VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMATIKOS KATEDRA

Baigiamasis bakalauro darbas

NoSQL duomenų bazių plėtimas
(NoSQL database scaling)

Atliko: 4 kurso, 2 grupės studentas
Norbert Žardin (parašas)
Darbo vadovas:
lekt. Liutauras Ričkus (parašas)
Recenzentas:
doc. Linas Laibinis (parašas)

Vilnius
2017

Turinys

Įvadas

Informacija šiais laikais yra viskas. Begalinis informacijos srautas privertė žmones susimąstyti kaip tą informaciją efektyviai saugoti, kadangi popierinis ir kitokie fiziniai duomenų kaupimo variantai reikalauja daug erdvės, priežiūros ir darbo. Daug dėmesio skirta informacijos saugojimui skaitmeniniu formatu. Patobulėjus technologijoms buvo suteiktos palankios sąlygos duomenų bazėms ir būdų kiekis kaip saugoti duomenis augo. Reliacinės duomenų bazės tai bene pirmas geriausias bandymo kaupti informaciją skaitmeniniu formatu rezultatas. Reliacinė duomenų bazė tai skaitmeninė struktūra sudaryta iš schemos, lentelių, lentelių stulpelių ir įrašų. Šio derinio dėka galima sistemiškai kaupti informaciją ir esant reikalui ją nesunkiai pasiekti.

Daug metų IT sferoje dominavo reliacinės duomenų bazės, o viskas pasikeitė, kai reliacinės duomenų bazės nebegalėjo išlaikyti joms suteikiamo krūvio arba dirbo nepakankamai efektyviai, kad sistema kuriai duomenų bazė naudojama tenkintų visus lūkesčius. NoSQL duomenų bazės buvo ir yra sprendimas šiai problemai, kadangi jų duomenų saugojimo struktūros leidžia efektyviau dirbti su duomenimis, dėl ko pagerėja našumas ir kitos duomenų bazių charakteristikos.

Šiandien, kai duomenys plūsta srautais ir jiems nėra galo, duomenų saugojimas yra dažniausiai aptariamas klausimas. Saugoti be galo daug duomenų vienoje vietoje yra neįmanoma, tai lygiai tokia pati situacija kaip ir su fiziniais objektais saugančiais informaciją, todėl privalo būti priimti įvairūs architektūriniai sprendimai. Tam, kad išspręsti problemą galima paprasčiausiai pridėti papildomų resursų. Tačiau, sistemoje, kurioje yra duomenų bazė, resursai yra ne begaliniai ir su laiku ateina metas, kuomet privaloma daryti pokyčius fiziniams resursams ir duomenų bazės struktūroje. Prie tokios situacijos gali privesti ir atvejis, kai bandoma užtikrinti duomenų saugumą, duomenų kopijų saugojimas reikalauja nemažai papildomų resursų, kadangi tas duomenų kopijas reikia ne tik saugoti bet ir pasiekti. Kilus šiai problemai duomenų bazių paskirstymas yra geriausias būdas aprūpinti duomenų bazę resursais ir funkcionalumu.

Duomenų bazės skaldymas tai toks procesas, kai duomenų bazė padalinama dalimis ir patalpinama per kelis įrenginius. Skaldant duomenų bazę, duomenims yra suteikiama daugiau „erdvės“ kur jie gali būti patalpinti, o duomenų bazei suteikiama daugiau resursų, su kuriais galima dirbti. Taip priklausomai nuo duomenų kiekio galima plėsti duomenų bazės tinklą tiek, kiek reikės. Tačiau, toks sprendimas ne visada yra optimalus, kadangi plečiant duomenų bazę gali kisti duomenų bazės našumas ir pabaigoje jo gali neužtekti. Todėl, kartais gali užtekti pridėti papildomų resursų duomenų bazei. Pridėjus operatyvios atminties, geresnį procesorių ir daugiau disko atminties, skaidymo galima išvengti. Būtent todėl, duomenų bazių plėtimui reikia atlikti nuodugnią analizę prieš apsistojant ties tam tikru variantu.

Pagrindinis šio darbo tikslas yra išanalizuoti NoSQL duomenų bazių plėtimo būdus. Pagrindiniai darbo uždaviniai:

a) Apžvelgti NoSQL duomenų bazes ir NoSQL duomenų bazių plėtimo būdus;
b) Apžvelgti duomenų bazes, kurioms atliekamas tyrimas, ir tų duomenų bazių testavimo įrankius;
c) Nustatyti pagrindinius kaštus, kurių pagalba galima būtų lyginti duomenų bazes;
d) Pasinaudojant įrankiais, atlikti pagrindinius duomenų bazių testus pagal nustatytus kaštus;
e) Išanalizuoti gautus rezultatus ir pateikti išvadas.

Gauti rezultatai parodo esminius skirtumus tarp duomenų bazių plėtimo būdų, neatlikus jokių optimizacijų. Remiantis tais pačiais metodais galima išmatuoti kaštus ir kitokioms situacijoms.

1. NoSQL duomenų bazės

Pirmą kartą terminas NoSQL buvo paminėtas Carlo Strozzi 1998 metais, kad pavadinti jo Strozzi NoSQL atviro kodo reliacinę duomenų bazę, kuri neatskleidė pagrindinių standartinės SQL duomenų bazės bruožų, tačiau buvo vis dar reliacinė sistema. Eric Evans iš naujo pristatė terminą „NoSQL“ 2009 metais renginyje, aptariant platinamas atviro kodo, nereliacines duomenų bazes. [LM10]

Įprastos SQL duomenų bazės yra reliacinės duomenų sistemos, joms būdingos lentelės užpildytos duomenimis, o tie duomenys po to gali būti lengvai pasiekiami. Lenteles galima apjungti įvairiais būdais, taip išgaunant reikiamus duomenis. NoSQL duomenų bazė turi mažai bruožų būdingų SQL reliacinei sistemai. NoSQL nėra būdinga reliacinė duomenų saugojimo sistema, nėra jokių lentelių, viskas yra daug paprasčiau. Dabartinės NoSQL duomenų bazės turi keturias pagrindines struktūrines klasifikacijas: stulpelinės, rakto reikšmių (angl. key-value) , dokumentinės, grafo tipų. Tačiau egzistuoja ir maišyto modelio tipų, kuriems būdingos daugiau kaip vienos kategorijos savybės. Kiekviena klasifikacija yra unikalus modelis pritaikytas tam tikrai sričiai, kiekvienas iš jų buvo suprojektuotas taip, kad tenkintų tam tikrus dalykus ir spręstų susiklosčiusias problemas.

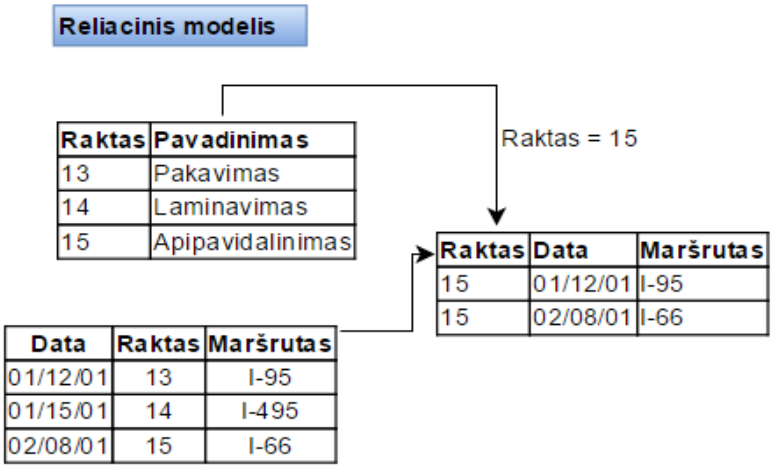
1.1. NoSQL

NoSQL apima begalę skirtingų duomenų bazių technologijų, kurios visos buvo išvystytos dėl to, kad sparčiai augo kiekis duomenų apie įvairius objektus, vartotojus ir kt., kuriuos reikėjo saugot, tuo pačiu labai dažnai tuos duomenis pasiekti, kas reikalavo didelio greičio, o tokioms technologijoms neegzistuojant tas buvo neįmanoma. Duomenų pasiekimas drastiškai sulėtindavo sistemos darbą ir sulaukdavo neigiamų atsiliepimų, todėl buvo sugalvotos nereliacinės duomenų bazės, kurios išsprendė šią keblią padėtį ir pagreitino sistemų funkcionalumą esant didžiuliam kiekiui duomenų (angl. big-data) [Tec16].

Kol NoSQL yra specifinė sistema, tuo pačiu tai yra apibūdinimas dideliam kiekiui naujų duomenų saugojimo sistemų, kurios nesinaudoja reliacinių duomenų bazės modeliu. NoSQL reiškia „ne tik SQL“, iš ko išplaukia, kad kuriant aplikaciją ar kitą produktą, yra daugiau kaip vienas duomenų saugojimo mechanizmas, kurį galimą naudoti priklausomai nuo reikalavimų. [Sad14]

1.2. NoSQL veikimo principas

Kiekviena iš sistemų kurios yra priskiriamos NoSQL dirba skirtingai, tačiau pagrindinė idėja yra pagerinti lankstumą ir efektyvumą apdorojant duomenis, naudojantis duomenų bazių modeliais kurie nepalaiko viso ACID funkcionalumo, tačiau funkcionuoja pakankamai, kad būtų naudingi. Pagrindinis dalykas kuo NoSQL skiriasi nuo SQL tai, kad duomenų bazės yra nereliacinio modelio, tai yra tarp duomenų nėra ryšių, bei abu modeliai remiasi skirtingais principais, SQL remiasi ACID (angl. atomicity, consistency, isolation, durability), o NoSQL remiasi minkštesniu ACID modeliu vadinamu BASE (angl. Basic Availability, Soft state, Eventual consistency).
Paveikslėlyje (žr. pav. 1) galima pamatyti kaip gali atrodyti reliacinis modelis.



pav. 1 Reliacinio modelio pavyzdys

Tuo tarpu, NoSQL yra daug paprastesnis modelis, saugomas JSON formatu, kuris nereikalauja jokių lentelių ar griežtos schemos. Viename įraše gali būti saugoma visa informacija (žr. pav. 2):

```
{
   _id: “studentas”,
   name: “Vardenis Pavardenis”,
   address: {
         street: “Didlaukio g. 59”
         city: “Vilnius”
         zip: “123456”
      }
}
```

pav. 2 NoSQL duomenų bazės - MongoDB saugomos informacijos pavyzdys JSON formatu

Kaip matyti pateiktame fragmente - visa informacija saugoma viename įraše, nėra nuorodų į kitą lentelę, kur būtų saugomi (tarkime) adresai, čia adresas saugomas kartu su kita informacija.

1.3. NoSQL klasifikacija

NoSQL DB (DB - duomenų bazė) modelis yra klasifikuojamas į keturis skirtingus modelius: *stulpelinių, rakto reikšmių* (angl. key-value) , *dokumentinis, grafo modelis*.

1) **Stulpelio** modelio duomenų bazėse, duomenys yra saugomi elementuose, kurie yra sugrupuoti stulpeliais, o ne kaip dažniausiai daroma – eilutėmis. Tada, šie stulpeliai yra sugrupuojami į stulpelių šeimas, kiekviena šeima gali turėti (virtualiai) begalybę stulpelių kurie gali būti sukurti veikimo metu arba schemos aprašymo metu. Nuskaitymai ir įrašymai vykdomi naudojantis stulpeliais, o ne eilutėmis. Palyginimui, dauguma RDBMS saugoja duomenis eilutėmis, šio būdo privalumas yra greita paieška/prieiga ir duomenų agregavimas. Reliacinės duomenų bazės saugoja kiekvieną eilutė kaip nepertraukiamą disko įrašą, stulpelio modelio duomenų bazės tuo tarpu, saugoja duomenis elementuose priklausančiuose stulpeliui kaip nepertraukiamą įrašą, dėl to paieška/prieiga tampa dar greitesnė [Kum16]. Vienas iš pavyzdžių galėtų būti: tam kad užklausti kelių milijonų straipsnių pavadinimus, reliacinėse duomenų bazėse, reikės praeiti kiekvieną individualų įrašą, norint išgauti pavadinimą. Stulpelio modelio duomenų bazėje, vienos užklausos į diską pagalba, galima gauti visus pavadinimus.

2) **Rakto reikšmės** modelis, tai modelis, kurio paskirtis yra saugoti, pasiekti ir tvarkyti susijusius asociatyvius elementus, taip pat šis modelis žinomas kaip *maišos* (angl. hash) arba *žodynas* (angl. dictionary). Žodynai savyje laiko kolekcijas objektų arba įrašų, kurie gali turėti didelį kiekį laukų iš kurių kiekvienas saugo savo duomenis. Šie duomenys yra saugomi ir pasiekiami rakto pakalba kuris unikaliai identifikuoja įrašą ir yra naudojamas, kad greitai galima būtų pasiekti informaciją duomenų bazėje. [LM10][Str09]

3) **Dokumentinis** modelis, kuris paveldi rakto reikšmės modelio funkcionalumą. Šie modeliai skiriasi tuo, kad rakto reikšmės modelio DB duomenys yra nesvarbu kaip paskirstyti, o dokumentinio modelio DB remiasi *dokumento* eiliškumo vidine struktūra, kad ištraukti *metaduomenys* (duomenys apie saugomus duomenis), kuriuos po to duomenų bazės variklis panaudoja tolimesnei optimizacijai. XML (angl. Extensible Markup Language) duomenų bazė yra viena iš duomenų bazių pagrįstų dokumentiniu modeliu. [Str09]

4) **Grafo** modelio duomenų bazės naudojasi grafo struktūra, kad atliktų semantines užklausas su grafo viršūnėmis, briaunomis ir savybėmis, kad po to atstovautų ir saugotų duomenis. Grafo duomenų bazė yra pagrįsta grafų teorija, kur viršūnės atstovauja objektus (žmones, laiškus, vartotojus, ar kitus dalykus), savybės, tai informacija kuri yra labai svarbi viršūnėms, pavyzdžiui: jeigu maistas yra viena iš viršūnių, tai kažkuri gali būti susieta su informacija tokia kaip: patiekalas, komponentai, skonis.... Briaunos, tai linijos, kurios jungia viršūnę su kitomis viršūnėmis arba viršūnes su savybėmis ir jos atstovauja ryšį tarp dviejų viršūnių arba viršūnių su savybėmis. Pati svarbiausia informacija yra laikoma briaunose. [Sad14][Str09]

Akivaizdu, kad kiekviena kategorija skiriasi viena nuo kitos ne tik veikimo principu, bet ir efektyvumu, Ben Scofield įvertino skirtingas NoSQL duomenų bazes (1 lentelė):

Privalumai	NoSQL (modeliai)				Reliacinė DBVS
	Stulpelio	Rakto reikšmės	Dokumentinis	Grafo	
Greitis	Aukštas	Aukštas	Aukštas	Kintantis	Kintantis
Plečiamumas (angl. scalability)	Aukštas	Aukštas	Kintantis (dažniausiai didelis)	Kintantis	Kintantis
Lankstumas	Vidutinis	Aukštas	Aukštas	Aukštas	Žemas
Sudėtingumas	Žemas	Nėra	Žemas	Didelis	Vidutinis
Funkcionalumas	Minimalus	Kintantis (nėra)	Kintantis (žemas)	Grafų teorija	Reliacinė algebra

Lentelė 1 Ben Scofield duomenų bazių įvertinimai [Sco10]

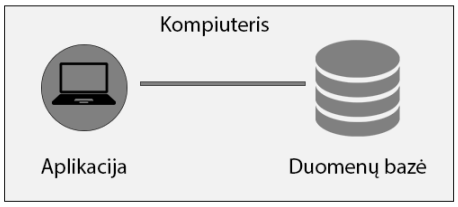
1.4. Duomenų bazės plėtimo būdai

Kuriant sistemą, kuri naudosis duomenų bazę, yra svarbu numatyti iš anksto, ko iš tos duomenų bazės reikėtų tikėtis, kadangi iš anksto nenustačius kaip bus plečiama duomenų bazė, rezultate nukentės kuriamos sistemos veikimas. NoSQL duomenų bazėms, našumas yra pagrindinis ir svarbiausias dalykas, todėl rinktis plėtimo būdą reik atsakingai, nes skirtingi plėtimo būdai skirtingai įtakoja NoSQL duomenų bazę. Renkantis duomenų bazę, būtina nustatyti kiek resursų bus skirta duomenų bazei, kokio pasiekiamumo tikimasi, kokia reikalinga sparta efektyviam sistemos veikimui ir daug kitų kriterijų. Tam, kad išvengti nuostolių, taip pat svarbu pasirinkti tinkamą plėtimo būdą, kadangi skirtingi duomenų bazės plėtimo būdai suteikia skirtingus rezultatus, vieni yra patikimesni ar spartesni už kitus. Pagrinde yra trys duomenų bazių plėtimo būdai: vieno egzemplioriaus duomenų bazė (angl. *standalone database*), replikų rinkinio duomenų bazė (angl.

replica set database) ir skaldyta duomenų bazę – klasters (angl. *sharded cluster*).

1.4.1. Vieno egzemplioriaus duomenų bazė

Vieno egzemplioriaus duomenų bazė, tai paprasčiausias variantas duomenų bazės koks tik gali būti. Tokios duomenų bazės susideda iš vieno duomenų bazės egzemplioriaus (žr. pav. 3). Šio plėtimo duomenų bazę galima įsivaizduoti kaip vieną egzempliorių, kuris atlieka viską pats, ir prie kurio jungiasi aplikacija, kai yra vykdomas kuriamos sistemos testavimas. Labai retai, šis plėtimo būdas yra naudojamas nuotoliniu būdu. Taip pat, šia duomenų baze naudotis yra paprasčiausia. [Mon17a]



pav. 3 Vieno egzemplioriaus duomenų bazė

Dažniausiai tokio tipo duomenų bazės nėra naudojamos sukurtoms sistemoms, kadangi šio tipo duomenų bazėms trūksta pasiekiamumo ir duomenų saugumo nuo praradimo. Jeigu serveris taptų nepasiekiamas, sistema momentaliai nustotų veikti. Tam, kad užtikrinti sistemos veikimą, yra naudojamos replikuotos duomenų bazės. [Mon17a] [Mon17c]

Privalumai:

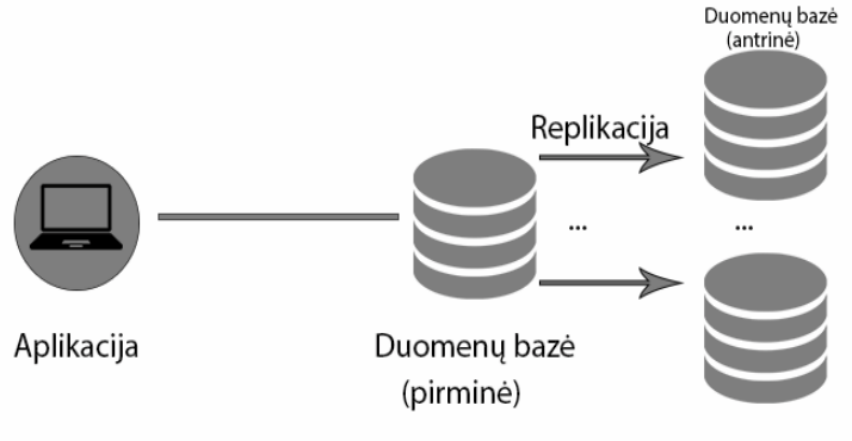
- Leidžia patogiai testuoti arba paleisti sukurtas sistemas
- Lengvas valdymas
- Reikalauja mažiausiai disko atminties ir kitų resursų, lyginant su kitais būdais

Trūkumai:

- Trūksta pasiekiamumo, egzemplioriui tapus nepasiekiamu, sistema naudojanti duomenų bazę nustos veikti.
- Visas duomenų bazės krūvis tenka vienam egzemplioriui

1.4.2. Replikuota duomenų bazė

Replikuotos duomenų bazės, toliau bus vartojama sąvoka *replikų rinkinys,* priešingai nuo vieno egzemplioriaus duomenų bazių, dažniausiai naudojamos jau sukurtoms sistemoms. Šios duomenų bazės užtikrina, duomenų pasiekiamumą - tapus nepasiekiamu bent vienam iš duomenų bazės egzempliorių, sistema veiks toliau, nes replikacijos dėka, duomenys bus pasiekiami iš pagrindinės duomenų bazės kopijos. Tuo tarpu, kol bus nepasiekiama pagrindinė duomenų bazė, iš antrinių duomenų bazių bus pasirinkta nauja pagrindinė, o kai senoji pagrindinė atsigaus, ji bus priskirta kaip antrinė. Grubiai įsivaizduoti šią duomenų bazę galima kaip parodyta paveikslėlyje [Mon17c] (žr. pav 4),



pav. 4 Replikų rinkinio duomenų bazė

Privalumai:

- Aukštas pasiekiamumas
- Užtikrinta, kad sistema veiks tol, kol bent vienas iš egzempliorių yra pasiekiamas

Trūkumai:

- Reikalauja daug daugiau disko atminties, kadangi kiekviena duomenų kopija užims tiek pat kiek originalas. Pavyzdžiui, jeigu planuojamų duomenų kiekis yra 1 TB, tai kiekviena replikacija reikalaus papildomo 1TB, kuo daugiau tokių replikų, tuo daugiau disko atminties reikia skirti.
- Vykdoma daugiau procesų, daugiau resursų yra sunaudojama

1.4.3. Skaldyta duomenų bazė

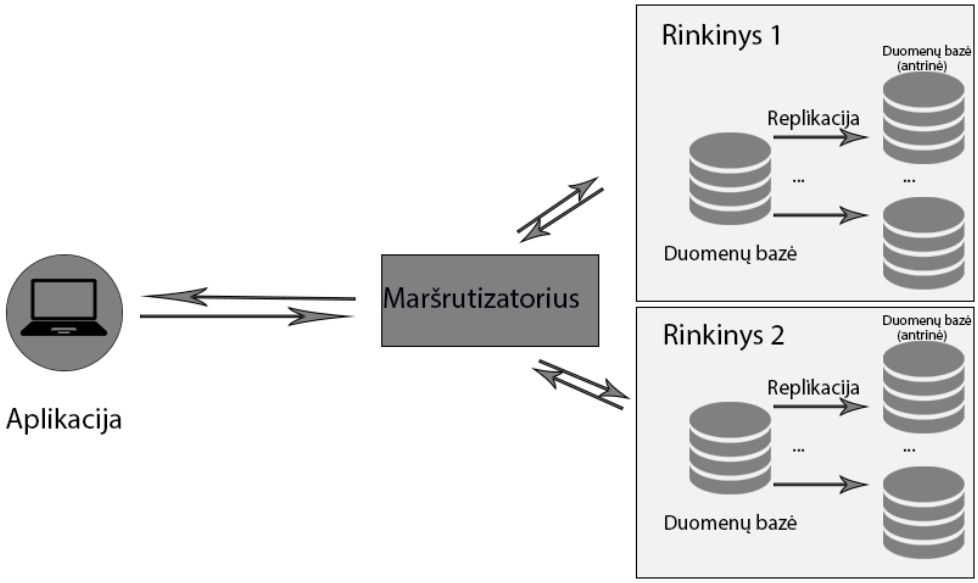
Skaldytos duomenų bazės (angl. *shard cluster*), toliau bus vartojama sąvoka *klasteris*, pasižymi tuo, kad galima turėti keletą duomenų bazės šukių (angl. *shards*) ir taip sudarant replikų rinkinių rinkinį, tuo tarpu, duomenys yra paskirstomi dalimis kiekvienai šukei.

Pavyzdžiui:

Duomenys = {A, B, C}

Egzempliorius 1 = {A} Egzempliorius 2 = {B} Egzempliorius 3 = {C}

Tokiu principu sistema gali paskirstyti sistemos krūvi skirtingoms dalims. Tas yra atliekama maršrutizatoriaus pagalba [Mon17b], kuris paskirsto darbą skirtingoms dalims, priklausomai nuo užklausos. (žr. pav 5).



pav. 5 Apibendrintas skaldytos duomenų bazės modelis su maršrutizatoriumi

Privalumai:

- Aukštas pasiekiamumas
- Net jeigu nepasiekiama taps visa duomenų bazės dalis, nepasiekiama bus tik dalis (rinkinys) duomenų, o sistema dalinai veiks.

Trūkumai:

- Kiekvienas egzempliorius reikalauja daug resursų
- Priklausomai nuo duomenų bazės, resursų kiekis gali būti dar didesnis negu planuojama plėtimui
- Dėl plėtimo charakteristikų, tam tikrais scenarijais nukenčia duomenų bazių sparta

2. Testavimo aplinka

2.1. Sisteminiai duomenys

Operacinė sistema	Ubuntu 16.04 LTS
Operatyvioji atmintis	16 GB 1600 MHz
Procesorius	Intel i7-3630QM (2,4 GHz – 3,2 GHz) (4 branduoliai, 8 loginiai procesoriai)
Atmintis	70 GB SSD (skaitymas 560MB/s, rašymas 320MB/s)

Lentelė 2 Duomenų bazei skirtos sistemos duomenys

Operacinė sistema	Ubuntu 16.04 LTS
Operatyvioji atmintis	8 GB 1600 MHz
Procesorius	Intel i7- 3635QM (2,4 GHz – 3,4 GHz) (4 branduoliai, 8 loginiai procesoriai)

Lentelė 3 Klientui skirtos sistemos duomenys

Testams atlikti yra naudojami du kompiuteriai, vienas skirtas duomenų bazei (žr. Lentelė 2), kitas testų paleidimui (žr. Lentelė 3). Taip yra daroma, kad kliento atliekami procesai neišeikvotų duomenų bazei skirtų resursų, kitokiu atveju testai, kurie intensyviai eikvoja resursus, sulėtins duomenų bazių darbą. Tam, kad testus minimaliai įtakotų jungtis tarp duomenų bazės ir „vartotojo“, testai yra atliekami lokaliai - iš vieno tinklo.

2.2. Duomenų bazės

Testams atlikti yra pasirinktos kelios populiarios NoSQL duomenų bazės: MongoDB, Redis ir CouchDB [Dbe17]. Jos yra aktualiausios, kadangi dėl jų populiarumo jos yra naudojamos daugumoje NoSQL pagrįstų aplikacijų ir tyrimų rezultatai taikytini platesniu mastu. Pasirinktos duomenų bazės taip pat dėl to, nes yra naudojamos aktualiose srityse, šios duomenų bazė turi nedaug ką bendro. Atliekant testus, tarp kiekvieno testo atlikimo duomenų bazės yra sukuriama dar kartą, o duomenys kuriuos duomenų bazė yra išsaugojus diske yra ištrinami (perkuriant duomenų bazę, duomenys automatiškai neišsitrina). Tai daroma tam, kad būtų įsitikinta, kad resursai panaudoti po praeito testo yra iš tiesų atlaisvinti (atliekant testus pastebėta, kad duomenų bazių egzemplioriams sunaudojama kiek daugiau resursų negu prieš testą). Pastebėta, kad didžiausią populiarumą turi dokumentinio ir rakto reikšmės tipo duomenų bazės, todėl būtent šios duomenų bazės ir yra pasirinktos.

2.2.1. MongoDB

Plačiausiai naudojama yra MongoDB. Tai yra dokumentinio modelio duomenų bazė, kurios duomenų perdavimo formatas yra vadinamas BSON[1], kuris gražina JSON[2] pavidalo dokumentų binarinę reprezentaciją. Šia duomenų baze naudotis yra labai paprasta, todėl ši duomenų bazė yra labai populiari tarp programuotojų ir nereikalauja duomenų bazės administratoriaus (DBA) naudojantis *bootstrap*[3] technologija. MongoDB yra funkciškai tvirta, bei suteikia versijų kontrolę, leidžiančią saugoti senesnes duomenų versijas, kad užtikrintų nuoseklumą sudėtingose transakcijose [Mon16a][Rou14].

MongoDB labiausiai tinkama scenarijams, kai reikia dirbti su didžiuliais įrašymo kiekiais ir stambiomis duomenų apimtimis (angl. *big-data*), prieinamumas turi būti maksimalus, duomenų bazė labai sparčiai augs ir reikės ją skaldyti (angl. sharding), schema nestabili ar duomenys yra priklausomi nuo vietos (angl. location based). [Rou14]

2.2.2. Redis

Redis, tai yra viena greičiausiu šiandien pasiekiamų duomenų saugyklų, tai atviro kodo duomenų bazė kuri remiasi vidine atmintimi, kurios dėka panaikinamas paieškos laikas, reikia atlikti mažiau CPU instrukcijų ir nereikia optimizuoti taip, kaip tenka optimizuoti diskinės atminties panaudojimo algoritmus. Redis yra rakto reikšmės tipo duomenų bazė. Kad pasiekti maksimalų greitį, Redis talpina visus duomenis pagrindinėje atmintyje. Kadangi, pagrindinė atmintis yra duomenų saugojimo vieta, kad duomenų bazė optimaliai veiktų reikia pasirūpinti būtent šia atmintimi, ne diskine. Diskinę atmintį Redis naudoja tik tuo atveju, kai išeikvojami pagrindinės

[1] BSON – angl. Binary JSON – MongoDB duomenų bazės, reprezentuoja JSON dokumentus užkoduotus binariniu formatu ir tai vadina BSON

[2] JSON – angl. JavaScript Object Notation – tai atviras, žmogaus ir kompiuterio perskaitomas standartas, kuris palengvina duomenų apsikeitimo procesą, tuo pačiu naudojantis XML formatu, kuris yra vienas pagrindinių formatų duomenų apsikeitimui naudojamų internete. Šis formatas palaiko pagrindinius duomenų tipus: skaičius, žodžius, loginius kintamuosius, masyvus ir maišos tipo kintamuosius.

[3] Bootstrap – viena populiariausių HTML, CSS ir JS struktūrų, naudojama reaguojančių (atvaizduoja tūrinį skirtingai kiekvieno tipo prietaisui) svetainių dizainų kūrimui.

atminties resursai. Duomenų bazę taip pat galima sukonfigūruoti, kad kas kažkiek laiko atmintyje esantys duomenys būtų perkelti į kietąjį diską, tačiau Redis yra dažniausiai naudojama kaip vidinės atminties slėptuvė, kurios pagalba greitai ir efektyviai pasiekiami duomenys, tai nėra optimaliausia duomenų bazė duomenims kaupti. Šią duomenų bazę geriausia naudoti tada, kai laikas yra problema kurią reikia išspręsti. [Red17a]

2.2.3. CouchDB

CouchDB, tai duomenų bazė, kuri pasinaudodama B+ medžiu (CouchDB B medžio algoritmo variacija), iškeičia dalį disko atminties į duomenų bazės našumą. B medžio struktūra, tai puikus variantas duomenims saugoti ir nuskaityti, kai jų yra milijonai ar milijardai [Als10]. CouchDB yra puiki vieno egzemplioriaus duomenų bazė, dauguma projektų pradeda nuo vieno egzemplioriaus, o esant reikalui nesunkiai plečiasi. Ši duomenų bazė naudojasi visur pasiekiamu HTTP protokolu ir JSON duomenų struktūra, dėl to ši duomenų bazė yra suderinama su viskuo, kas palaiko JSON ir HTTP [Apa16a].

2.3. Naudojami įrankiai

Atliekant testus yra svarbu, kad tiek duomenų bazė, tiek testai būtų teisingi. Tam užtikrinti galima naudotis įvairiais įrankiais, kurių pagalba testavimas gali būti palengvintas. Testavimui tokius įrankius galima skirstyti į dvi grupes: duomenų bazių valdymo įrankiai ir duomenų bazių testavimo įrankiai.

Duomenų bazių valdymo įrankis pagrinde yra skirtas tam, kad palengvintų vartotojo darbą kuriant, bei valdant duomenų bazę. Įrankis taip pat užtikrina, kad duomenų bazė būtų sukurta, bei valdoma teisingai, nepraleidžiant jokių reikalingų operacijų. Tokie įrankiai yra ypač naudingi skaldant duomenų bazes, todėl esant galimybei, jais vertėtų naudotis. Šių testų metu, gali būti naudojami duomenų bazių siūlomi įrankiai, duomenų bazių būsenoms stebėti.

Testavimo įrankiai yra naudingi tuo, kad vartotojui nereikia ranka rašyti testų, kurie jau galimai yra parašyti kitų. Tai sutaupo daug laiko ir pastangų, taip pat tokie įrankiai dažnai siūlo įvairią informaciją susijusią su testais. Dažniausiai naujai įdiegtoms sistemoms įvertinti yra naudojami būtent tokie įrankiai, o ne atliekami naujai parašyti testai. Duomenų bazėms diegti ir resursams paskirstyti, darbo metu naudojamas įrankis *Docker*.

2.3.1. Docker

Docker, tai aplikacija, kurios pagalba įvairių duomenų bazių servisus galima paversti konteineriais, taip valdant jiems skiriamą resursų kiekį. Taip pat Docker Hub siūlo daug įvairių jau paruoštų konteinerių, kurie palengvina duomenų bazių diegimą. Šis įrankis pasirinktas todėl, nes jo pagalba galima atskirti duomenų bazių egzempliorius ir simuliuoti paskirstytą sistemą. Kiekviena duomenų bazė dirbs su savo procesoriumi ir vidine atmintimi. Taip išvengiama konfliktų tarp skirtingų egzempliorių, kai egzemplioriams negalima išskirti atskiro loginio procesoriaus ir keliems egzemplioriams tenka dalintis vienu procesoriumi [Doc17].

Šis įrankis yra pasiekiamas tiek Windows, tiek Linux operacinėse sistemose ir yra labai patogus įrankis administruoti procesus, kurti servisus ar duomenų bazes testams atlikti. Naudotis Docker galima tiek pasinaudojant įvairiais grafinio atvaizdavimo papildiniais, tiek per terminalą. Grafinio atvaizdavimo papildiniai yra patogesni, nes jų pagalba suteikiama daug daugiau informacijos akimirksniu ir taip pat yra pateikiami resursų sąnaudų grafikai. Testų metu naudojama Portainer grafinė sąsaja. Norint paleisti duomenų bazę tereikia terminalo arba grafinės sąsajos pagalba sukurti konteinerį, kuris talpins nurodyta procesą ar jų rinkinį [Doc17].

Testams skirtoms duomenų bazėms paleisti, naudojama komanda:

docker run [--name <konteinerio pavadinimas>] --cpus <suteikiamų branduolių kiekis> --memory <operatyvios atminties kiekis> [–d] –v <kompiuterio direktorija>:<duomenų bazės direktorija> <proceso vaizdinys> <komanda vaizdiniui>

Čia <...> yra reikšmė, o [..] yra neprivalomi argumentai.

- --*name* nurodo konteinerio pavadinimą
- --*cpus* suteikia konteineriui nurodytą procesoriaus branduolių skaičių
- --*memory* suteikia konteineriui nurodytą kiekį operatyvios atminties
- -*d* leidžia konteinerį sukurti „atskirtą“ nuo vartotojo, kitaip tariant, konteineris yra startuojamas, o vartotojui nėra spausdinama į ekraną jokia konteinerio išvestis
- -*v* susieja disko atminti su konteinerio atmintimi, tai yra ypač naudinga norint matyti konteinerio turinį, o pasiekti turinį tiesiai iš konteinerio nėra galimybės. Reikia nurodyti direktoriją kur saugoti duomenis, o po dvitaškio konteinerio direktoriją su kuria susieti
- *Proceso* vaizdinys, tai iš DockerHub atsisiųstas duomenų bazės atvaizdas, galima siųstis tiek oficialius, tiek vartotojų konfigūruotus (optimizuotus) vaizdinius. Taip pat, galima nurodyti ir tam tikrą versiją.
- *Komanda vaizdiniui,* tai komanda kurią galima naudoti įprastu atveju paprasčiausiai iš terminalo, šiuo atveju tai yra duomenų bazės egzemplioriaus paleidimo komanda su argumentais

2.3.2. MongoDB įrankiai

MongoDB reikalingiems testų rezultatams gauti yra naudojami du įrankiai: Mongo-perf ir NoSQL spartos testai.

- *mongo-perf* [Mon16b] - tai įrankis kurio pagalba bus testuojama MongoDB duomenų bazė. Įrankis siūlo gausybę įvairių testų, pradedant paprastais ir baigiant sudėtingais. Šis įrankis maksimaliai apkrauna naudojamą sistemą per duomenų bazę ir taip gaunami duomenys, kurie parodo duotos duomenų bazės našumą esant naudojamai sistemai. Šio įrankio pagalba galima atlikti testus įvairaus plėtimo būdo duomenų bazėms, o po to rezultatus palyginti. Taip pat, šio įrankio dėka matuojama maksimali sparta, esant skirtingiems gijų skaičiams.

Testas paleidžiamas:

> python benchrun.py [parametrų sąrašas]

Parametrai:

-*f* <*testai*> - direktorija kur randasi testai arba pats testas

-t <gijos> - šiam parametrui nurodomi gijų kiekiai, kuriais bus atliekami testai, pavyzdžiui: 1 2 4 8 16 32 64, atliekami testai kiekvienam nurodytam gijų skaičiui.

--*host* <*adresas*> - duomenų bazės adresas (pvz. 127.0.0.1)

--*port* <*jungtis*> - jungtis per kurią bus jungiamasi prie adreso (pvz. 27000)

--*trialCount* <*bandymų kiekis*> - kaip ilgai yra atliekamas kiekvienas bandymas*

--*trialTime* <*bandymų trukmė*> - kiek kartų atliekamas kiekvienas bandymas*

* - Pasirenkamas trialCount **arba** trialTime.

Taip pat yra ir kiti parametrai, kurie veikia kaip filtrai testams arba nurodo kur nukreipti

išvedimo srautą, tačiau jie nėra aktualūs.

Šio įrankio pagalba, bus testuojamas duomenų bazės našumas. Kartu su šiuo įrankiu yra pateikiami ir testavimo failai, kuriuose yra apibrėžti testai. Testai bus atliekami geriausiu, spartos atveju, kai vienai gijai tenka vienas loginis procesorius, kadangi klientui yra galimi 8 loginiai procesoriai, tai *mongo-perf* testas bus atliekamas 8 gijų atveju. Kiekvienas šio testo testavimo failas turi tam tikrą skaičių testų, kurie atlieka skirtingus, tos pačios srities testus. Pavyzdžiui paprastų įrašymų testavimo faile yra apibrėžti testai: įterpti 10 simbolių ilgio žodžius į duomenų bazę; įterpti tuščius dokumentus į duomenų bazę; įterpti į duomenų bazę dokumentų vektorių, kur kiekvienas dokumentas turi skaičiaus reikšmės lauką ir daug kitų testų.

Tuo tarpu sunkesnis įterpimo testo variantas:

Pasiruošimas: Sukurti indeksą laukui X.

Testas: Įterpti dokumentus su numatytuoju objekto ID (angl. *default ObjectID*) ir lauku X, kuris laiko atsitiktinį skaičių, skaičius priklauso nuo gijos.

Tokie testai, kurie atlieka tam tikras procedūras prieš vykdant transakcijas, dirba su įvairiais sistemos resursais. Tam, kad tikėtis tikslesnių rezultatų, testai atliekami keletą kartų, o testas paleistas su „trialTime 180“ parametru, kurio dėka kiekvienas iš testų pakartotinai atliekamas 3 minutes, ir gražinamas rezultatų vidurkis.

- NoSQL spartos testai (angl. *NoSQL Performance Tests)* [Wei16] - šio įrankio pagalba galima testuoti duomenų bazės gebėjimą apdoroti didelį kiekį informacijos vienu momentu. Šių testų atveju, įrankis panaudotas įterpti 1 000 000 dokumentų į duomenų bazę, tam kad galima būtų išmatuoti duomenų bazės sąnaudas disko atžvilgiu.

  node benchmark <duomenų bazė> -a <adresas> -t <testai>

Testas yra atliekamas trims plėtimo būdams. Testui bus pateikta MongoDB kaip parametras, kuri duomenų bazė testuojama. Iš pradžių bus atliktas įrašymo testas, kuris patalpins 1 000 000 įrašų į duomenų bazę, po testo bus matuojama duomenų bazės apimtis disko atmintyje. Testai pakartotinai vykdomi 20 kartų, kad įsitikinti gautų rezultatų tikslumu.

2.3.3. Redis įrankiai

Kartu su Redis duomenų baze, gaunamas ir įrankis Redis Benchmarks, kurio pagalba galima atlikti pagrindinius testus, sistemos našumui testuoti. Pats naudojimas yra labai paprastas, tereikia terminale parašyti [Red17b]:

redis-benchmark [-h <IP adresas>] [-p <prievadas>] [-n <užklausų kiekis]> ...

Testams paleisti užtenka ir tokios paprastos komandos, taip pat galima:

- *-c* pagalba nurodyti klientų kiekį;
- *-p* pagalba užklausas siųsti partijomis, tokiu būdu išgaunant didesnę spartą, tačiau šiems testams, šis parametras neaktualus
- *-d* pagalba nurodyti paketų dydį

Taip pat, galimos ir kitos komandos, kurios labiau susijusios su prisijungimu, bei galimi testų parametrai. Šio testo pagalba pagrinde bus išmatuotas duomenų bazės našumas įvairiais atvejais. Rezultatų pagalba galima lengvai palyginti duomenų bazių plėtimo būdus ir atlikti duomenų analizę. Įrankio pagalba galima atlikti šiuos testus:

- PING_INLINE, PING_BULK
- SET, GET, INC
- LPUSH, RPUSH, LPOP, RPOP
- SADD, SPOP
- LRANGE_X (čia X gali būti 100, 300, 500 arba 600)
- MSET.

Visi testai atliekami atitinkamoms pagal testo pavadinimą komandoms, kur kiekviena vienaip ar kitaip apdoroja duomenis, tik dvi komandos – PING_INLINE ir PING_BULK yra skirtos testuoti tinklo pralaidumą. Prieš atliekant LRANGE_X testus, yra atliekamas LPUSH, kuris patalpina duomenis į duomenų bazę, kad LRANGE_X galėtų po to tuos duomenis skaityti.

Skirtingai nuo kitų duomenų bazių testų, atliekant testus šiai duomenų bazei disko atmintis nebus matuojama. Taip yra daroma todėl, nes ši duomenų bazė remiasi vidine atmintimi, duomenys dažniausiai yra saugomi tam, kad juos būtų galima greitai pasiekti, pavyzdžiui vartotojo prisijungimo sesijos duomenys. Į diską duomenys rašomi tuo atveju, jeigu yra nustatytas tam tikras operacijų ar laiko intervalas, kitaip tariant, yra galimybė nesaugoti duomenų diske [Red17b].

Redis Benchmarks, tai puikus įrankis norint išbandyti vartojamos Redis duomenų bazės operacijų našumą. Kadangi šiuo metu dar nėra oficialiai publikuoto gero įrankio, kuris leistų patogiai testuoti skaldytą Redis duomenų bazę, testuojant klasterio atvejį, testai buvo minimaliai pakoreguoti, kad jie dirbtų pasinaudojant klasterio funkcionalumu. Tam, kad būtų tokia galimybė, reikėjo papildyti testus raktų generavimu, kad kiekvienas testas generuotų atsitiktinį raktą iš nurodyto intervalo, kad duomenys būtų skirstomi skirtingiems duomenų bazės egzemplioriams. Tai reikėjo padaryti, nes dalis testų naudojosi bendrais raktais, dėl ko skaldytos duomenų bazės nebuvo naudojamos ir testai dirbo tik su vienu egzemplioriumi. Pavyzdžiui SET komandos modifikacija būtų – vietoj testų numatytos komandos *SET RAKTAS REIKŠMĖ* galima pasinaudoti atsitiktinio skaičiaus generavimu ir komanda modifikuoti į *SET __rand_int__ __rand_int__*, toks testų papildymas nurodomas per testo parametrą *–r <reikšmė>*. Tai padaryti padėjo Redis vartotojų modifikuotas Redis Cluster Benchmarks. Veikimo principas ir paleidimas yra iš esmės toks pat. Šis įrankis yra komandų rinkinys, kuris pasinaudodamas Redis Benchmarks ir minėta modifikacija, leidžia testus [Mai17]. Taip pat, kadangi testai buvo atliekami kai replikų rinkiniai sudaromi iš 3 egzempliorių, reikėjo priimti nestandartinį sprendimą. Kadangi Redis reikalauja nemažiau kaip 3 šukių, o tokiu atveju neužtektų branduolių visai duomenų bazei (reikėtų 9 branduolių, o daugiausia galima suteikti 8), buvo sukurtas papildomas replikų rinkinys, kuriam buvo maksimaliai apribotas resursų panaudojimas ir leidžiant testus, atsitiktinių raktų aibė buvo apribota, kad nebūtų naudojama trečioji šukė. Testas pakartotinai atliekamas 50 kartų, kad gauti tiksliausius rezultatus.

2.3.4. CouchDB įrankiai

Iron Cushion tai vienas populiariausių CouchDB testų. Šis įrankis yra licencijuotas MIT licencija ir viešai publikuotas. Iron Cushion yra skirtas atlikti dažniausias operacijas kurios atliekamos WEB aplikacijoms [Par17]. Įrankis atlieka du tipus testų:

- Didelio mąsto įterpimus (angl. *Bulk insert*) vietiniu ir nuotoliniu būdu, toliau naudojama savoka BULK.
- Įterpimo/Nuskaitymo/Atnaujinimo/Ištrynimo operacijas (angl. *CRUD operations*), toliau naudojama sąvoka CRUD.

Atliekami testai yra sukurti taip, kad galima būtų simuliuoti realaus gyvenimo įvykius.

Testas paleidžiamas naudojantis JAVA ir Netty biblioteka. Testui reikia nurodyti parametrus:

--num_connections= <skaičius> - skaičius nurodantis kiek prisijungusių klientų simuliuoti
--database_address=<http://ip_adresas:prievadas> - adresas kuriuo jungtis prie duomenų bazės
--database_name=<duomenų bazės pavadinimas> - testams naudojamos duomenų bazės pavadinimas
--json_document_schema_filename=<schemos JSON failo pavadinimas> - JSON formato failas, kuriame nurodyta schema, kuria testai remsis ir užpildys atsitiktiniais duomenimis.
--num_documents_per_bulk_insert=<skaičius> - kiekis nurodantis kiek dokumentų perduoti duomenų bazei per vieną užklausą
--num_bulk_insert_operations=<skaičius> - kiekis nurodantis kiek didelio mąsto įterpimų atlikti
--num_crud_operations=<skaičius> - kiekis nurodantis kiek CRUD operacijų atlikti
CRUD operacijų svorių parametrai:
--create_weight=<skaičius>
--read_weight=<skaičius>
--update_weight=<skaičius>
--delete_weight=<skaičius>

Šie svoriai nurodo, kuri dalis operacijų yra skiriama būtent nurodytoms CRUD operacijoms. Pavyzdžiui jeigu nurodomas įterpimo svoris – 1, tai įterpimų bus atlikta

$$\frac{1}{visų\_svorių\_suma} * crud\_operacijų\_kiekis$$

Taip pat, taikomos taisyklės svoriams:

- Įterpimo ir nuskaitymo svorių suma turi būti didesnė arba lygi trynimo svoriui
- Jeigu atnaujinimo svoris yra didesnis už 0, tai įrašymo ir nuskaitymo svorių suma taip pat turi būti didesnė už 0.
- Trynimo svoris negali viršyti ribos, kai operacijų kiekis viršija įterptų dokumentų masinio įterpimo ir CRUD įterpimo metu kiekį.

Nesilaikant taisyklių parametrai nepraeis patikrinimo, kadangi tam, kad testai galėtu teisingai dirbti, duomenų bazėje turi egzistuoti tam tikras kiekis duomenų. Kitokiu atveju, testai neturės ką atnaujinti ar ištrinti [Par17].

Iron Cushion pagalba, į CouchDB patalpinta 1 000 000 dokumentų, vykdant našumo patikros testus. Po testų gauti rezultatai yra duomenų bazės našumas, atliekant vietinį įterpimą, nuotolinį įterpimą ir CRUD operacijas. Papildomai, galima išmatuoti disko atminties ir vidutines operatyvios atminties sąnaudas atliekant testų operacijas. Disko atminties ir operatyvios atminties rezultatų šiuo atveju neįtakoja suteiktų resursų kiekis, todėl kiekvienam plėtimo būdui testas atliekamas tik vienu iš atvejų.

CouchDB duomenų bazės plėtimu šiek tiek skiriasi nuo Redis ir MongoDB – čia plėtimo būdui galima priimti įvairius architektūrinius sprendimus, pavyzdžiui nesunkiai galima nustatyti, kad visi replikų rinkinio egzemplioriai replikuotų iš visų replikų rinkinio egzempliorių. Taip padarius, duomenis patalpinus į bet kurį iš egzempliorių, su laiku likusieji egzemplioriai duomenis turės taip pat. Kadangi CouchDB sprendžia pati, kada tinkamiausias momentas replikuoti duomenis, operatyvios atminties sąnaudos matuojamos tik tada, kai yra atliekama pilna replikacija. Disko atmintis po testo matuojama pasinaudojant duomenų bazės Futon įrankiu, kuris leidžia pasiekti

duomenų bazės turinį per grafinę sąsają [Cou17]. Grafinės sąsajos pagalba galima pamatyti, kiek įrašų patalpinta ir kiek disko atminties sunaudojama. Tam, kad atlikti testus klasteriui, reikėjo sukonfigūruoti maršrutizatorių, kuris nustatytų, kuriai užklausai kuri duomenų bazė tenka. To reikėjo, nes pasinaudojant Docker ir CouchDB yra sunku teisingai sukonfigūruoti skaldytą duomenų bazę, kadangi didžioji dalis konfigūravimo atliekama per vieną egzempliorių [Cou17], bet tokiu atveju visas klasteris bus patalpintas į vieną konteinerį.

3. Plėtimo būdo pasirinkimas

Žiūrint į duomenų bazę iš resursų perspektyvos, duomenų bazė yra procesas kuriam svarbiausia:

- Disko atmintis
- Operatyvioji atmintis
- Procesorius skaičiavimams atlikti.

Šie resursai būtent ir yra svarbiausi komponentai serveriui, o jų tinkamas paskirstymas yra svarbi užduotis. Jeigu ištinka situacija, kurioje duomenų bazė reikalauja daugiau resursų, juos galima paprasčiausiai pridėti, tačiau pasiekus tam tikrą ribą (priklauso nuo turimos įrangos), resursų pridėti galimybės nėra. Todėl reikia plėstis per kelis serverius.

Duomenų bazėms, tokiu būdu yra suteikiama galimybė suteikti daugiau resursų, atsiranda galimybė saugoti daugiau duomenų ir aptarnauti dar daugiau vartotojų. Tačiau resursų pridėjimas nėra skirtas vien tik tam, kad papildyti turimų duomenų bazių resursus, papildomi resursai gali būti skirti ir tam, kad paleisti dar kelis duomenų bazės egzempliorius turimai duomenų bazei. Plėsti resursus galima įvairiai, tačiau visi variantai priveda prie vienos išvados – didelės išlaidos.

Duomenų bazių atveju, kad duomenų bazė optimaliai veiktų, svarbu užtikrinti tokį resursų kiekį, kad to užtektų. Plečiant duomenų bazę resursų poreikis yra ypač aktualus, nes dėl nurodytos priežasties resursų kiekį galimai gali tekti net padvigubinti ar padidinti dar daugiau, priklausomai nuo situacijos. Resursų koregavimas norint keisti duomenų bazės plėtimo būdą yra vienas variantas, kitas būtų priklausomai nuo resursų parinkti optimaliausią duomenų bazės plėtimo būdą. Įvertinant tai kas yra svarbu sistemai, kuri naudosis duomenų baze, duomenų bazės našumą tam tikru plėtimo būdo atveju ir resursų sąnaudas, galima nustatyti tinkamiausią plėtimo būdą duotu atveju. Duomenų bazės plėtimą galima interpretuoti keliais būdais: kaip papildomas išlaidas arba duomenų bazės našumo paaukojimą vardan duomenų pasiekiamumo saugumo ir naujų galimybių optimizuoti duomenų bazę.

Šiame darbe atliekamas tyrimas nustatyti kiek resursų reikalauja duomenų bazės plėtimas. Palyginimui reikia gauti duomenis, kurių pagalba duomenų bazių plėtimo būdai gali būti palyginami tarpusavyje. Duotu atveju kaštai, pagal kuriuos yra lyginami duomenų bazių plėtimo būdai:

- Procesoriaus branduolių skaičius skirtas visam duomenų bazės plėtimo būdui
- Sunaudotas operatyvios atminties kiekis
- Sunaudotas disko atminties kiekis, patalpinant didelį kiekį duomenų. Čia bus nagrinėjama kiek yra papildomų išlaidų patalpintiems duomenims (angl. *overhead)*
- Duomenų bazės našumas matuojamas operacijomis per sekundę

Tam, kad duomenų bazes galima būtų palyginti reikia nustatyti atvejus, kuriems bus atliekami testai. Šių testų atlikimo atveju, vienam duomenų bazės egzemplioriui yra suteikiama 2GB operatyvios atminties už kiekvieną suteiktą branduolį ir nemažiau kaip vienas branduolys, o palyginimui skirti atvejai yra nurodyti lentelėje (žr. Lentelė 4**),** čia resursai yra skirti kiekvienam duomenų bazės egzemplioriui.

Plėtimo būdas	Vieno egzemplioriaus duomenų bazė			Replikų rinkinys		Klasteris
Branduolių kiekis	1	3	6	1	2	1
Operatyvioji atmintis	2GB	6GB	12GB	2GB	4GB	2GB
Egzempliorių skaičius	1			3		6

Lentelė 4 Resursų paskirstymas skirtingiems plėtimo būdams (resursai vienam egzemplioriui)

Palyginimai gali būti vykdomi tik tarp duomenų bazių, kurioms yra suteiktas vienodas resursų kiekis. Pavyzdžiui klasteris kuriam yra skirtas 1 branduolys ir 2 GB operatyviosios atminties gali būti lyginamas tik su replikų rinkiniu, kuriam yra skirti 2 branduoliai ir 4GB operatyviosios atminties kiekvienam egzemplioriui, ir su vieno egzemplioriaus duomenų bazę kuriai yra skirti 6 branduoliai ir 12 GB RAM. Toks palyginimas yra galimas todėl, nes kiekvienam paminėtam duomenų bazės variantui, sumuojant visas resursų išlaidas, viskas susieina į vieną atvejį, kai iš viso visam duomenų bazės rinkiniui yra skirta 6 branduoliai ir 12 GB RAM. Tik lyginant duomenų bazes, kurios naudojasi vienodu resursų kiekiu, galima padaryti išvadas.

Tokiu būdu atlikus testus kiekvienam iš atvejų, galima palyginti duomenų bazių plėtimo būdus, kai replikų rinkiniai yra sudaryti iš 3 egzempliorių, o klasteris iš 2 šukių, kur kiekviena šukė yra replikų rinkinys sudarytas iš 3 egzempliorių (viso 6 egzemplioriai). Taip galima atlikti palyginimus tarp plėtimo būdų ir taip pat nustatyti rezultatus kurie gali būti gauti plečiant duomenų bazes, o ne resursus.

Vieno egzemplioriaus duomenų bazė	Replikų rinkinio duomenų bazė	Klasteris
1 branduolys, 2 GB RAM		
3 branduoliai, 6GB RAM	1 branduolys, 2GB RAM	
6 branduoliai, 12GB RAM	2 branduoliai, 6GB RAM	1 branduolys, 2GB RAM

Lentelė 5 Duomenų bazių palyginimo būdai (lyginti galima horizontaliai ir vertikaliai, bet ne įstrižai)

Tokių palyginimo būdų scenarijai galėtų būti (žr. Lentelė 5):

- Turime 1 branduolio duomenų bazę, dėl poreikio duomenų bazei reikalingi pokyčiai ir pridedami resursai, lieka nuspręsti ar likti prie to pačio plėtimo būdo, ar plėstis į kitą;
- Turime 3 branduolių duomenų bazę ir svarstoma ar plėsti resursų atžvilgiu, ar egzempliorių kiekio atžvilgiu (plėsti į replikų rinkinį)
- Turime 6 branduolius ir svarstoma plėstis į replikų rinkinį, kur suteikiama po du branduolius egzemplioriui, arba į klasteri, kur kiekvienam egzemplioriui tenka po vieną branduolį.
- Turime replikų rinkinį, kur egzemplioriui tenka 1 branduolys, yra svarstoma pridėti resursų (egzemplioriui tektų 2 branduoliai), arba iškeisti duomenų pasiekiamumą ir saugumą į vieno egzemplioriaus duomenų bazę su 3 branduoliais, manant, kad sparta patrigubės
- Turimas tam tikras biudžetas leidžiantis, reikia nuspręsti kokį plėtimo būdą rinktis, kad būtų išgauta didžiausia nauda ir sutaupyta biudžeto dalis.
- Kiti atvejai

Verta paminėti, kad gauti rezultatai galioja tik pačiam bendriausiam atvejui, kai visi operacijų tipai yra svarbūs, o duomenų bazėms nėra atlikti jokie optimizavimo darbai. Todėl, norint išgauti tiksliausius rezultatus reiktų prieš atliekant testus, duomenų bazę pritaikyti pagal poreikį ir atlikti reikalingus optimizavimo ir kitus darbus, kurie būtų atlikti jeigu būtų naudojamas plėtimo būdas.

Atliekamos duomenų užklausos yra pačio paprasčiausio varianto, kitaip tariant – nėra priimtas joks architektūrinis ar programinis sprendimas, kuris leistų išgauti geresnius rezultatus vienu ar kitu atveju. Tokių optimizacijų pavyzdžiu gali būti visų egzempliorių iš vieno replikų rinkinio panaudojimas, norint išgauti maksimalų kiekį duomenų vienu metu. Kadangi, egzempliorių skaičius replikų rinkinyje yra N tai ir duomenų nuskaitymo sparta teoriškai padidėja N kartų. Tokio

pobūdžio optimizavimai leidžia kai kurioms operacijoms suteikti daugiau našumo. Atlikus testus bendriausiu variantu, po to galima teoriškai numatyti, kokie bus išgauti rezultatai vieno ar kito plėtimo būdo atveju ir lengviau apsispręsti, ar tai būtent rezultatas kurio norima.

4. Testai

4.1. MongoDB rezultatai

4.1.1. MongoPerf rezultatai

	Vieno egz. 1 branduolys, 2 GB RAM	Vieno egz. 3 branduoliai 4 GB RAM	Vieno egz. 6 branduoliai 12 GB RAM	Replikų rinkinys 1 branduolys, 2 GB RAM per egz. (3 egz.)	Replikų rinkinys 2 branduoliai, 4 GB RAM per egz. (3 egz.)	Klasteris 1 branduolys, 2GB RAM per egz. (6 egz.)
	Operacijos per sekundę					
Sudėtingas įterpimas	9178,72	30790,54	55299,46	8126,70	14557,16	10050,67
Paprastas įterpimas	9937,82	28865,48	47551,42	8090,66	15855,12	10979,80
Paprastas atnaujinimas	7276,80	23316,21	33141,69	5858,53	11708,28	7829,78
Sudėtingas atnaujinimas	7671,34	17936,09	40724,18	5838,76	10423,71	8292,01
Paprastas pašalinimas	8514,92	15839,96	29059,96	8315,70	14139,72	9637,58
Vidutinis našumas:	8515,92	23349,66	41155,34	7246,07	13336,80	9357,97

Lentelė 6 MongoPerf rezultatai, matuojami operacijomis per sekundę

MongoDB rezultatai parodė, kad geriausiai pasirodė vieno egzemplioriaus duomenų bazė su 6 branduoliais ir 12 GB RAM. Atliekant šiuos testus, lėčiausiai operavo replikų rinkinys, kai jam suteiktas 1 branduolys ir 2 GB RAM (žr. Lentelėje 6). Replikų rinkinys turintis 1 branduolį operavo lėčiau už vieno egzemplioriaus duomenų bazę, nes replikų rinkinio atveju dalis resursų buvo panaudojama replikacijai. Replikacijos metu skirtingi egzemplioriai komunikuoja tarpusavyje ir yra perduodami duomenys.

Lyginant vieno egzemplioriaus duomenų bazes tarpusavyje (žr. Diagrama 1), akivaizdu, kad kuo daugiau suteikiama resursų, tuo geresnė išgaunama sparta. Taip pat, paprastiems atnaujinimams matomas nuosmukis 6 branduolių atveju. Tai įvyko todėl, nes buvo atliekamas didelių dokumentų įterpimas ir duomenų bazių plėtimo būdų našumas šio testo atveju mažai skyrėsi. Didžiausias resursų kiekio privalumas matomas atliekant sudėtingus įterpimus, o mažiausias atliekant pašalinimus. Vidutiniškai, vieno egzemplioriaus duomenų bazės plėtimas iš 1 iki 3 branduolių, padidina našumą 174,18%[4], našumas beveik patrigubėja. Papildomas plėtimas iki 6 branduolių, kainuoja papildomus 3 branduolius, o našumas padidėja 76,25%.

[4] Našumo pokytis X % - skaičiuojant visų testų rezultatų našumo padidėjimą ar sumažėjimą, pradiniu našumu laikomas duomenų bazės iš kurios yra plečiamą į kitą plėtimo bazę našumas. Pavyzdžiui plėtimas iš vieno egzemplioriaus duomenų bazės su 1 branduoliu į vieno egzemplioriaus su 3 branduoliais, 1 branduolio duomenų bazės našumas yra atskaitos taškas.

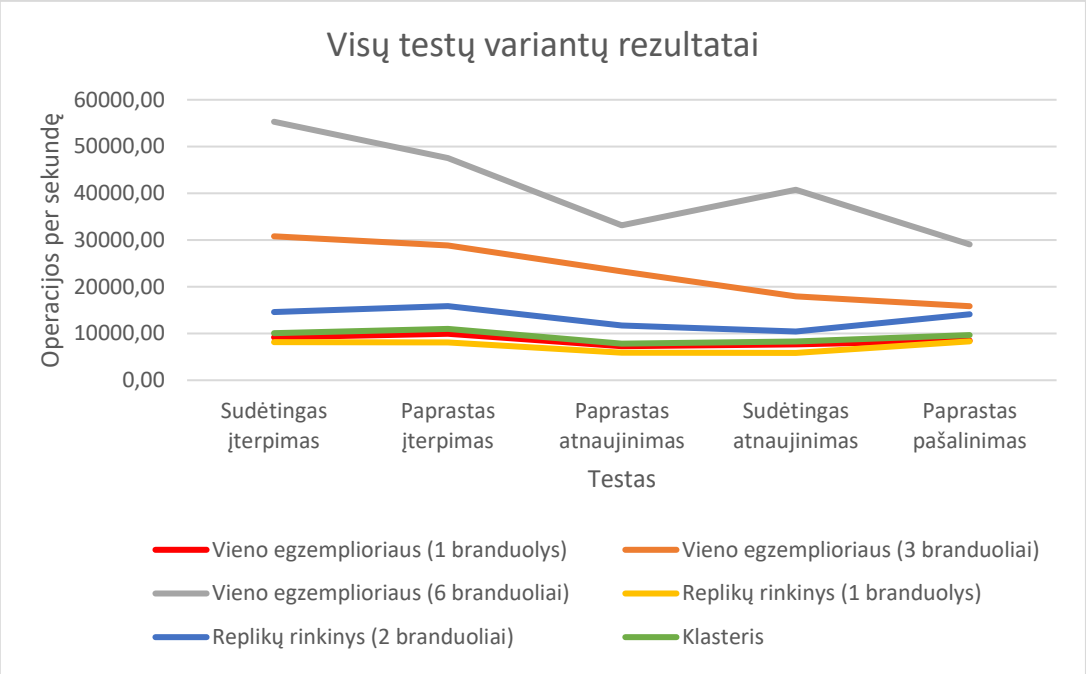


Diagrama 1 Visų MongoDB mongo-perf testų rezultatai

Lyginant atvejus, kai iš viso buvo skiriami 3 branduoliai matyti, kad vienas egzempliorius stipriausiai pasirodė įterpimo testų atveju. Likusių testų atvejais duomenų bazės operavo vis panašiau. Vieno egzemplioriaus su 3 branduoliais ir 6 GB RAM plėtimas iki replikų rinkinio, kur egzemplioriui skiriamas 1 branduolys ir 2 GB RAM, kainuoja 2 papildomus egzempliorius, o našumas sumažėjo 68,97% (lyginant su vieno egzemplioriaus). Šių rezultatų atveju, replikų rinkinio pranašumas yra tas, kad yra užtikrinamas duomenų saugumas ir pasiekiamumas.

Lyginant duomenų bazes, kai iš viso suteikiama 6 branduoliai, vieno egzemplioriaus duomenų bazė nurungia klasterį ir replikų rinkinį labai dideliu skirtumu atliekant sudėtingus įterpimus, kitų testų atvejais šiek tiek mažesniu. Plečiant vieno egzemplioriaus duomenų bazę iki replikų rinkinio kainuoja 2 papildomus branduolius, o našumas suprastėja 67,59%, o replikų rinkinį plečiant iki klasterio, už papildomus 3 branduolius, replikų rinkinio duomenų bazės našumas suprastėja 29,89%.

Plečiant duomenų bazę į daugiau egzempliorių turinčius plėtimo būdus labai nukenčia duomenų bazės našumas, bet mainais už našumą ir resursus įgyjamas duomenų pasiekiamumas, saugumas ir galimybė resursų atžvilgiu plėsti duomenų bazę daugiau. Pagal poreikius sukonfigūravus duomenų bazę, rezultatai vienu ar kitu atveju teoriškai galėtų pagerėti porą kartų.

4.1.2. NoSQL spartos testo rezultatai

Duomenų bazė	Užimta disko atmintis (duomenų katalogas)	Užimta disko atmintis (įterpti ir apdoroti duomenys)
Vieno egzemplioriaus duomenų bazė	1676,42 MB	1366 MB
Replikų rinkinys (3 egzemplioriai)	5492,22 MB	4587 MB
Klasteris (6 egzemplioriai)	6693,26 MB	4571 MB

Lentelė 7 NoSQL spartos testų rezultatai disko atminčiai

Atlikus NoSQL spartos įterpimo testą, į duomenų bazę buvo patalpintas 1 000 000 įrašų dydžio dokumentas. Patikrinus duomenų katalogus po to, kai dokumentas buvo apdorotas, duomenų bazės katalogas vieno egzemplioriaus atvejų užėmė 1676,42 megabaitus (žr. Lentelė 7), iš kurių 1366 MB buvo įterpti duomenys. Praplėtus duomenų bazę iki trijų egzempliorių replikų rinkinio, duomenų bazės katalogas padidėjo 3816,8 megabaitais, iš kurių 3221 megabaitas yra papildomai skirta atmintis įterptų duomenų saugojimui. Po to, praplėtus duomenų bazę į Klasterį, kuris buvo sudarytas iš dviejų šukių, kur kiekviena yra replikų rinkinys sudarytas iš 3 egzempliorių (viso 6 egzemplioriai, prie kurių prisideda 2 konfigūraciniai), katalogas padidėjo 1201,06 megabaitais, o atminties dydis skirtas įterptiems duomenims saugoti, beveik nepasikeitė nuo replikų rinkinio. Vidutiniškai, disko atminties dydis skirtas įterptų duomenų saugojimui išaugo 1529 megabaitais už kiekvieną duomenų bazės egzempliorių replikų rinkinio atveju. Klasterio atveju toks prieaugis skaičiuojasi trims egzemplioriams, kadangi įterpti duomenys buvo padalinti kiekvienai šukei, kitaip tariant – klasterio atveju lyginant su replikų rinkiniu, beveik nėra papildomo prieaugio kiekvienai šukei, įterptiems duomenims saugoti.

4.2. Redis rezultatai

	Vieno egzemplioriaus (1 branduolys, 2GB)	Vieno egzemplioriaus (3 branduoliai, 6GB)	Vieno egzemplioriaus (6 branduoliai, 12GB)	Replikų rinkinys (1 branduolys, 2GB)	Replikų rinkinys (2 branduoliai, 4GB)	Klasteris (1 branduolys, 2GB)
PING_INLINE	46712,24	85940,18	100949,0859	40348,07	76337,82	95838,50
PING_BULK	46003,1	85741,23	98993,1384	40998,79	75585,79	83411,66
SET	46210,01	84438,06	100208,4057	32719,66	61107,26	89220,52
GET	44981,56	85731,43	100662,4827	41147,18	77517,38	93814,30
INC	46593,25	85234,54	100859,1714	32633,53	63197,81	86343,32
LPUSH	47569,97	85399,52	100584,5958	33860,81	61369,77	89564,41
RPUSH	47782,11	88121,26	102248,8974	34166,62	58823,53	85570,60
LPOP	47292,5	84913,66	98493,1272	33644,73	61494,31	85617,90
RPOP	46696,24	84376,32	101627,1945	33514,31	61253,24	82586,43
SADD	47693,24	87583,57	101527,2531	42621,51	78007,18	94357,98
SPOP	46583,85	84499,91	100998,4716	42352,54	77529,4	88755,88
LRANGE_100	25564,99	46036,28	54038,2401	21717,88	44170,26	52705,38
LRANGE_300	25564,99	20619,27	23790,4875	11771,54	19483,68	34428,80
LRANGE_500	9418,06	15184,88	17713,2267	9156,53	14751,51	15226,06
LRANGE_600	7405,62	12179,08	14090,778	7441,66	11894,43	20217,05
MSET	46844,26	91196,5	104024,6532	22737,78	39733	65426,21

Lentelė 8 Redis Benchmarks rezultatai

Atlikus Redis Benchmarks testus, gauti rezultatai (žr. Lentelė 8) parodė, kad geriausi rezultatai yra gauti vieno egzemplioriaus su 6 branduoliais atveju. Mažiausiu skirtumu visos duomenų bazės pasirodė atliekant LRANGE_300/500/600 testus.

Iš testų rezultatų matyti, kad geriausiai pasirodė vieno egzemplioriaus su 6 branduoliais duomenų bazė, tačiau lyginant su vieno egzemplioriaus duomenų baze, kai buvo suteikti 3 branduoliai, skirtumas nėra labai didelis, lyginant šiuos du atvejus rezultatai skiriasi tik 17%. Nors resursų yra suteikta net dvigubai daugiau, dėl kažkokios priežasties nėra matomas spartos padvigubėjimas.

Tam gali būti trys priežastys:

1) Klientas nebegalėjo išgauti daugiau užklausų, nes sunaudoti visi kliento resursai, todėl neparodytas tikras duomenų bazės potencialas
2) Duomenų bazė pasiekė ribą kai jau nebegali greičiau veikti, nes reikalingi optimizacijos darbai
3) Duomenų bazė išeikvojo visus resursus, tačiau vis tiek nejaučiamas didelis prieaugis

Tam, kad tai išsiaiškinti buvo atlikti papildomi testai ir stebimas resursų sunaudojimas. Rezultate pamatyta, kad duomenų bazė sunaudodavo daugiau kaip 90% suteiktų procesoriaus resursų, tačiau nepasiekdavo 100%, todėl galimai reikalingi tam tikri optimizavimo darbai, bet sprendžiant iš resursų likučio – prieaugis nebūtų didelis. Iš kliento pusės jokių keblumų nepastebėta, kaip tik atvirkščiai, klientas gebėjo maksimaliai išeikvoti duomenų bazę. Pastebėta, kad kai buvo gaunami blogesni rezultatai tam tikriems testams, klientas sisteminių resursų pilnai nesunaudodavo. Turimu atveju, galima daryti išvadą, kad vieno egzemplioriaus duomenų bazei užtektų mažiau resursų negu jai maksimaliai suteikta, todėl būtų geras sprendimas apie 30% resursų panaudoti replikacijai, sukuriant papildomą egzempliorių, kuriam yra suteikta nedaug resursų.

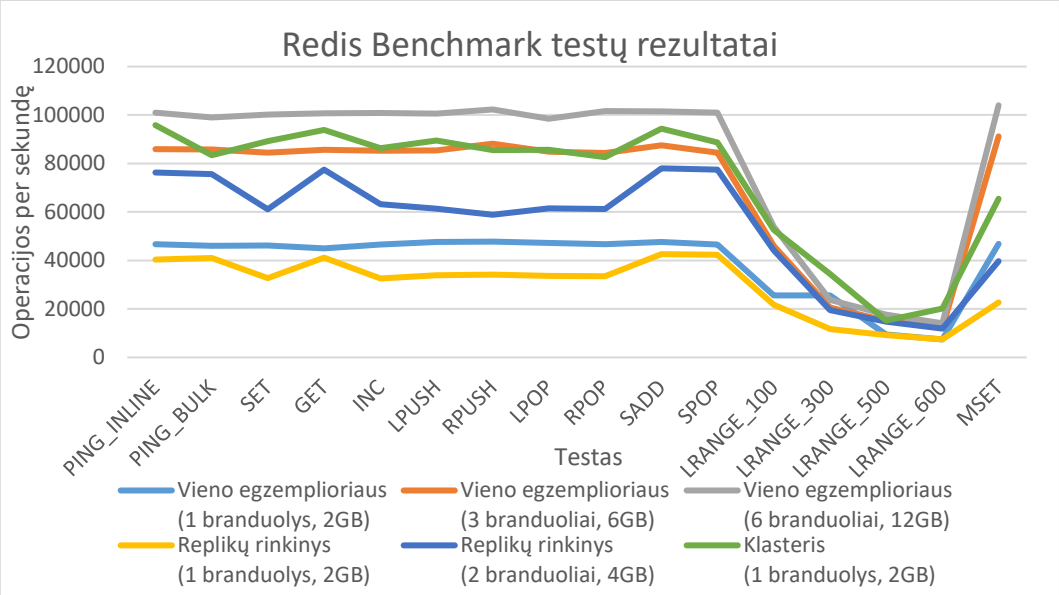


Diagrama 2 Redis Benchmarks rezultatų lentelė, vaizduojanti testų rezultatus kiekvienai duomenų bazei

Diagramoje 2 pateiktas grafikas, kuriame vizualiai parodytos vidutinės duomenų bazių spartos atliekant atitinkamus testus. Pastebėta, kad atliekant LRANGE operacijas, visų duomenų bazių našumas staigiai nukrito ir visais atvejais duomenų bazės operavo panašiu našumu. Todėl jeigu labiausiai reikalinga šių operacijų sparta, duomenų bazėms egzemplioriams nebūtina suteikti daug resursų, nes kaip parodė vieno egzemplioriaus duomenų bazės su 6 branduoliais atvejis – našumas nedaug kuo skiriasi nuo replikų rinkinio, kur vienam egzemplioriui tenka tik 2 branduoliai. Todėl šiuo atveju labiau apsimoka turėti duomenų bazės egzempliorių, kurie užtikrintų duomenų pasiekiamumą, mainais vietoj duomenų bazės kuri yra labai sparti visų likusių testų atvejų.

Stebint testų veikimo metu sunaudojamą operatyviosios atminties kiekį, rezultatai taip pat parodė, kad didėjant duomenų bazių našumui, didėja ir sunaudotos operatyviosios atminties kiekis atliekant testus. Gautų operatyvios atminties sąnaudų[5] pagalba galima pamatyti, kiek didėjantis našumas didina operatyvios atminties sąnaudas. Vieno egzemplioriaus su 1 branduoliu ir 2 GB RAM duomenų bazės plėtimas iki vieno egzemplioriaus su 3 branduoliais ir 6 GB RAM, sąnaudas padidino 13,93%, o tolimesnis plėtimas iki vieno egzemplioriaus su 6 branduoliais ir 12 GB RAM,

[5] Sąnaudos – duomenų bazės panaudotas operatyviosios arba disko atminties kiekis.

jas padidino dar 19,52%. Vieno egzemplioriaus su 3 branduoliais ir 6 GB RAM duomenų bazės plėtimas iki replikų rinkinio su 1 branduoliu ir 2 GB RAM plėtimas sąnaudas padidino 35%. Plečiant vieno egzemplioriaus su 6 branduoliais ir 12 GB RAM duomenų bazę iki replikų rinkinio, kur egzemplioriui tenka 2 GB ir 4 GB RAM, operatyvios atminties sąnaudos padidėjo 35,33%, o tolimesnis plėtimas iki klasterio sąnaudas padidino net 18,8%. Iš viso, plečiant nuo vieno egzemplioriaus su 6 branduoliais duomenų bazės iki klasterio, sąnaudos padidėjo labai nedaug, todėl plėtimo metu suteiktas resursų kiekis galėjo būti ir mažesnis.

Replikų rinkiniuose ir klasteryje pastebėta, kad antriniai egzemplioriai duomenų bazės metu dirbo labai nedideliu pajėgumu, augant duomenų bazės resursams atitinkamai padidėjo ir pajėgumas. Tokie dideli sąnaudų pokyčiai plečiant į duomenų bazes su daugiau egzempliorių yra matomi todėl, nes papildomų egzempliorių atsiradimas, padidino ne tik sumines duomenų bazės sąnaudas, bet ir pirminio egzemplioriaus, iš kurio vykdoma replikacija.

4.3. CouchDB rezultatai

Duomenų bazė	Testas (rezultatai pateikti dokumentais per sekundę)					
	Nuotolinis įterpimas	Vietinis įterpimas	Įterpimas	Nuskaitymas	Atnaujinimas	Ištrynimas
Vieno egzemplioriaus 1 branduolys, 2GB RAM	6205,69	6189,50	432,14	1,046,18	414,40	393,91
Vieno egzemplioriaus 3 branduoliai, 6GB RAM	14923,26	14874,38	1204,81	2134,75	1196,48	1171,81
Vieno egzemplioriaus 6 branduoliai, 12GB RAM	19932,28	19877,96	1588,09	2410,89	1400,36	1569,04
Replikų rinkinys 1 branduolys, 2GB RAM per egzempliorių (3 egz.)	4857,51	4852,06	318,57	812,62	314,09	278,03
Replikų rinkinys 2 branduoliai, 4GB RAM per egzempliorių (3 egz.)	11115,78	11113,66	686,09	1114,850	650,243	512,27
Klasteris 1 branduolys, 2GB RAM per egzempliorių (6 egz.)	6850,07	6201,10	489,92	1012,82	551,54	472,01

Lentelė 9 Iron Cushion testų rezultatai

Iron Cushion atlikti testai parodė, kad geriausiai šį testą įveikė vieno egzemplioriaus duomenų bazė, esant daugiausiai resursų. Blogiausiai pasirodė replikų rinkinys, esant 3 branduoliams (žr. Lentelė 9). Matuojant sunaudotos disko atminties kiekį 1 000 000 resursų patalpinti, nustatyta, kad CouchDB reikalauja pakankamai daug papildomos disko atminties (žr. Lentelė 10).

	Pirminis egz.	Antrinis egz. 1	Antrinis egz. 2
Disko atmintis (MB)	654,3	672,6	672,3
Operatyvioji atmintis (MB)	389	179	181

Lentelė 10 Vidutinės egzempliorių disko atminties ir operatyviosios atminties sąnaudos

Atlikus CouchDB testus pastebėta, kad visų plėtimo būdų atvejais testų rezultatų tendencija yra labai panaši (žr. Diagrama 3). Visais atvejais, duomenų bazės geriausius rezultatus parodė atliekant nuskaitymo testą.

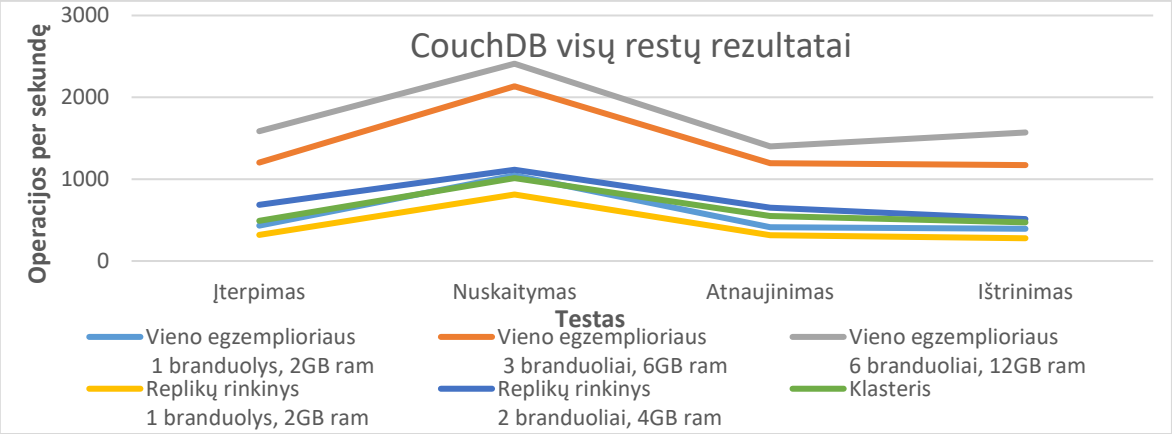


Diagrama 3 Iron Cushion testų rezultatų palyginimo diagrama

Nustatyta, kad geriausius rezultatus pasiekė vieno egzemplioriaus duomenų bazė su 6 branduoliais ir 12 GB RAM, o blogiausiai replikų rinkinys su 1 branduoliu ir 2 GB RAM. Vidutiniai našumo kaštai plečiant duomenų bazes (žr. Lentelė 11):

Plėtimo būdas	Egzempliorių kiekis	Našumo pagerėjimas (%)[6]	
		BULK[7]	CRUD[8]
Vieno egz. su 1 branduoliu į vieno egz. su 3 branduoliais	1	140,40 %	149,62 %
Vieno egz. su 3 branduoliais į vieno egz. su 6 branduoliais	1	33,60 %	22,08 %
Vieno egz. su 3 branduoliais į replikų rinkinį su 1 branduoliu	1	-67,41 %	-69,81 %
Replikų rinkinys su 1 branduoliu per egz. į replikų rinkinį su 2 branduoliais per egz.	3	128,92 %	71,96 %
Vieno egzemplioriaus su 6 branduoliais į replikų rinkinį su 2 branduoliais per egz.	3	-65,59 %	-63,75 %
Replikų rinkinys su 2 branduoliais per egz. į klasterį su 1 branduoliu per egz.	6	-38,36 %	-14,75 %

Lentelė 11 Duomenų bazės našumo pagerėjimas atliekant plėtimą, matuojamas procentais.

Iš rezultatų matyti, kad geriausias pagerėjimas matomas plečiant CouchDB duomenų bazę iš vieno egzemplioriaus su 1 branduoliu į vieno egzemplioriaus su 3 branduoliais ir kai plečiami replikų rinkinio resursai. Net daugiau kaip 120% pagerėjo duomenų bazių BULK operacijų našumas, o CRUD operacijų našumas vieno egzemplioriaus duomenų bazei pagerėjo net 149,62%. Vieno egzemplioriaus su 3 branduoliais plėtimas į vieno egzemplioriaus su 6 branduoliais duomenų bazę, našumą pagerino apie 30%. Vieno egzemplioriaus su 3 branduoliais plėtimas į replikų rinkinį su 1 branduoliu per egzempliorių ir vieno egzemplioriaus su 6 branduoliais plėtimas į replikų rinkinį, kur kiekvienam egzemplioriui tenka 2 branduoliai, duomenų bazės našumą sumažino 63-70% visų testų atveju.

Plėtimo būdas	Disko atmintis	Operatyvioji atmintis
Vieno egzemplioriaus	218,1 MB	389 MB
Replikų rinkinys	666,4 MB	749 MB
Klasteris	1394,4 MB	1396 MB

Lentelė 12 CouchDB disko atminties ir operatyvios atminties sąnaudos testams atlikti.

[6] Našumo pokytis – duomenų bazės našumo pagerėjimas ar pablogėjimas yra skaičiuojamas, kai duomenų bazės iš kurios atliekamas plėtimas našumas yra atskaitos taškas.
[7] BULK – testai kurių metu buvo masiškai talpinami duomenys.
[8] CRUD – duomenų įrašymo, nuskaitymo, atnaujinimo ir trynimo operacijų testai.

Taip pat, buvo nustatyta, kad duomenų bazė disko atminties sunaudoja net iki 42% daugiau negu įterpiama duomenų. Tai galima buvo nustatyti padauginant vidutinį įrašo dydį iš įterptų dokumentų skaičiaus. Testų metu vidutinis įrašo dydis buvo 400 baitų, įterpus 1 000 000 įrašų, duomenų bazėje faktiškai turėtų būti kiek daugiau negu 380 MB, tačiau nustatyta (žr. Lentelė 12), kad duomenų bazėje vieno egzemplioriaus atveju buvo 654,3 MB. Antriniai egzemplioriai replikų rinkiniui ir klasteriui papildomai pridėjo vidutiniškai po 672,45 MB. Toks disko atminties sąnaudų padidėjimas matomas, nes CouchDB iškeičia disko atmintį į papildomą našumą [Als10]. Tam, kad sutaupyti disko atminties, CouchDB siūlo įvairius duomenų kompresijos algoritmus. Taip pat, pastebėta, kad replikų rinkinyje ir klasteryje antriniams egzemplioriams sąnaudos po replikacijos yra maždaug 3% didesnės. Atliekant testus taip pat pastebėta, kad vidutiniškai, pirminis egzempliorius sunaudodavo 390 MB operatyvios atminties, o antriniai po 180 MB.

5. Rezultatai

Norint palyginti iš testų gautus duomenis, juos reikia supaprastinti, paskaičiuojant individualių duomenų bazės egzempliorių resursų sąnaudas ir kiek našumo tos resursų sąnaudos suteikia. Supaprastinus duomenis galima sudaryti lenteles, kurių pagalba plėtimo būdai gali būti palyginami tarpusavyje skirtingomis perspektyvomis. Šiuo atveju nagrinėjamos perspektyvos:

- Kokia dalis našumo tenka vienam branduoliui priklausomai nuo plėtimo būdo vienam duomenų bazės egzemplioriui. Skaičiuojant našumą priklausomai nuo egzempliorių skaičiaus ir nuo branduolių skaičiaus, galima gauti apytikslius duomenis, kokį našumą suteikia branduolys vienam ar kitam plėtimo būdui ir galima nustatyti tendenciją kaip kinta branduolių naudingumas (branduolio suteikiama nauda duomenų bazei, sakant, kad sumažėjo naudingumas yra sakoma, kad branduolys yra mažiau naudingas duomenų bazei, nes suteikia X% mažiau našumo).
- Kokie pokyčiai vyksta disko ir operatyviosios atminties sąnaudoms, didinant egzempliorių kiekį skirtingiems plėtimo būdams. Skirtingi plėtimo būdai elgiasi skirtingai su duomenimis, gali atsirasti įvairus konfigūraciniai ir papildomi duomenys, kurie yra reikalingi duomenų bazei, duomenys gali būti performatuoti ir saugomi kitokiu negu pateikta būdu, ir kitokie atvejai. Tokiais atvejais kinta ir duomenų bazės apimtis diske. Operatyvioji atmintis ne išimtis, daugėjant egzempliorių atsiranda daugiau komunikacijos tarp jų, o nuo to kinta ne tik procesoriaus panaudojimas bet ir operatyviosios atminties sąnaudos.

Normalizuojant gautus rezultatus galima įvesti metriką, pagal kurią galima palyginti ne tik vienos duomenų bazės plėtimo būdus, bet ir visų testuotų duomenų bazių gautus rezultatus. Tą galima padaryti nustatant atskaitos tašką, jam suteikiant įvestos metrikos pradinį vienetą. Šiuo atveju atskaitos taškas yra vieno egzemplioriaus duomenų bazė su 1 branduoliu ir 2 GB RAM, įvesta metrika yra vienam iš duomenų bazės tiriamų resursų (našumas, branduolių skaičius, operatyvioji atmintis, disko atmintis) nustatytas pokyčio santykinis vienetas vykdant duomenų bazės plėtimą. Pavyzdžiui, jeigu plečiant vieno egzemplioriaus duomenų bazę su 1 branduoliu, našumas pagerėjo 50%, tai pagerėjimo santykis būtų 1,5. Sudarius santykinę lentelę duomenų bazes galima palyginti.

5.1. MongoDB

- Operacijos per sekundę (našumas)

Plėtimo būdas	Operacijų skaičius per sekundę	Egzempliorių kiekis	Operatyviosios atminties kiekis egzemplioriui	Branduolių kiekis egzemplioriui
Vieno egzemplioriaus	8515,92	1	2 GB	1
	7783,22	1	6 GB	3
	6859,22	1	12 GB	6

Replikų rinkinys	2415,36	3	2 GB	1
	2222,80	3	4 GB	2
Klasteris	1559,66	6	2 GB	1

Lentelė 13 Našumas tenkantis vienam branduoliui, priklausomai nuo plėtimo būdo

Išanalizavus gautus testų rezultatus ir sudarius našumo lentelę (žr. Lentelė 13) nustatyta, kad vieno egzemplioriaus ir kitais atvejais, kuo daugiau branduolių yra suteikiama, tuo labiau mažėja jų efektyvumas. Vieno egzemplioriaus duomenų bazei suteikus 3 branduolius vietoj 1, branduolio naudingumas sumažėjo 9,41% papildomai pridėjus dar 3 branduolius, naudingumas sumažėjo 13,47%. Vieno egzemplioriaus duomenų bazę su 3 branduoliais praplėtus iki replikų rinkinio, kur kiekvienam egzemplioriui tenka 1 branduolys, naudingumas sumažėjo 68,96%. Vieno egzemplioriaus su 6 branduoliais plėtimas iki replikų rinkinio, kur kiekvienam egzemplioriui tenka 2 branduoliai, naudingumą sumažino net 77,26%. Tolimesnis plėtimas iki klasterio naudingumą sumažina dar 29,83%. Gauti rezultatai rodo, kad didinant egzempliorių skaičių, branduolio vertė mažėja labai įvairiai.

- Operatyvioji atmintis

Plėtimo būdas	Egzempliorių sk.	Branduolių sk. (egzemplioriui)	Operatyvios atminties kiekis (egzemplioriui)	Pirminis egz.[9]	Antrinis egz. 1	Antrinis egz. 2
Vieno egzemplioriaus	1	1	2 GB	188,6 MB	-	-
	1	3	6 GB	440,9 MB	-	-
	1	6	12 GB	805,8 MB	-	-
Replikų rinkinys	3	1	2 GB	131,1 MB	122,2 MB	116,5 MB
	3	2	4 GB	197,5 MB	181,9 MB	188,3 MB
Klasteris	6	1	2 GB	146,7 MB	127,4 MB	126,2 MB

Lentelė 14 MongoDB operatyvios atminties sąnaudos testams atlikti

Operatyviosios atminties rezultatai (žr. Lentelė 14) parodė, kad MongoDB duomenų bazės atveju, kuo didesnis našumas, tuo daugiau operatyviosios atminties duomenų bazė sunaudoja atlikdama vienodas operacijas. Net testams pabaigus darbą, operatyviosios atminties sąnaudos nesikeitė. Vieno egzemplioriaus duomenų bazės atveju, resursų pridėjimas, operatyviosios atminties sąnaudas vidutiniškai padvigubindavo (visais atvejais). Vidutiniškai, vieno branduolio pridėjimas operatyviosios atminties sąnaudas padidindavo 54%. Iš vieno egzemplioriaus duomenų bazės plečiant į replikų rinkinį, sąnaudos padidėdavo 20-30% visai duomenų bazei. Replikų rinkinio plėtimas į klasterį, sąnaudas padidino 41,37%.

- Disko atmintis

Plėtimo būdas	Užimta disko atmintis duomenų katalogui (egzemplioriui)	Užimta disko atmintis įterptiems duomenims (egzemplioriui)
Vieno egzemplioriaus	1676,42 MB	1366 MB
Replikų rinkinys (3 egzemplioriai)	1830,74 MB	1529 MB
Klasteris (6 egzemplioriai)	1115,54 MB	761,83 MB

Lentelė 15 NoSQL spartos testų sunaudotos disko atminties rezultatai

Analizuojant disko atminties sąnaudas, galima pamatyti (žr. Lentelė 15), kad vieno egzemplioriaus duomenų bazės plėtimas į replikų rinkinį disko atminties sąnaudas padidina apie 10% kiekvienam egzemplioriui. Replikų rinkinio plėtimas į klasterį (kitaip tariant suskaldymas) iš pirmo žvilgsnio sąnaudas sumažino 64%. Tačiau, prisimenant, kad duomenys yra padalinti į dvi

[9] egz. – trumpinys žodžiui „egzempliorius“. Ta pati reikšmė vartojama ir tolimesnė lentelėse.

dalis, galima nesunkiai suprasti, kad iš tikro sąnaudos padidėja, įterpiant tuos pačius duomenis, net 21,87%, lyginant replikų rinkinį ir susumavus klasterio šukių sąnaudas į vieną replikų rinkinį. Bendrai paėmus rezultatus, vidutiniškai vienas egzempliorius padidina sąnaudas 3,5%.

Bendrai paėmus visus rezultatus į visumą, galima manyti, kad replikų rinkinys, kur egzemplioriui tenka 2 branduoliai ir 4 GB RAM, šiuo atveju yra optimaliausias variantas. Taip yra, kadangi testas yra atliktas ant kompiuterio, kuriam nebėra vietos naujiems resursams ir vienintelis dalykas ką galima būtų praplėsti tai disko atmintis. Tokiu atveju, kad nereikėtų veltui pridėti disko atminties, pasirenkant variantą kuris suteikia didžiausią galimą našumą, pakenčiamas disko atminties sąnaudas ir vidutines operatyviosios atminties sąnaudas, minėtas pasirinkimas yra optimalus. Šis variantas yra geresnis našumu negu mažesnį resursų kiekį turintis replikų rinkinio variantas, užtikrina pasiekiamumą ir duomenų saugumą, disko atminties sąnaudos yra 18% mažesnės negu klasterio ir operatyviosios atminties sąnaudos yra ganėtinai nedidelės. Nors kompiuteriui netikėtai išsijungus bet kokiu atveju nutrūks ryšis su duomenų baze, replikų rinkinys leidžia apsidrausti nuo nustojusių veikti procesų, kas labiau tikėtina, negu el. energijos praradimas.

5.2. Redis

- Operacijos per sekundę (našumas)

Plėtimo būdas	Operacijų skaičius per sekundę	Egzempliorių kiekis	Operatyviosios atminties kiekis egzemplioriui	Branduolių kiekis egzemplioriui
Vieno egzemplioriaus	39307,25	1	2 GB	1
	23483,24	1	6 GB	3
	13758,43	1	12 GB	6
Replikų rinkinys	10017,36	3	2 GB	1
	9190,171	3	4 GB	2
Klasteris	12115,47	6	2 GB	1

Lentelė 16 Redis Benchmarks testų rezultatai

Sudarius Redis našumo lentelę (žr. Lentelė 16) galima įžvelgti, kad beveik visų duomenų bazių atvejais vieno branduolio naudingumas mažėja. Pridedant resursų vieno egzemplioriaus duomenų bazei iš pradžių našumas sumažėjo 40,28%, o pridėjus dar 3 branduolius ir 6 GB RAM, sumažėjo dar 41,41%. Plėtimas iš vieno egzemplioriaus su 3 branduoliais ir 6 GB RAM į replikų rinkinį, kur kiekvienam egzemplioriui tenka po branduolį ir 2 GB RAM, naudingumą pablogina 57,34%. O plėtimas iš vieno egzemplioriaus su 6 branduoliais ir 12 GB RAM į replikų rinkinį, kur kiekvienam egzemplioriui tenka 2 branduoliai ir 4 GB RAM, naudingumą sumažino 33,2%. Pastarojo replikų rinkinio plėtimas į klasterį, priešingai nuo kitų atvejų, naudingumo pagerino net 31,83%. Galima daryti išvadą, kad plėtimas arba resursų pridėjimas, visų atvejų išskyrus klasterio, duomenų bazės našumą pablogina vidutiniškai 43%. Klasterio atveju, kadangi buvo du pirminiai egzemplioriai ir krūvis buvo paskirstomas per abu, naudingumas šiek tiek padidėjo.

- Operatyviosios atminties sunaudojimas

Plėtimo būdas ir resursų kiekis vienam egzemplioriui	Egzempliorius			Egzempliorių kiekis
	Pirminis	Antrinis 1	Antrinis 2	
Vieno egzemplioriaus su 1 branduoliu ir 2 GB RAM	100,2 MB			1 pirminis
Vieno egzemplioriaus su 3 branduoliais ir 6 GB RAM	114,1 MB			1 pirminis
Vieno egzemplioriaus su 6 branduoliais ir 12 GB RAM	136,4 MB			1 pirminis
Replikų rinkinys su 1 branduoliu ir 2 GB RAM egzemplioriui	96,8 MB	21,3 MB	22,9 MB	1 pirminis ir 2 antriniai

Replikų rinkinys su 2 branduoliais ir 6 GB RAM egzemplioriui	120,6 MB	39,4 MB	24,6 MB	1 pirminis ir 2 antriniai
Klasteris su 1 branduoliu ir 2 GB RAM egzemplioriui	73,4 MB	22,1 MB	14,1 MB	2 pirminiai ir 4 antriniai

Lentelė 17 Operatyviosios atminties sunaudojimas tenkantis vienam Redis egzemplioriui

Užsirašant tikslias operatyviosios atminties sąnaudas ir imant jų vidurkį galima sudaryti operatyvios atminties sąnaudų lentelę (žr. Lentelė 17). Lyginant su kitomis duomenų bazėmis, Redis operatyviosios atminties sąnaudos skirtingais atvejais daug nesiskiria. Vieno egzemplioriaus duomenų bazei pridedant resursų, vidutiniškai operatyviosios atminties sąnaudos padidėdavo 16-17%. Replikų rinkiniui pridėjus resursų, priešingai negu vieno egzemplioriaus duomenų bazės atveju, abiejų atvejų plėtimas į replikų rinkinius, sąnaudas padidino kiek daugiau negu 35%, net dvigubai daugiau negu vieno egzemplioriaus atvejų prieaugis. Plečiant iš replikų rinkinio į klasterį buvo nustatytas 18,8% padidėjimas. Iš rezultatų yra matyti, kad Redis labai efektyviai naudojasi operatyviąją atmintimi.

Tokius rezultatus nusako vien tai, kad Redis yra vidinės atminties duomenų bazė. Skirtingai nuo disko atminties, kompiuteryje operatyviosios atminties gali būti daug kartų mažiau. Todėl Redis duomenų bazėms efektyviai saugoti duomenis padeda įvairios duomenų tipų optimizacijos, kurių dėka yra sutaupoma nemažai atminties. Redis siūlo galybę funkcijų, kurios leidžia maksimaliai taupyti atmintį, todėl Redis ir yra viena populiariausių NoSQL duomenų bazių.

Duotu atveju, plėtimo būdo pasirinkimas gali būti labai įvairus, priklausomai nuo poreikių, kadangi Redis operatyvios atminties sąnaudos yra labai mažos, tai reikėtų orientuotis į tai kas svarbiau – našumas ar duomenų saugumas. Jeigu našumas, tai vieno egzemplioriaus su 3 branduoliais ir 6 GB yra optimaliausias variantas, kadangi lyginant su 6 branduolių ir 12 GB RAM atveju, skirtumas labai nedidelis ir minėto atvejo pasirinkimas yra resursų švaistymas.

5.3. CouchDB

- Operacijos per sekundę (našumas)

Plėtimo būdas	BULK op. /sek.	CRUD op. / sek.	Egzempliorių skaičius	Branduolių kiekis	Operatyviosios atminties kiekis
Vieno egzemplioriaus	6197,60	571,66	1	1	2 GB
	4966,27	475,65	1	3	6 GB
	3317,52	290,35	1	6	12 GB
Replikų rinkinys	1618,26	143,61	3	1	2 GB
	1852,28	123,48	3	2	4 GB
Klasteris	1141,68	105,26	6	1	2 GB

Lentelė 18 CouchDB Iron Cushion testų rezultatai.

Iron Cushion testai parodė (žr. Lentelė 18), kad plečiant iš vieno egzemplioriaus su 1 branduoliu ir 2GB RAM į vieno egzemplioriaus su 3 branduoliais ir 6 GB RAM, branduolio naudingumas suprastėjo 24,79%, toliau plečiant į vieno egzemplioriaus su 6 branduoliais ir 12 GB RAM, naudingumas suprastėjo net 49,69%. Replikų rinkinio atveju, resursų pridėjimas priešingai nuo vieno egzemplioriaus, naudingumą padidino 14,46%. Vieno egzemplioriaus su 3 branduoliais ir 6 GB RAM plėtimas į replikų rinkinį, kur egzemplioriui tenka 1 branduolys ir 2 GB RAM, naudingumą sumažino net 206%. Vieno egzemplioriaus su 6 branduoliais ir 12 GB RAM plėtimas į replikų rinkinį naudingumą sumažino 55,83%. Replikų rinkinio, kur egzemplioriui tenka 2 branduoliai ir 4 GB RAM plėtimas į klasterį naudingumą sumažino net 62,24%.

- Operatyviosios atminties ir disko atminties sunaudojimas

Plėtimo būdas	Disko atmintis	Operatyvioji atmintis	Egzempliorių kiekis
Vieno egzemplioriaus	218,1 MB	389 MB	1
Replikų rinkinys	222,1 MB	249,7 MB	3
Klasteris	232,4 MB	232,7 MB	6

Lentelė 19 Vienam duomenų bazės egzemplioriui tenkančios disko ir operatyviosios atminties sąnaudos.

Priešingai nuo kitų duomenų bazių (žr. Lentelė 19), operatyviosios atminties sunaudojimas kiekvienam egzemplioriui ne didėjo, o mažėjo, kuo daugiau egzempliorių buvo duomenų bazėje, tuo mažiau operatyviosios atminties kiekvienas sunaudodavo. Disko atmintis, tuo tarpu, palaipsniui augo. Sprendžiant iš rezultato, didinant egzempliorių skaičių taip pat auga ir papildomai sunaudota disko atmintis. Testų rezultatai parodė, kad daugiau kaip 40% sunaudotos disko atminties yra CouchDB optimizaciją, vėlesniam našumui išgauti.

Normalizuojant visus gautus duomenis ir sudarius santykinę lentelę (žr. Lentelė 20), galima nesunkiai palyginti tiek įvairius duomenų bazių plėtimo būdus, tiek skirtingas duomenų bazes tarpusavyje.

Duomenų bazė	Matuojami resursai	Plėtimo būdas					
		Vieno egzemplioriaus,1 branduolys ir 2 GB RAM	Vieno egzemplioriaus, 3 branduoliai ir 6 GB RAM	Vieno egzemplioriaus, 6 branduoliai ir 12GB RAM	Replikų rinkinys, 1 branduolys ir 2GB RAM egzemplioriui (3 egz.)	Replikų rinkinys, 2 branduoliai ir 4GB RAM egzemplioriui (3 egz.)	Klasteris, 1 branduolys ir 2GB RAM egzemplioriui (6 egz.)
MongoDB	Našumas	1	0,914	0,805	0,284	0,261	0,183
	Operatyvioji atmintis	1	2,338	4,273	0,654	1,004	0,707
	Disko atmintis	1			1,092		0,667
Redis	Našumas	1	0,597	0,350	0,255	0,234	0,308
	Operatyvioji atmintis	1	1,139	1,362	0,910	1,204	0,700
CouchDB	Našumas	1	0,816	0,522	0,256	0,259	0,184
	Operatyvioji atmintis	1			0,642		0,598
	Disko atmintis	1			1,014		1,066

Lentelė 20 Duomenų bazių plėtimo santykinė lentelė

Gautų rezultatų lentelių pagalba, galima priklausomai nuo poreikių ir galimybių, pasirinkti duomenų bazę. Rezultatai suteikia ne tik žinias apie galimus rezultatus atliekant įvairius plėtimus, bet leidžia ir teoriškai apskaičiuoti galimų kitokių variantų rezultatus.

Išvados

Šiame darbe buvo apžvelgtos NoSQL duomenų bazės ir duomenų bazių plėtimo būdai: vieno egzemplioriaus duomenų bazė, replikų rinkinio duomenų bazė ir klasterio duomenų bazė, pasinaudojant MongoDB, Redis ir CouchDB duomenų bazėmis. Taip pat, buvo išanalizuoti įrankiai, kurių pagalba buvo atliekami testai pasirinktoms duomenų bazėms:

- Docker
- Mongo-perf
- NoSQL spartos testai (angl. *NoSQL performance tests*)
- Redis Benchmarks
- Iron Cushion

Buvo nustatyti resursai, kurių pagalba būtų galima palyginti duomenų bazių plėtimo būdus:

- Disko atmintis
- Operatyvioji atmintis
- Operacijos per sekundę
- Procesoriaus branduoliai

ir atlikti testai duomenų bazėms, pasinaudojant pasirinktais įrankiais. Testų rezultatų pagalba buvo apskaičiuoti kaštai, kurie nurodo duomenų bazės plėtimo kainą.

Gauti rezultatai buvo supaprastinti, kad galima būtų įžvelgti vieno branduolio naudingumą duomenų bazei arba vieno egzemplioriaus reikalaujamų resursų kaštus tam tikru plėtimo būdo atveju. Palyginus duomenų bazes pasinaudojant iš testų rezultatų gautais duomenimis nustatyta, kad plėtimo būdo branduolių naudingumas kiekvienai duomenų bazei yra skirtingas. Pavyzdžiui MongoDB vieno egzemplioriaus atveju, resursų pridėjimas, branduolio naudingumą vidutiniškai sumažina 11.5%, o Redis atveju naudingumas sumažėja net daugiau kaip 40%. Turint tą omenyje, galima nuspręsti kiek resursų pridėti. Plečiant duomenų bazes, pastebėta, kad duomenų bazės plėtimas iš vieno egzemplioriaus į replikų rinkinį ar klasterį, branduolių naudingumą gali sumažinti labai skirtingai, vienu atveju našumo nuosmukis gali būti labai didelis (MongoDB), o kitu sąlyginai menkas arba gali net pagerinti situaciją (Redis). Tam tikrais atvejais resursų pridėjimas nedaug padeda pagerinti duomenų bazės veikimą. Gautų rezultatų pagalba galima nustatyti, kiek vidutiniškai pagerėja arba suprastėja branduolio naudingumas ir duomenis pritaikyti ateityje.

Operatyviosios atminties ir disko atminties atveju, įžvelgti skirtumai parodė, kad priklausomai nuo duomenų bazės, sąnaudos gali smarkiai skirtis. Kiekviena duomenų bazė yra vis kitokia ir resursų sunaudojimas skiriasi. Vienai duomenų bazei operatyvios atminties sąnaudos gali smarkiai išaugti pridedant resursų, o kitai tokiu pat atveju gali išaugti tik truputi. Todėl, neužtenka remtis tais pačiais faktais renkantis skirtingų duomenų bazių plėtimo būdus ir reikia geriau įvertinti situaciją.

Taip pat, buvo sudaryta santykinė MongoDB, Redis ir CouchDB plėtimo būdų lentelė, kurios pagalba galima ne tik plėtimo būdus duomenų bazėms palyginti tarpusavyje, bet galima palyginti ir pačias duomenų bazes, ko dėka buvo nustatyta:

- Plečiant Redis duomenų bazę, branduolio naudingumas suprastėja labiausiai lyginant su MongoDB ir CouchDB.
- Pridedant resursų vieno egzemplioriaus duomenų bazėms, labiausiai operatyvios atminties sąnaudos išaugo MongoDB.
- Santykinės metrikos atžvilgiu, CouchDB plėtimo į klasterį disko atminties reikalavimai išauga labiausiai.

Summary

Since long ago, storing data was very important. People used to store all of the data on paper, stone and other physical objects, that required a lot of space. At that time, data did not grow as fast as it does now. Constantly growing amount of data required more and more storage, but there was always a limit to space available. That was the point when digital data storage was the solution, relational database was one of the first efficient digital data storage ways, that provided not only storage, but functionality as well.

As data grew, relational databases kept getting slower, because of the amount of data it had to process. Then, Eric Evans re-introduced NoSQL databases in 2009. It was the solution that everyone sought for. But data never stopped growing, at some point servers couldn't handle the amounts of data they had to process, so resources would be added, but when the addition limit was reached, another way of expanding was required. Database scaling was the answer to the problem. By placing parts of data on different servers, it is possible to scale out as much as database needs to. But that does not come for free. Scaling out has its own advantages and disadvantages, such as scaling out requires more resources and a part of performance might be lost, but instead system gains persistency and availability.

The goal of this work is to analyze what is a NoSQL database and how different NoSQL database scaling methods perform while working in same conditions. That has been achieved by making an analysis of NoSQL databases and their scaling methods. Afterwards, some of the NoSQL popular databases were picked and tests were run for each of the databases, where each database was tested while scaled out in all of the ways available for the given situation. The selected databases were two document-oriented: MongoDB and CouchDB, as well as one Key-store oriented: Redis. The tests that were run tested database performance, as well as resource usage, such as CPU cores, RAM memory and disk space. The tests used were some of the most popular tests to each of the databases. New tests were not written, because the tools that were already being used were community approved and were developed correctly.

The results that were gotten by using selected tools provided data, by using which it was possible to analyze performance and memory usage of different database scaling types. To do that, data had to be simplified to single core case to analyze performance, and to single node case, to analyze memory usages. By normalizing the results, a ratio table was composed. Using the result tables, it is easier to decide on the optimal database scaling type for a given situation. By using ratio table selected databases can be easily compared and also theoretically it is possible to calculate data for some other scaling scenarios.

By analyzing the results, it was determined, that depending on a database, the performance increases or decreases differently. For example, by adding resources to MongoDB standalone database, single core performance decreases about 11,5% for each core added, but Redis results have shown, that the impact is much worse, around 40% of core performance was lost by expanding core resource from single core to 3 cores. On the other hand, MongoDB requires a lot of RAM and the demand increases the more resource is added, meanwhile Redis is very efficient in this field. Also, it has been determined that scaling into CouchDB cluster nearly doubles required disk space, compared to MongoDB Cluster. The ratio table has shown that by scaling Redis, core performance value decreases the most, by adding more resources to a database, MongoDB memory usage grew the most and CouchDB database scaling to sharded cluster disk memory requirements grew mostly compared to the other two databases.

Literatūra

[Als10] J. Chris Anderson, Jan Lehnardt, Noah Slater, CouchDB: The Definitive Guide: Time to Relax, O'Riley Media, January 2010.

[Apa16a] Apache CouchDB, http://couchdb.apache.org/

[Cou17] CouchDB Sharding, http://docs.couchdb.org/en/2.0.0/cluster/sharding.html

[Dbe17] DB-Engines rangink, https://db-engines.com/en/ranking

[Doc17] What is docker, https://www.docker.com/what-docker

[Kum16] Girish Kumar, Exploring the different types of NoSQL databases part II, http://www.3pillarglobal.com/insights/exploring-the-different-types-of-nosql-databases

[LM10] Adam Lith, Jakob Mattson, Investigating storage solutions for large data, Chalmers University of Technology, June 2010.

[Mai17] Redis Cluster Benchmark, https://github.com/maiha/redis-cluster-benchmark.cr

[Mon16a] NoSQL Databases Explained, https://www.mongodb.com/nosql-explained

[Mon16b] Mongo-perf, https://github.com/mongodb/mongo-perf

[Mon17a] MongoDB Standalone, https://docs.cloudmanager.mongodb.com/tutorial/deploy-standalone/

[Mon17b] MongoDB Shading, https://docs.mongodb.com/manual/sharding/

[Mon17c] MongoDB Replication, https://docs.mongodb.com/manual/replication/

[Par17] Iron Cushion, https://github.com/mgp/iron-cushion

[Red17a] Introduction to Redis, http://redis.io/topics/introduction

[Red17b] Redis Benchmarks, https://redis.io/topics/benchmarks

[Rou14] Margaret Rouse, MongoDB, March 2014.
http://searchdatamanagement.techtarget.com/definition/MongoDB

[Sad14] Pramod Sadalage, NoSQL Databases: An Overview, October 1, 2014.
https://www.thoughtworks.com/insights/blog/nosql-databases-overview

[Sco10] Ben Scofield, Overwiev of the NoSQL movement, CodeMach 2010,
http://www.slideshare.net/bscofield/nosql-codemash-2010/33-Data_Typesstringslistssetssorted_sets_33

[Str09] Christof Strauch, NoSQL Databases, Stuttgard Media University, June 10, 2009.

[Tec16] Guide to NoSQL databases: How they can help users meet big data needs
http://searchdatamanagement.techtarget.com/essentialguide/Guide-to-NoSQL-databases-How-They-can-help-users-meet-big-data-needs

[Wei16] Claudius Weinberger, NoSQL Performance Tests,
https://github.com/weinberger/nosql-tests