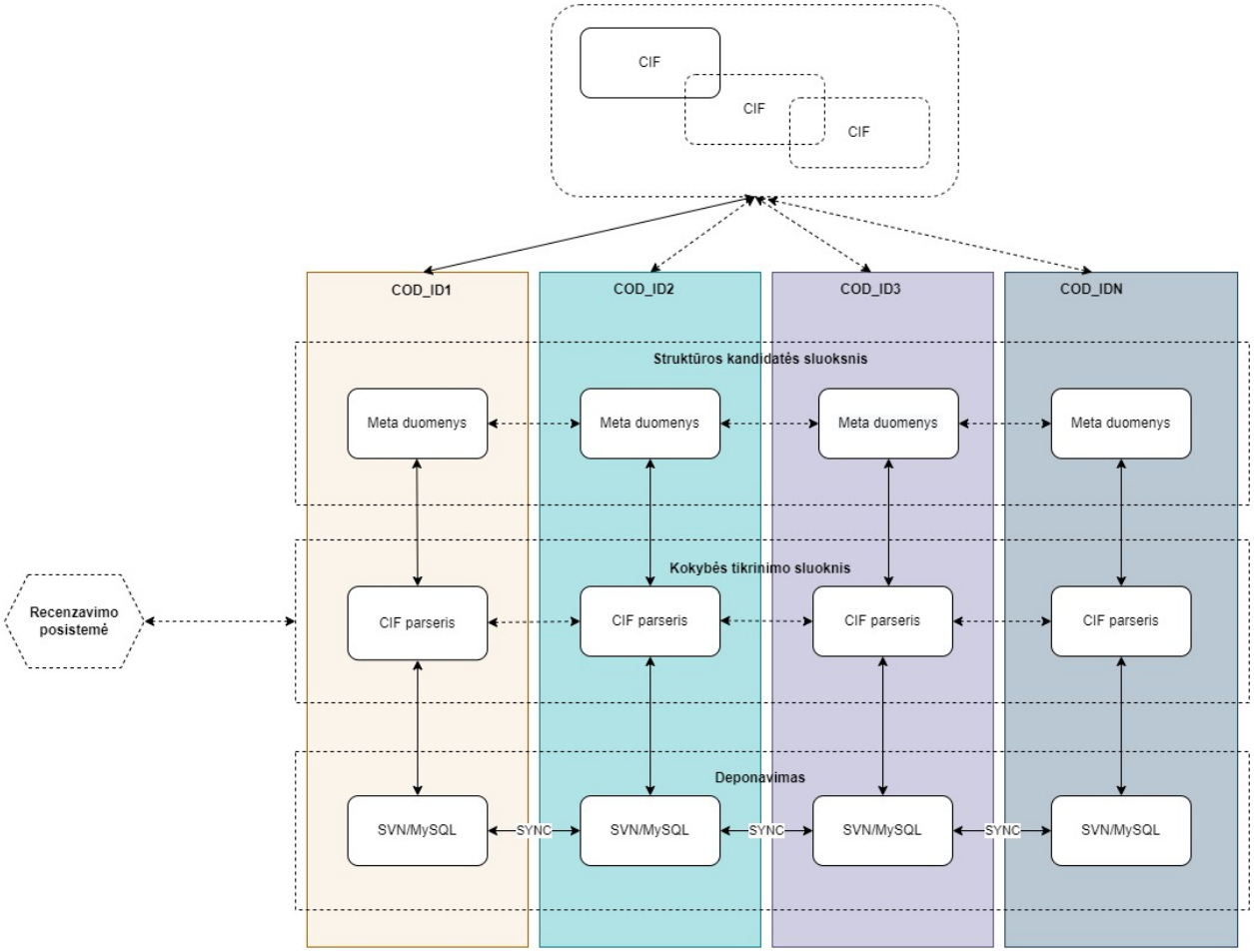
Algoritmas 2 Neformali II architektūros specifikacija, keliant naują struktūrą

1. Spiečiaus narys priima struktūrą.
2. Patikrina ar tokia struktūra jau yra deponuota spiečiuje.
3. Struktūra kandidatė išsiuntinėjama visiems COD nariams (struktūros kandidatės sluoksnis). Jei ta pati struktūra tuo pačiu metu keliama į kitą narį, pirmumą turi mažesnį identifikavimo numerį turintis narys.
4. Suteikiamas struktūros kandidatės numeris.
5. Kandidatas išsiuntinėjimas visiems COD nariams (kokybės tikrinimo sluoksnis).
6. Kandidatas priimamas tik kai patvirtina kvorumas.
7. Jei nariai nusprendžia nepraleisti struktūros, siunčiama papildomam patikrinimui į recenzavimo posistemę.
8. Struktūrą priėmęs narys suteikia *COD ID* iš savo ruožo ir pasidalina su kitais spiečiaus nariais.

Struktūros priėmimo eiga, esant naujai struktūros revizijai Algoritmas 3:

Algoritmas 3 Neformali II architektūros specifikacija, naujai struktūros revizijai

1. Spiečiaus narys priima struktūrą.
2. Patikrina ar tai yra naujesnė versija.
3. Jei ta pati struktūra tuo pačiu metu keliama į kitą narį, pirmumą turi mažesnį ID turintis narys.
4. Struktūrai suteikiamas struktūros kandidatės numeris.
5. Struktūrą kandidatę priėmęs narys, struktūra pasidalina su kitais aktyviais spiečiaus nariais.
6. Visi spiečiaus nariai tikrina CIF atitikimą kokybiniams reikalavimams ir siunčia patikrinimo rezultatus atgal.
7. Gavus teigiamą daugumos spiečiaus narių balsavimą, struktūrai suteikiamas COD ID ir struktūra įkeliama į repozitoriją.
8. Išsiunčiamas pranešimas visiems spiečiaus nariams apie naują struktūrą.
9. Nauja struktūra deponuojama visuose spiečiaus nariuose.
10. Vientisumas tarp mazgų, užtikrinimas SVN funkcionalumu.



11 pav. Išskirstyta architektūra su konsensuso algoritmu

Šioje architektūroje, struktūros deponavimas pirmiausia vykdomas struktūrą priimančiame spiečiaus naryje. Įvykus sėkmingai operacijai, struktūra sinchronizuojama tarp spiečiaus narių. Papildomas patikrinimas nėra atliekamas. Struktūros atitikimas sutartiems reikalavimams vykdomas pirmuose dviejuose sluoksniuose.

Privalumai:

1. Galimas didelės apimties struktūrų pateikimas išvengiant dublikatų.
2. Vieno ar kelių narių laikini sutrikimai neturi įtakos viso spiečiaus veikimui. Leidžiamas neaktyvių narių skaičius priklauso nuo bendro narių skaičiaus.
3. Struktūros tinkamumo patikrinimą atlieka visi aktyvūs spiečiaus nariai, todėl išlaikoma aukšta duomenų kokybė.
4. Pirmieji sluoksniai turi minimalų informacijos kiekį, reikiamą nustatyti dublikatams ir ar jau tokia struktūra egzistuoja.

Trūkumai:

1. Ilgesnis struktūros deponavimo laikas, esant didesniam, geografiškai plačiai išdėstytam spiečiaus narių tinklui.
2. Sudėtingesnė sprendimo priežiūra lyginant su pirma architektūra.

Šio architektūrinio modelio struktūros deponavimo ir saugojimo sluoksniams taip pat galėtų būti naudojamas Raft algoritmo funkcionalumas – žurnalinių įrašų replikavimas, kuris užtikrintų vientisumą po padalinimo, tačiau vis tiek būtų naudojama SVN repozitorija, kaip saugojimo pagrindas, todėl toks pakeitimas būtų perteklinis.

10. Realizacijos aprašas

10.1. Architektūros pasirinkimas

Pasirinkta serverių spiečių realizuoti pagal II architektūrą. Šį pasirinkimą lemia šie kriterijai:

1. Automatizuotas konsensuso mechanizmas leis kiekvienam nariui naudoti didelės spartos automatizuotus struktūrų surinkimo robotus (angl. *Web bot*) išvengiant dublikatų.
2. Atskirų sluoksnių architektūra palengvins klaidų paiešką ir leis aptikti „blogus“ narius.
3. SVN funkcionalumas užtikrins vientisumą esant stabiliam sistemos veikimo rėžimui ir vientisumo užtikrinimą po padalinimo. Taip pat kaip ir esamoje realizacijoje užtikrins kiekvienos revizijos išsaugojimą.
4. Elixir programavimo kalba turi standartinius mechanizmus dirbti išskirstytoje aplinkoje, todėl tai užtikrins sistemos stabilumą ir automatinius veiksmus esant sutrikimams.

Serverių spiečiaus išskirstytos sistemos realizacijai pasirinkta naudoti Elixir programavimo kalbą ir Raft protokolą [LAF21]. Raft protokolas gali veikti net ir esant tik vienam nariui, todėl nereikia papildomų algoritmų, skirtingoms konfigūracijoms valdyti. Elixir programavimo kalba taip pat suteikia galimybę paprastai iškviesti kitomis programavimo kalbomis parašytus vykdomuosius failus, pvz.:„cod-tools“ [MVB+16], kurie parašyti Perl programavimo kalba. Taip pat Elixir turi numatytus mechanizmus dirbti išskirstytos sistemos rėžimu ir valdyti narius (angl. *Supervisor*), esant sutrikimui juos perkrauti, taip leidžiant atstatyti sistemos darbą be žmogaus įsikišimo. Duomenų saugojimo sluoksnis bus sukurtas esamos realizacijos pagrindu, kaip technologinius sprendinius naudojant MySQL ir SVN. SVN priemonėmis bus užtikrinimas vientisumas ir vientisumas po sistemos padalinimo tarp visų spiečiaus narių, net ir tuo atveju, jei vienas ar daugiau narių ilgesnį laiką bus neaktyvūs. 3 lentelėje pateikiama toleruojamų gedimų skaičiaus priklausomybė nuo narių skaičiaus.

2 lentelė. Galimų gedimų priklausomybė nuo narių skaičiaus RAFT protokole.

Narių skaičius	Dauguma	Toleruojamas gedimų skaičius
1	1	0
2	2	0
3	2	1
4	3	1
5	3	2
6	4	2
7	4	3

RAFT algoritme nariai gali būti trijose būsenose: lyderis, sekėjas ir kandidatas. Lyderis yra būtina rolė algoritmo veikimui. Prieš pradedant veikti sistemai lyderis privalo būti išrinktas.

Kiekvieno mazgas prisijungia nuo sekėjo būsenos. Pasibaigus numatytam laikui, jis pereina į kandidato būseną – tai reiškia, kad mazgas dabar yra tinkamas kandidatas tapti lyderiu. Kai kandidatas gauna aiškią daugumą balsų, jis pereina į lyderio būseną. Jei per rinkimų procesą nėra aiškaus laimėtojo, kandidatui vėl baigiasi laikas ir prasideda nauji rinkimai.

Jei yra tinklo padalinimas, dabartinis lyderis gali būti atjungtas nuo daugumos, tuomet dauguma išsirenka naują lyderį. Kai prieš tai buvęs lyderis grįžta ir aptinka, kad naujas lyderis jau yra išrinktas su aukštesne kadencija, senasis lyderis atsistatydina ir tampa pasekėju. Nelyginis mazgų skaičius sudaro geresnę kombinaciją nei lyginis mazgų skaičius, nes jis yra labiau atsparus gedimams.

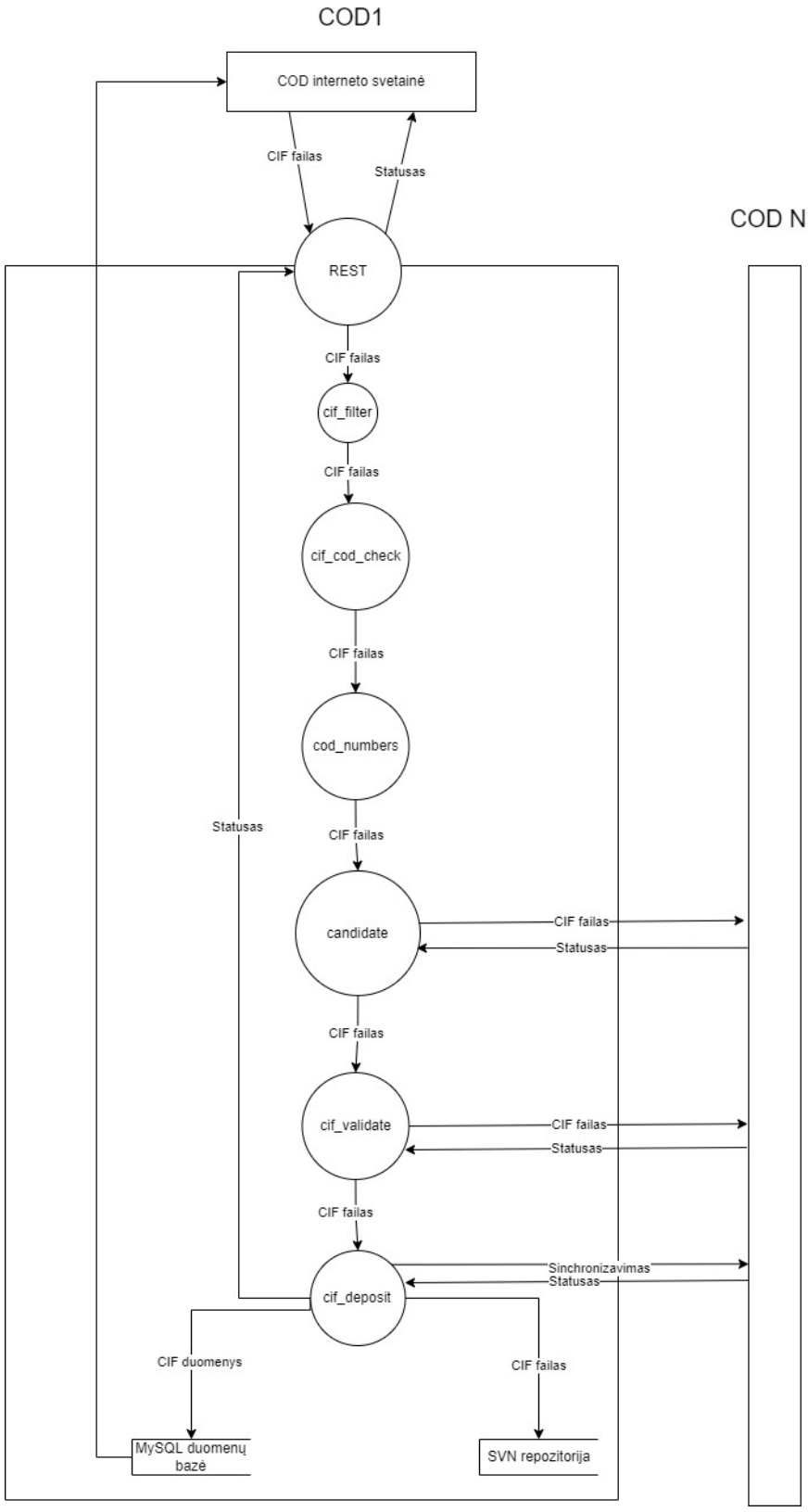
Raft algoritmo biblioteka nebus kuriama, o šios sistemos įgyvendinimui bus naudojama jau sukurta realizacija [Rab23]. Naudojama biblioteka turi reikalingus algoritmo funkcionalumus, kurie bus reikalingi serverių spiečiaus įgyvendinimui. Taip pat biblioteka yra ištestuota, laikomasi semantinio versijavimo ir paruošti naudoti darbinėje aplinkoje. Žemiau 3 lentelėje pateikiamas apibendrintas sistemos veikimo modelis.

3 lentelė. Sistemos sluoksnių veikimo modelis

Sluoksnis	**Konsensusas**	**Detalizacija**
COD svetainė	Ne	Kiekvienas mazgas turi vienas nuo kito nepriklausomą interneto svetainę.
REST	Ne	CIF failas per interneto naršyklę pateikiamas per REST sąsają. Kiekvienas narys turi savo sąsają.
Struktūros kandidatės	Dauguma	CIF failas priimamas tik tada, kai priimamas teigiamas sprendimas daugumos narių.
Kokybės patikrinimo	Dauguma	CIF failas priimamas tik tada, kai failas priimamas daugumos narių.
Saugojimo	Dauguma	Mazgas lyderis, po sėkmingo deponavimo, siunčia sinchronizavimo žinutę visiems aktyviems nariams.

10.2. Duomenų srautai

Pateikiama detali CIF failo tikrinimo eiga, bei komunikacija su kitais spiečiaus nariais. Pirminis struktūros patikrinimas atliekamas COD interneto svetainėje. Po to per sąsają duomenys pateikiami Elixir sluoksniams, kur atliekami struktūros sintaksės, dublikatų patikrinimai, struktūra išsiunčiama kitiems nariams. Praėjus visus žingsnius – struktūra sinchronizuojama su visais aktyviais nariais. Vartotojui išskirstyti sluoksniai nėra matomi.



12 pav. Duomenų srautų diagrama

10.3. Sistemos realizacija ir testavimas

Kaip buvo numatyta „Architektūros“ skyriuje, išskirstyta sistema realizuota *Elixir* programavimo kalba. Darbui su CIF failais sukurtos funkcijos, kviečiančios „cod-tools“ įrankius. Sukurtas sistemos prototipas iš esmės atitinkantis numatytą architektūrinį sprendinį, su apribojimais:

1. Reikalingas papildomas mazgų gyvybingumo stebėjimas angl. (*Monitoring*). Tai užtikrintų savalaikę reakciją, esant nenumatytoms situacijoms.

2. Nėra sąsajos struktūros pateikimui į recenzavimo posistemę.

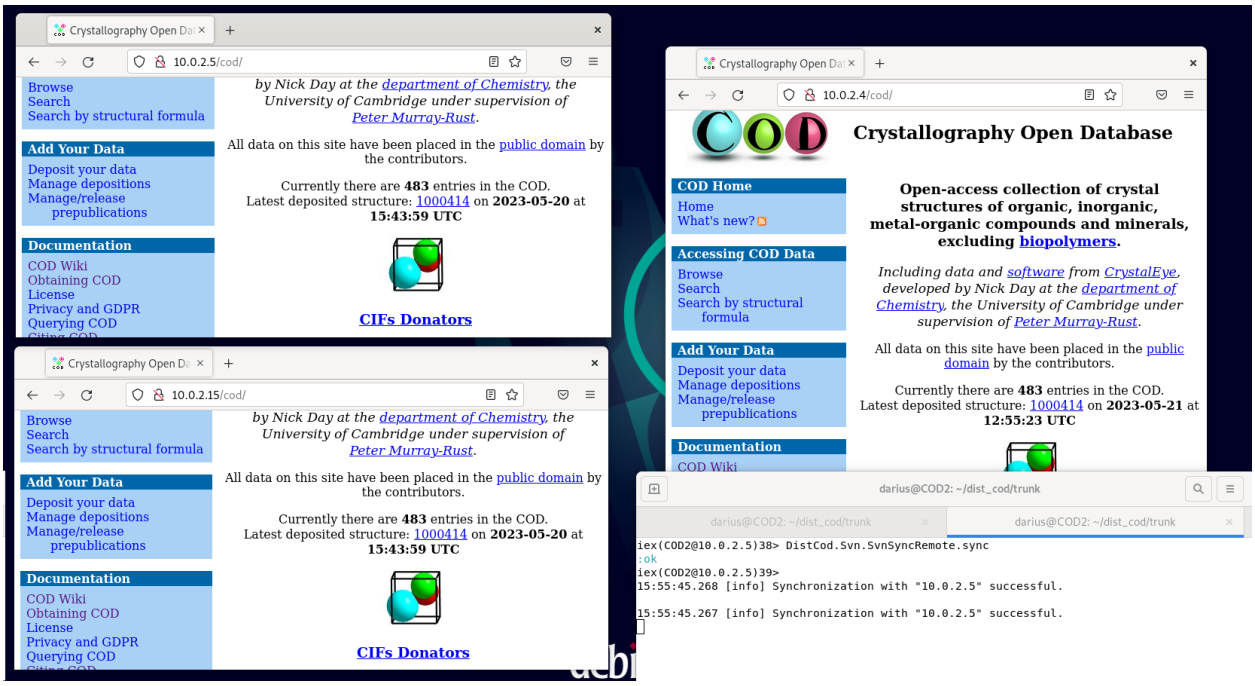
Panaudojant Raft algoritmą ir standartinius Elixir komunikavimo mechanizmus, sukurta sistemos veikimo logika. Raft algoritmo veikimas užtikrina, kad tik mazgas lyderis atliks deponavimą ir po deponavimo išsiųs signalą kitiems atsinaujinti repozitorijų kopijas, taip sumažinant repozitorijos išsišakojimų riziką. Papildomai naudojami SVN konfigūraciniai failai „pre-commit hooks“, kurie užtikrina, kad bet koks deponavimas, taip pat ir sinchronizavimas, galimas tik gavus nurodymą iš mazgo lyderio. Elixir išskirstyti sluoksniai atlieka ne tik CIF failų tikrinimą, bet taip pat valdo SVN repozitorijas. Struktūrų saugojimo sluoksniui panaudota esamos sistemos realizacija. Elixir išskirstyti sluoksniai ir COD duomenų saugojimo sluoksnis sėkmingai integravosi.

Esant mazgo lyderio sutrikimui, automatizuotai išrenkamas naujas lyderis. Narių pasikeitimai yra sekami ir inicijuojamos funkcijos, kurios perjungia pagrindinę repozitoriją naujai išrinktam lyderiui, siunčia signalą sinchronizavimui. Sistemos architektūra modulinė, todėl nesudėtinga pridėti naujai sukurtas funkcijas. Tai leis nesudėtingą plėtrą ateityje, neperkuriant visos sistemos.

10.3.1. Prototipo testavimas

Išskirstyta sistema testuojama sukuriant mazgus virtualioje aplinkoje. Visuose mazguose sudiegiamas programinis kodas, COD sistemos duomenų saugojimo sluoksnis. Inicijuojamas išskirstytos sistemos darbas. Patikrinamas visų mazgų gyvybingumas. Siekiant užtikrinti tinkamą veikimą, sistemos prototipas testuojamas keliais būdais:

- Automatizuoti testai – automatizuotais testais, atliekamas atskirų sistemos komponentų testavimas, pvz.: funkcijų, kurios skirtos dirbti su CIF failais. Taip pat naudojant vietinio spiečiaus (angl. *Local cluster*) funcionalumą, testuojamas išskirstytų sluoksnių veikimas. Padengimas automatizuotais testais ≈60%. Papildomai paruoštas skriptas, didelės apimties struktūrų pateikimui. Teikiamos įvairios struktūros, neatsižvelgiant į korektiškumą. Prie spiečiaus prijungiamas narys, su atsiliekančia revizija. Duomenys sėkmingai sinchronizuojami Pav. 13.



13 pav. Spiečiaus narių sinchronizavimas.

- Rankinis testavimas – atskiri sistemos komponentai taip pat testuojami rankiniu būdu. Elixir suteikia galimybę bet kurią funkciją iškviesti komandinėje eilutėje. Šio testavimo metu taip pat buvo tikrinama:
  - Sintaksiškai neteisingos struktūros deponavimas, kurios negali ištaisyti numatytos funkcijos. Laukiamas rezultatas – struktūra atmetama pirminio patikrinimo metu.
  - Struktūros deponavimas, kurios sintaksė gali būti pataisoma numatytais įrankiais. Laukiamas rezultatas – struktūra siunčiama tinklu kitiems nariams.
  - Korektiškos struktūros deponavimas. Laukiamas rezultatas – struktūra sėkmingai priimama visuose mazguose.

Rezultatai

1. Atliktas COD duomenų bazės apžvalga, aprašyti jos ypatumai. COD duomenų bazėje pirmumą teikiamas duomenų prieinamumas. Atlikta saugomų failų analizė, atsižvelgiant į jų kiekį ir dydį. Didžioji failų dalis ≈65% yra tarp 11-50 kilobaitų dydžio, tačiau pasitaiko ir daug didesnių failų. COD nėra standartinė SQL duomenų bazė, o kompleksas informatikinių sprendimų: MySQL, SVN, cod-tools, kurie sukurti skirtingomis programavimo kalbomis ir sąveikauja tarpusavyje.

2. Atliktas COD tyrimas remiantis FAIR principais. Išsiaiškinta, kad esamos COD realizacijos technologiniai ir procedūriniai principai atitinka FAIR reikalavimus. COD interneto svetainėje pateikiama išsami informacija, kaip pasiekti duomenis, sistemoje saugomos ir prieinamos visos revizijos, duomenys licencijuojami CC0 licencija. Nesudėtingai galima atsisiųsti tiek pavienius failus tiek ir visą duomenų bazę.

3. Atliktas atkuriamumo problemos tyrimas. Šiai problemai spręsti taikomos procedūrinės ir technologinės priemonės, siekiant palengvinti tyrimų dokumentavimą. Atsižvelgiant į tai sudarytos priėmimo - išleidimo taisyklės, kurių privalės laikytis tyrėjai, norintys prisijungti prie COD serverių spiečiaus.

4. Sudarytos dvi galimos naujos sistemos architektūros.

   (a) Abu architektūriniai modeliai tenkina minimalius reikalavimus, tačiau tolimesniam įgyvendinimui pasirinktas antras architektūrinis modelis su automatizuotais konsensuso algoritmais, pagal 10.1 skyriuje nurodytus kriterijus. Išskirstytos sistemos realizavimui parinkta Elixir programavimo kalba ir Raft algoritmas. CIF failų deponavimui ir saugojimui panaudoti esamos sistemos realizacijos sprendimai. SVN pagalba užtikrinamas duomenų vientisumas ir vientisumas po padalinimų.

   (b) Pagal žinomus duomenis atliktas eksponentinio nykimo modeliavimas.Keičiant modelio parametrus, simuliuotos galimos skirtingos sistemos būsenos. Papildomai sumodeliuota stacionarinė sistemos būsena.

5. Realizuotas ir ištestuotas prototipas pagal II architektūrinį sprendinį. Testavimui panaudoti keturi mazgai sukurti virtualioje aplinkoje vietiniame tinkle. Parašytas skriptas, kuris siunčia CIF failus į sistemos priėmimo sąsają. Testavimui panaudota daugiau kaip 1000 įvairių struktūrų. Išjungiant ir vėl prijungiant mazgus simuliuoti galimi sistemos sutrikimai.

Išvados

1. COD tipo mokslo duomenų bazėms nėra tinkami tipiniai sprendiniai, kaip atsarginės kopijos, duomenų bazės replikavimas, nes neužtikrina nuolatinio prieinamumo.

2. Spiečiaus architektūra užtikrina FAIR principus nenutrūkstamai.

3. Sutartų taisyklių laikymasis ir automatizuotas tyrimo duomenų kokybės tikrinimas padeda pakartotinai atlikti tyrimus.

4. Atlikus modeliavimą, išsiaiškinta, kad didelis pradinis mazgų skaičius neužtikrina gyvavimo trukmės. Būtina sąlyga siekiamam ilgaamžiškumui ($\approx$100 m.) užtikrinti – naujų mazgų atsiradimas per pusėjimo trukmę. Tai galimai užtikrins technologiškai nesudėtingas prisijungimas prie serverių spiečiaus.

5. Sukurtas prototipas užtikrino vientisumą po padalinimų ar naujai prijungus mazgą. Sistema sudaryta iš atskirų sluoksnių, tarp kurių nėra griežtų ryšių, todėl sutrikimai kurie susiję tik su išskirstytų sluoksnių veikimu, nedaro įtakos duomenų prieinamumui. Sistemos prototipas ištestuotas vietiniame tinkle su 4 mazgais. Reikalingas papildomas testavimas geografiškai išskirstytoje aplinkoje.

Šaltinių sąrašas

[AM13] Omar H. Alhazmi ir Yashwant K. Malaiya. Evaluating disaster recovery plans using the cloud. *2013 Proceedings Annual Reliability and Maintainability Symposium (RAMS)*, p.p. 1–6, 2013. DOI: 10.1109/RAMS.2013.6517700.

[AS04] Stephanos Androutsellis-Theotokis ir Diomidis Spinellis. A Survey of Peer-to-Peer Content Distribution Technologies. *ACM Comput. Surv.*, 36(4):335–371, 2004-12. ISSN: 0360-0300. DOI: 10.1145/1041680.1041681. URL: https://doi.org/10.1145/1041680.1041681.

[Bar10] Nick Barnes. Publish your computer code: it is good enough. *Nature*, 467(7317):753–753, 2010-10. DOI: 10.1038/467753a.

[BBB+20] Stephen K Burley, Charmi Bhikadiya, Chunxiao Bi, Sebastian Bittrich ir k.t. RCSB Protein Data Bank: powerful new tools for exploring 3D structures of biological macromolecules for basic and applied research and education in fundamental biology, biomedicine, biotechnology, bioengineering and energy sciences. *Nucleic Acids Research*, 49(D1):D437–D451, 2020-11. ISSN: 0305-1048. DOI: 10.1093/nar/gkaa1038. eprint: https://academic.oup.com/nar/article-pdf/49/D1/D437/35364241/gkaa1038.pdf. URL: https://doi.org/10.1093/nar/gkaa1038.

[BCH+20] Daina R. Bouquin, Daniel A. Chivvis, Edwin Henneken, Kelly Lockhart, August Muench ir Jennifer Koch. Credit Lost: Two Decades of Software Citation in Astronomy. *The Astrophysical Journal Supplement Series*, 249(1):8, 2020-06. DOI: 10.3847/1538-4365/ab7be6. URL: https://doi.org/10.3847/1538-4365/ab7be6.

[Bre12] Eric Brewer. CAP twelve years later: How the "rules" have changed. *Computer*, 45(2):23–29, 2012-02. DOI: 10.1109/mc.2012.37.

[CGH+18] Sandra Collins, Françoise Genova, Natalie Harrower, Simon Hodson, Sarah Jones, Leif Laaksonen, Daniel Mietchen, Rūta Petrauskaitė ir Peter Wittenburg. Turning FAIR into reality. Tech. atask., European Commission Expert Group on FAIR Data, 2018. DOI: 10.2777/1524. URL: https://www.eoscsecretariat.eu/sites/default/files/ki0618206enn.en_.pdf.

[CLS16] Saksham Chand, Yanhong A. Liu ir Scott D. Stoller. Formal Verification of Multi-Paxos for Distributed Consensus. John Fitzgerald, Constance Heitmeyer, Stefania Gnesi ir Anna Philippou, redaktoriai, *FM 2016: Formal Methods*, p.p. 119–136, Cham. Springer International Publishing, 2016. ISBN: 978-3-319-48989-6.

[Com09] Creative Commons. CC0 1.0 Universal (CC0 1.0) Public Domain Dedication, 2009. URL: https://creativecommons.org/publicdomain/zero/1.0/.

[DDA+12] Nick Day, Jim Downing, Sam Adams, N. W. England ir Peter Murray-Rust. *CrystalEye*: automated aggregation, semantification and dissemination of the world's open crystallographic data. *Journal of Applied Crystallography*, 45(2):316–323, 2012-04. DOI: 10.1107/S0021889812006462. URL: https://doi.org/10.1107/S0021889812006462.

[FPF+14] Lars Frank, Rasmus Ulslev Pedersen, Christian Havnø Frank ir N. Jesper Larsson. The CAP theorem versus databases with relaxed ACID properties. ACM Press, 2014. ISBN: 9781450326445. DOI: 10.1145/2557977.2557981. URL: http://dx.doi.org/10.1145/2557977.2557981.

[Gan15] Deka Ganesh Chandra. BASE analysis of NoSQL database. *Future Generation Computer Systems*, 52:13–21, 2015. ISSN: 0167-739X. DOI: https://doi.org/10.1016/j.future.2015.05.003. URL: https://www.sciencedirect.com/science/article/pii/S0167739X15001788. Special Section: Cloud Computing: Security, Privacy and Practice.

[GDM+11] Saulius Gražulis, Adriana Daškevič, Andrius Merkys, Daniel Chateigner ir k.t. Crystallography Open Database (COD): an open-access collection of crystal structures and platform for world-wide collaboration. *Nucleic Acids Research*, 40(D1):D420–D427, 2011-11. ISSN: 0305-1048. DOI: 10.1093/nar/gkr900. URL: https://doi.org/10.1093/nar/gkr900.

[HAB91] S. R. Hall, F. H. Allen ir I. D. Brown. The crystallographic information file (CIF): a new standard archive file for crystallography. *Acta Crystallographica Section A*, 47(6):655–685, 1991-11. DOI: 10.1107/S010876739101067X. URL: https://doi.org/10.1107/S010876739101067X.

[HJ11] Robin Hecht ir Stefan Jablonski. NoSQL evaluation: A use case oriented survey. *2011 International Conference on Cloud and Service Computing*, p.p. 336–341, 2011. DOI: 10.1109/CSC.2011.6138544.

[LAF21] Matthew Alan Le Brun, Duncan Paul Attard ir Adrian Francalanza. Graft: General Purpose Raft Consensus in Elixir. *Proceedings of the 20th ACM SIGPLAN International Workshop on Erlang*, Erlang 2021, p.p. 2–14, Virtual, Republic of Korea. Association for Computing Machinery, 2021. ISBN: 9781450386128. DOI: 10.1145/3471871.3472963. URL: https://doi.org/10.1145/3471871.3472963.

[Lam01] Leslie Lamport. Paxos Made Simple. *ACM SIGACT News (Distributed Computing Column) 32, 4 (Whole Number 121, December 2001)*:51–58, 2001-12.

[Lam19] Leslie Lamport. *The Part-Time Parliament*. *Concurrency: The Works of Leslie Lamport*. Association for Computing Machinery, New York, NY, USA, 2019, p.p. 277–317. ISBN: 9781450372701. URL: https://doi.org/10.1145/3335772.3335939.

[LBL+21] Edward A. Lee, Soroush Bateni, Shaokai Lin, Marten Lohstroh ir Christian Menard. Quantifying and Generalizing the CAP Theorem, 2021. arXiv: 2109.07771 [cs.DC].

[LGK+20] Anna-Lena Lamprecht, Leyla Garcia, Mateusz Kuzak, Carlos Martinez ir k.t. Towards FAIR principles for research software. *Data Science*, 3:37–59, 2020. ISSN: 24518492, 24518484. DOI: 10.3233/DS-190026.

[LXD15] Gregory Levitin, Liudong Xing ir Yuanshun Dai. Optimal backup frequency in system with random repair time. *Reliability Engineering and System Safety*, 144:12–22, 2015. ISSN: 0951-8320. DOI: https://doi.org/10.1016/j.ress.2015.06.014. URL: https://www.sciencedirect.com/science/article/pii/S0951832015001891.

[MVB+16] Andrius Merkys, Antanas Vaitkus, Justas Butkus, Mykolas Okulič-Kazarinas, Visvaldas Kairys ir Saulius Gražulis. *COD::CIF::Parser*: an error-correcting CIF parser for the Perl language. *Journal of Applied Crystallography*, 49(1):292–301, 2016-02. DOI: 10.1107/S1600576715022396. URL: https://doi.org/10.1107/S1600576715022396.

[Pen11] Roger D. Peng. Reproducible Research in Computational Science. *Science*, 334:1226–1227, 2011. DOI: 10.1126/science.1213847. eprint: http://www.sciencemag.org/content/334/6060/1226.full.pdf. URL: http://www.sciencemag.org/content/334/6060/1226.abstract.

[Pri10] Heather Pringle. Space science. NASA dives into its past to retrieve vintage satellite data. *Science (New York, N.Y.)*, 327:1322–3, 2010. DOI: 10.1126/science.327.5971.1322. URL: http://www.sciencemag.org/content/327/5971/1322.short.

[Rab23] RabbitMQ. A Raft implementation for Erlang and Elixir that strives to be efficient and make it easier to use multiple Raft clusters in a single system. 2023. URL: https://github.com/rabbitmq/ra.

[Rip01] M. Ripeanu. Peer-to-peer architecture case study: Gnutella network. *Proceedings First International Conference on Peer-to-Peer Computing*, p.p. 99–100, 2001. DOI: 10.1109/P2P.2001.990433.

[SKC00] M. Schwab, N. Karrenbach ir J. Claerbout. Making scientific computations reproducible. *Computing in Science Engineering*, 2:61–67, 2000. ISSN: 1521-9615. DOI: 10.1109/5992.881708.

[SKN+16] Arfon M. Smith, Daniel S. Katz, Kyle E. Niemeyer ir FORCE11 Software Citation Working Group. Software citation principles. *PeerJ Computer Science*, 2:e86, 2016-09. ISSN: 2376-5992. DOI: 10.7717/peerj-cs.86. URL: https://doi.org/10.7717/peerj-cs.86.

[SWK+20] Suryanarayana Sankagiri, Xuechao Wang, Sreeram Kannan ir Pramod Viswanath. The Checkpointed Longest Chain: User-dependent Adaptivity and Finality. *CoRR*, abs/2010.13711, 2020. arXiv: 2010.13711. URL: https://arxiv.org/abs/2010.13711.

[TPP13] Clarence JM Tauro, Baswanth Rao Patil ir KR Prashanth. A comparative analysis of different nosql databases on data model, query model and replication model. *Proceedings of the International Conference on ERCICA*, 2013.

[VBT+21] Foteini Valeonti, Antonis Bikakis, Melissa Terras, Chris Speed, Andrew Hudson-Smith ir Konstantinos Chalkias. Crypto Collectibles, Museum Funding and OpenGLAM: Challenges, Opportunities and the Potential of Non-Fungible Tokens (NFTs). *Applied Sciences*, 11(21), 2021. ISSN: 2076-3417. DOI: 10.3390/app11219931. URL: https://www.mdpi.com/2076-3417/11/21/9931.

[Vog09] Werner Vogels. Eventually Consistent. *Commun. ACM*, 52(1):40–44, 2009-01. ISSN: 0001-0782. DOI: 10.1145/1435417.1435432. URL: https://doi.org/10.1145/1435417.1435432.

[WDA+16] Mark D. Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton ir k.t. The FAIR guiding principles for scientific data management and stewardship. *Scientific Data*, 3(1), 2016-03. DOI: 10.1038/sdata.2016.18.

[WOY+19] Shuai Wang, Liwei Ouyang, Yong Yuan, Xiaochun Ni, Xuan Han ir Fei-Yue Wang. Blockchain-Enabled Smart Contracts: Architecture, Applications, and Future Trends. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 49(11):2266–2277, 2019. DOI: 10.1109/TSMC.2019.2895123.

[Žyg20] LRT TV naujienų tarnyba Žygintas Abromaitis. Po Registrų centro užliejimo – dėmesys šalia vykstančioms statyboms: dėl jų galimai trūko sienos, o nuotekų sistema neatlaikė apkrovos, 2020. URL: https://www.lrt.lt/naujienos/lietuvoje/2/1223785/po-registru-centro-uzliejimo-demesys-salia-vykstancioms-statyboms-del-ju-galimai-truko-sienos-o-nuoteku-sistema-neatlaike-apkrovos.

Sąvokų apibrėžimai

1. Serverių spiečius – geografiškai nutolusių, daugelio serverių sistema, kurioje saugomi kristalografijos duomenys.
2. Pusėjimo trukmė – laiko tarpas, per kurį išnyksta pusė spiečiaus narių.
3. cod-tools – atviro kodo programinių įrankių rinkinys, darbui su kristalografine informacija.

Santrumpos

1. COD – atvira kristalografijos duomenų bazė (angl. *„Crystalography Open Database“*).
2. CAP – CAP teoremoje teigia, kad bet kuri paskirstyta duomenų saugykla gali užtikrinti tik dvi iš šių trijų savybių (prieinamumas arba vientisumas esant tinklo padalinimams) vienu metu.
3. SVN – versijų valdymo sistema.
4. Raft – išskirstytų sistemų konsensuso algoritmas.
5. CIF – kristalografinės informacijos duomenų failo formatas.
6. TPM – patikimos platformos modulis.
7. P2P – klientas-klientas sistema.
8. BASE – NoSQL duomenų bazių savybės, teikiant pirmumą prieinamumui prieš vientisumą.
9. NFT – nekintamieji žetonai.
10. FAIR – duomenų saugojimo principai, apibrėžiantys randamumą, prieinamumą, sąveikumą ir pakartotinį panaudojimą.
11. F/LOSS – laisva ir atvirojo kodo programinė įranga.
12. ACID – atomiškumas (angl. *Atomicity*), vientisumas (angl. *Consistency*), izoliavimas (angl. *Isolation*), patvarumas(angl. *Durability*).
13. PDB – baltymų duomenų bankas (angl. *Protein Data Bank*)