VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMŲ KATEDRA

NoSQL, NewSQL ir SQL duomenų bazių vertinimas ir palyginimas

NoSQL, NewSQL and SQL databases evaluation and comparison

Bakalauro baigiamasis darbas

Atliko:	Mantas Gedrimas	(parašas)
Darbo vadovas:	lekt. Andrius Adamonis	(parašas)
Darbo recenzentas:	doc. dr. Sigitas Dapkūnas	(parašas)

Vilnius – 2015

SANTRAUKA

Šio darbo tikslas – sudaryti kriterijų rinkinį, kuris padėtų palyginti SQL, NoSQL ir NewSQL duomenų bazių sistemas bei įvertinti kaip jų pasirinkti sprendimai gali įtakoti greitaveiką. Pagal sudarytą kriterijų, lyginamos ir vertinamos įvairios duomenų bazių sistemos. Darbą sudaro teorinė ir eksperimentinė dalis. Teorinėje dalyje aptariama, kas yra SQL, NoSQL ir NewSQL duomenų bazių sistemos, kokie parametrai ir programiniai sprendimai joms būdingi. Taip pat, aptariami kitų autorių atlikti eksperimentai, tiriant duomenų bazių sistemas skirtingais metodais. Eksperimentinėje dalyje atliktas eksperimentas, kiek kitokiu metodu lyginantis DBVS, lyginant su kitais darbe minėtais eksperimentais bei taip juos papildydamas. Gauti tyrimų rezultatai lyginami, pateikiamos išvados.

Raktiniai žodžiai: SQL, NoSQL, NewSQL, duomenų bazė, DBVS, greitaveika, palyginimas.

SUMMARY

The main purpose of this paper is to create a set of parameters which would allow to compare SQL, NoSQL and NewSQL database systems and evaluate how their differences affect their performance. In this paper, a set of parameters is created which is used to show how different database management systems could be compared and evaluated. The paper consists of two parts – theoretical and experimental. In first part it is stated what SQL, NoSQL and NewSQL systems are and what kind of characteristics they have. Besides that, experiments conducted by other authors are discussed while examining performance of database systems using various methods. In the second part experiment done by the author of the paper is explained. Results are compared and evaluated.

Keywords: SQL, NoSQL, NewSQL, database, DBMS, performance, benchmark.

TURINYS

ĮVADAS

Didėjant duomenų kiekiams, saugomiems duomenų bazėse, bei esant skirtingiems poreikiams, kaip su tais duomenimis dirbti, neužtenka vienos rūšies duomenų bazės, patenkinančius visų naudotojų lūkesčius. Šiame darbe nagrinėjamos trys duomenų bazių valdymo sistemų rūšys – SQL, NoSQL ir NewSQL.

Iškeltas toks šio **darbo tikslas** – sudaryti kriterijų rinkinį, kuris padėtų palyginti SQL, NoSQL ir NewSQL duomenų bazių sistemas bei įvertinti kaip jų pasirinkti sprendimai gali įtakoti greitaveiką.

Pagrindiniai darbo uždaviniai:

- Paaiškinti, kas yra SQL, NoSQL ir NewSQL duomenų bazių sistemos ir kokios esminės jų savybės,
- Sudaryti kriterijų rinkinį, kurį būtų galima naudoti norint palyginti SQL, NoSQL ir NewSQL duomenų bazių sistemas ir įvertinti, kaip jų pasirinktos savybės gali įtakoti greitaveiką,
- Remiantis šaltiniais, išanalizuoti pasirinktas SQL, NoSQL, NewSQL duomenų bazių valdymo sistemas pagal sudarytą kriterijų rinkinį,
- Išskirti metodus, kuriais būtų galima ištirti duomenų bazių sistemų greitaveiką,
- Praktiškai ištirti pasirinktų SQL, NoSQL ir NewSQL duomenų bazių valdymo sistemų užklausų vykdymo spartą.

Darbo temos aktualumas pasireiškia skirtingu skaitmeninių duomenų naudojimo pobūdžiu, t. y. atskiriems naudotojams svarbūs skirtingi aspektai ir galimybės, kurias suteikia duomenų bazės. Visgi esant didžiuliam duomenų bazių valdymo sistemų kiekiui bei kasmet atsirandant vis naujoms jų atstovėms, sunku objektyviai įvertinti ir palyginti duomenų bazių sistemas. Šiam tikslui pasitarnauti galėtų sudarytas kriterijų rinkinys.

Sudarytas rinkinys, nagrinėjamas darbe, susideda iš komponentų, leidžiančių palyginti ir įvertinti duomenų bazių spartą, plečiamumą, duomenų organizavimą bei ACID transakcijų palaikymą. Šie parametrai turėtų leisti palyginti ir įvertinti didžiąją dalį egzistuojančių duomenų bazių valdymo sistemų. Pagal kriterijų rinkinį, tiriamos skirtingos DBVS, remiantis įvairiais moksliniais straipsniais, nagrinėjant kitų autorių atliktas analizes, apžvalgas, pateikiami samprotavimai, įvertinami įvairūs jų aspektai bei pateikiamos išvados. Taip pat, remiantis nagrinėtais šaltiniais bei siekiant papildyti kitų autorių atliktus tyrimus, praktiškai išnagrinėta atrinktų SQL, NoSQL ir NewSQL duomenų bazių sistemų CRUD operacijų vykdymo sparta, atliekant simuliaciją pagal sudarytus operacijų vykdymo scenarijus.

Eksperimentinėje dalyje nagrinėjamos trys pasirinktos duomenų bazių valdymo sistemos. Nagrinėjamas šių duomenų bazių sistemų operacijų vidutinis vykdymo laikas. Gauti duomenys lyginami su kitų autorių pastebėjimais, įvertinamos duomenų bazių sistemos.

Teorinė ir praktinė darbo dalis parodo, kokiais aspektais gali būti lyginamos ir vertinamos skirtingos DBVS, nagrinėjant pavyzdžius parodomas tokio lyginimo ir vertinimo efektyvumas naudojant sudarytus kriterijus ir vertinant bet kokią kitą pasirinktą SQL, NoSQL ar NewSQL duomenų bazių valdymo sistemą.

1. KAS YRA SQL, NOSQL IR NEWSQL DUOMENŲ BAZIŲ SISTEMOS?

1.1. SQL duomenų bazių sistemos

SQL duomenų bazių sistemos yra seniausiai sukurtos tarp SQL, NoSQL ir NewSQL. Jos apibrėžiamos naudojant reliacinį modelį. Pagrindinė modelio sąvoka yra lentelė, sudaryta iš eilučių ir stulpelių (žr. paveiksliuką žemiau). Bendravimas su tokiomis duomenų bazėmis įprastai vykdomas rašant užklausas SQL (angl. „Structured Query Language“) kalba. Duomenys jose yra tipizuoti – t. y. jie gali būti numerio, eilutės, datos, baitų ar kito formato. Svarbu pastebėti, kad SQL duomenų bazių sistemoms galimas lentelių apjungimas suformuojant naujas, sudėtingesnes lenteles, kadangi joms būdinga matematinė aibių teorijos prigimtis [RW12, 3]. Taip pat skiriamieji SQL duomenų bazių valdymo sistemų bruožai yra stabilių ACID transakcijų palaikymas (apie tai plačiau vėlesniuose skyriuose), užtikrinantis duomenų nuoseklumą, „vertikalus“ plečiamumas, grindžiamas techninių komponentų pridėjimu į serverį [RW12, 3].

Remiantis db-engines.com populiarumo reitingu [DBE16] (2016 metų gegužės mėn.) SQL duomenų bazių sistemos dešimtuke užima 7 vietas: „Oracle” (1), „MySQL“ (2), „Microsoft SQL Server“ (3), „PostgreSQL” (5), „DB2” (6), „Microsoft Access” (8), „SQLite” (10).

ID	MODEL	YEAR	DRIVERID
1	updated0 car	2001	1
2	updated0 car	1990	2
3	updated0 car	2013	3
4	updated0 car	1998	4
5	updated0 car	1996	5
6	updated0 car	2001	6
7	updated0 car	1992	7
8	updated0 car	1996	8

1 pav. Lentelės pavyzdys.

1.2. NoSQL duomenų bazių sistemos

NoSQL (arba angl. „Not only SQL“) yra sąlyginai naujos (sukurtos apytiksliai šio tūkstantmečio pradžioje) ir gana sparčiai populiarėjančios duomenų bazių sistemos. Jų veikimo principas nėra paremtas įprastu SQL duomenų bazėms reliaciniu modeliu, t. y. klientas gauna duomenis kaip sąrašą „raktas-reikšmė“ objektų, stulpeliais, dokumentais, grafais ar kitu būdu. Analogiškai išskiriamos tokios pagrindinės NoSQL sistemų rūšys:

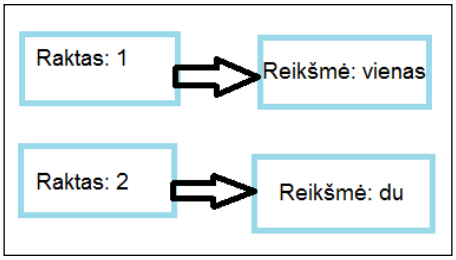
- „Raktas-reikšmė“,
- Dokumentų,
- Stulpelių,
- Grafų.

Kaip byloja pavadinimas (liet. „Ne tik SQL“), šios duomenų bazių sistemos yra tarsi kitas polius lyginant su SQL duomenų bazių sistemomis. Nors kai kurios iš NoSQL duomenų bazių valdymo sistemų irgi turi galimybę vykdyti užklausas SQL kalba, visgi, skirtingai nei SQL duomenų bazių sistemos, NoSQL dažniausiai pilnai neįgyvendina ACID principų (3, 4 skyriai), nes tai yra DBVS, kuriose yra įgyvendintas duomenų bazės padalijimas į atskirus serverius, sujungtus tarpusavyje. Remiantis CAP teorema, kompiuterinė sistema gali užtikrinti tik 2 dalykus iš 3 – darnos, pasiekiamumo ir padalijimo [RW12, 317]. Kalbant apie NoSQL duomenų bazių sistemas, laikoma, kad padalijimas pagrįstas tinklu, tad reikia įgyvendinti darną arba pasiekiamumą. Dažniausiai siekiama didesnio pasiekiamumo apie darną galvojant kaip „galiausiai įvyksiančią darną“ (angl. „eventual consistency“) [Pok14] – t. y. duomenys, esant pasikeitimams gali būti netvarkingi ir kartais pasenę, tačiau galiausiai jie bus atnaujinti.

Pagal db-engines.com populiarumo reitingą [DBE16] (2016 metų gegužės mėn.) NoSQL duomenų bazių valdymo sistemos dešimtuke turi 3 atstovus: „MongoDB” (4), „Cassandra” (7), „Redis” (9).

1.2.1. „Raktas-reikšmė“ DBVS

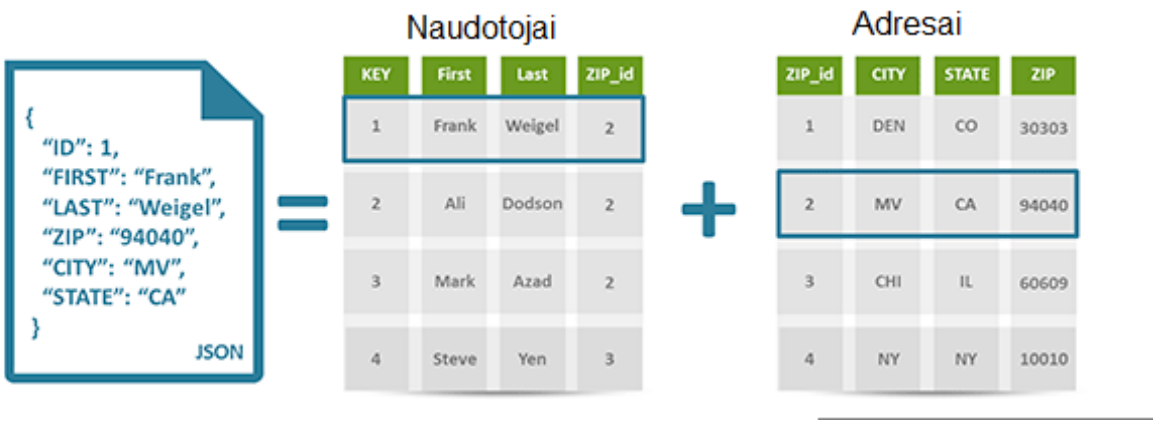
„Raktas-reikšmė“ duomenų bazių sistemų struktūra sąlyginai paprasta – kaip jau nusakoma pavadinimo, duomenų bazėse saugomos rakto ir reikšmės poros (visai panašiai kaip maišos lentelėse ar žodynuose saugomi raktai ir reikšmės kokios nors programavimo kalbos aplinkoje, žr. paveiksliuką žemiau) [RW12, 3]. Kai kurios „raktas-reikšmė“ duomenų bazių sistemų atstovės leidžia tokias reikšmės kaip sąrašai, tačiau tai nėra būtina – reišmės gali būti ir teksto formos. Paieška tokiose duomenų bazėse vykdoma tik pagal raktą, todėl operacijų sparta itin didelė, o užklausos paprastos. Visgi, tokios duomenų bazių valdymo sistemos netinkamos, esant poreikiui vykdyti sudėtingas užklausas ar agreguoti duomenis. Dėl didelės spartos ir plečiamumo yra tinkamos sistemoms, kurioms reikia sparčiai pasiekti nedideles porcijas duomenų (pvz., vartotojų profilių valdymas). Žymesni atstovai: „Dynamo“ (naudojamas „Amazon“), „Voldemort“ (naudojamas „LinkedIn“), „Redis“, „BerkeleyDB“, „Riak“. [MH13]



2 pav. Rakto-reikšmės pavyzdys.

1.2.2. Dokumentų DBVS

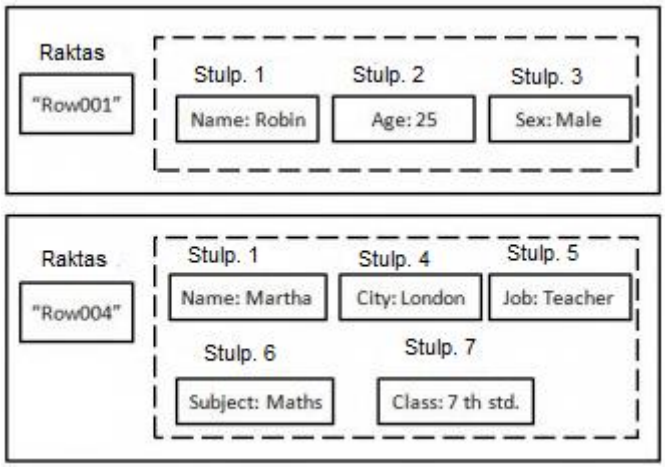
Dokumentų duomenų bazių sistemos, kaip teigia jų pavadinimas, buvo sukurtos darbui su dokumentais, kurie saugomi tokiais formatais kaip XML, JSON ar BSON. Priešingai nei „raktas-reikšmė“ duomenų bazėse, reikšmė turi dalinai struktūrizuotus duomenis, susidedančius iš atributų ir tų atributų reikšmių. Stulpelis gali turėti daugybę atributų, o jų skaičius ir tipas kiekvienoje eilutėje gali būti skirtingas. Taip pat dokumentų duomenų bazėse galima paieška tiek pagal raktą, tiek pagal reikšmę. [MH13] Be darbo su dokumentais šios duomenų bazių valdymo sistemos puikiai tinka tokiais atvejais, kur reliacinėse duomenų bazėse reikšmes reikėtų apibrėžti kaip „null“ – čia jas tiesiog galima praleisti [HJ11]. Žymesni atstovai: „MongoDB“ (BSON), „CouchDB“ (JSON).



3 pav. Dokumentų DB saugomo dokumento palyginimas su lentelėmis. Originalas: <http://refreshmymind.com/wp-content/uploads/2016/02/nosql_document_stores_mongodb_couchdb.png>

1.2.3. Stulpelių DBVS

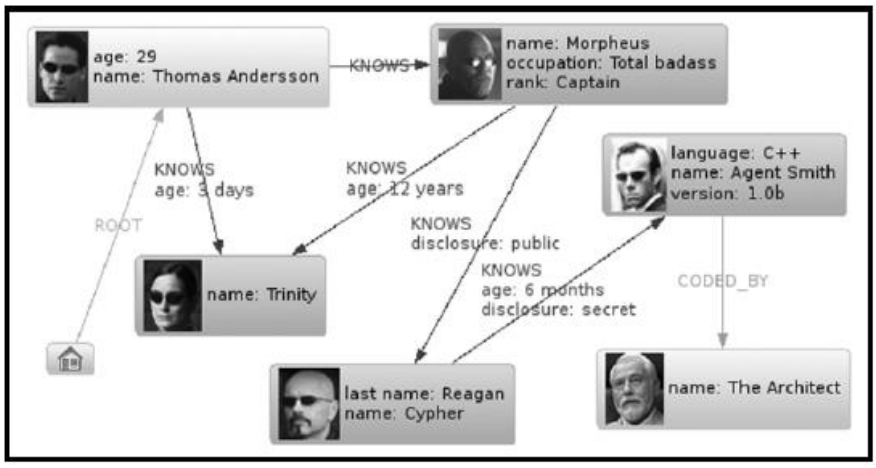
Stulpelių duomenų bazių sistemose pagrindinis elementas – stulpelis. Duomenys saugomi stulpeliais (panašiai kaip reliacinėse duomenų bazėse jie saugomi eilutėse, žr. paveikslėlį žemiau). Jose saugomi į stulpelius orientuoti duomenys (t. y. saugomi lentelės atributų sąrašai), kurie gali turėti savo atributus. Tai padeda sparčiai išanalizuoti norimo atributo (stulpelio) visų įrašų duomenis. Šioks toks skirtumas atsiranda, nes stulpeliais grįstų duomenų bazių struktūra šiek tiek griežtesnė, tačiau tai netrukdo dinamiškai pridėti atributus (stulpelius). Stulpeliais grįstų duomenų bazių sistemos gali būti naudojamos siekiant saugoti paskirstytus duomenis, norint atlikti didelio masto tos pačios rūšies duomenų operacijas (pvz. įvairūs konvertavimai, rikiavimai ir pan.), siekiant atlikti tiriamojo ar prognozuojamojo pobūdžio analitinius uždavinius. [MH13] Žymesni atstovai: „Bigtable“ (naudojamas „Google“), „Hypertable“, „Cassandra“ (naudojamas „Facebook“), „SimpleDB“ (naudojamas „Amazon“), „DynamoDB“.



4 pav. Stulpelių DB saugomų duomenų pavyzdys

1.2.4. Grafų DBVS

Grafų duomenų bazių sistemos, kiek rečiau naudojamos, labiau specializuotos NoSQL duomenų bazių valdymo sistemų atstovės. DB struktūros esmė – grafas, t. y. duomenų bazė sudaryta iš viršūnių ir kraštinių, kurie turi „raktas-reikšmė“ poras, saugančias duomenis (žr. paveiksliuką žemiau). Dažniausiai naudojamos ten, kur svarbūs ryšiai tarp viršūnių, o jų pagrindinis privalumas – greitas perėjimas duomenų bazės viršūnėmis, pvz. vaizduojant ar „pereinant“ pro socialinių tinklų duomenis. Verta pabrėžti, kad grafų duomenų bazių sistemos būtent tinkamos „pereiti“ per reikšmės, o ne gauti jas užklausų pagalba. [MH13] Žymesni atstovai: „Neo4j”, „InfoGrid”, „Sones GraphDB”, „AllegroGraph”, „InfiniteGraph”.



5 pav. Grafo DB saugomų duomenų pavyzdys. Šaltinis: [MH13]

1.3. NewSQL duomenų bazių sistemos

NewSQL yra naujausios tarp šių trijų duomenų bazių sistemų. Visgi tai net ne visai atskira duomenų bazių sistemų rūšis. Kalbant apie NewSQL reikėtų paminėti, kad jų DB, kaip ir SQL duomenų bazės, yra reliacinių duomenų bazių porūšis, įgyvendinami ACID principai [UC14], o

bendravimas su jomis vykdomas rašant SQL užklausas. Vis dėlto esminis skirtumas tas, kad šiomis duomenų bazių valdymo sistemomis siekiama išvystyti tokią pačią spartą ir plečiamumą kaip NoSQL. Tai įgyvendinama pasitelkiant tokius architektūrinius sprendimus kaip komponentų autonomiškumas blokiniuose arba duomenų talpinimas pagrindinėje atmintyje [Pok14].

Remiantis db-engines.com populiarumo reitingu [DBE16] (2016 metų gegužės mėn.) NewSQL duomenų bazės dešimtuke savo atstovų neturi. Arčiausiai dešimtuko yra „SAP HANA“ (19). Kiti žymesni atstovai: „MySQL Cluster“, „Clustrix“, „NuoDB“, „VoltDB“ [Ima16].

2. KRITERIJŲ RINKINYS

Norint ištirti ir palyginti skirtingas SQL, NoSQL ir NewSQL duomenų bazes ir jų valdymo sistemas bei jų greitaveiką būtina sudaryti kriterijų rinkinį, leidžiantį tai atlikti. Šiame skyrelyje aptariamas rinkinys, kuriuo remiantis darbe tiriamos duomenų bazių sistemos.

Pirmasis kriterijus – ACID transakcijų palaikymas. Tai įtakoja, ar transakcijos bus stabilios, duomenys nuoseklūs, svarstoma, kodėl jas įgyvendinant galima mažesnė greitaveika bei kokios galimos alternatyvos.

Antrasis kriterijus – plečiamumas. Tai svarbus kriterijus, nes geras plečiamumas reiškia, kad duomenų bazių sistema galės sparčiai dirbti su dideliais kiekiais duomenų bei bus lanksti, jei tų duomenų daugės, kadangi sistemą galės praplėsti. Pagal šį kriterijus tiriama, kokiu būdu plečiamos duomenų bazių sistemos, ar vykdomas duomenų padalijimas ir kaip jis įgyvendinamas. Analizuojama, kaip sistemos plėtimas įtakoja greitaveiką.

Trečiasis kriterijus – duomenų organizavimas. Efektyviai valdomi ir tvarkomi duomenys arba pritaikomi įvairūs programiniai sprendimai, kartais galbūt aukojantys mažiau svarbius aspektus, kurie reikalingi darbui su duomenimis, o tai reiškia geresnę duomenų bazių sistemos spartą. Šis kriterijus kiek platesnis – aptariama, kaip ir kur saugojami duomenys, kokiu būdu vykdomos operacijos, ar yra tokių dalykų kaip indeksai, kurie pagreitintų operacijų spartą, kaip sprendžiama „konkuruojančių“ operacijų problema. Analizuojama, kaip skirtingas duomenų organizavimas veikia sistemų greitaveiką.

Ketvirtasis kriterijus – operacijų vykdymo sparta. Tiriama, kaip veikia duomenų bazių sistemos įvairiais skirtingais veikimo scenarijais – kiek operacijų įvykdoma per sekundę (angl. „throughput“), koks gaišties laikas. Atsižvelgiama į plečiamumą. Analizuojama, per kiek laiko atliekamos įvairios CRUD operacijos.

3. DBVS ACID ATŽVILGIU

Pirmasis nagrinėjamas kriterijus, kuris gali svarbus naudojantis duomenų bazių valdymo sistemoms – stabilių ACID transakcijų palaikymas. ACID (angl. „Atomicity, Consistency, Isolation and Durability“) – tai tokios transakcijos, kurioms būdingas:

- Vientisumas (arba nedalumas) – savybė, kuri reiškia, kad bus arba įvykdytos visos transakcijos, arba nė viena. Pvz.: klaidos atveju jeigu viena transakcija nepavyksta, visos kitos taip pat yra atstatomos į prieš tai buvusią būseną,
- Darna (arba nuoseklumas) – savybė, kuri reiškia, kad tiek prieš transakciją, tiek po jos turi būti išlaikoma duomenų bazės būsena be jokių klaidų,
- Izoliuotumas – savybė, kuri apibrėžia, kad skirtingų transakcijų rezultatai būtų matomi, jei viena transakcija eina po kitos,
- Patvarumas – savybė, kuri reiškia, kad jeigu transakcija buvo įvykdyta, duomenų bazės būsena išliks tokia, kaip buvusi, net jei pvz. įvyktų kokia klaida, ar dingtų maitinimas.

Pagal CAP (angl. „Consistency, Availability, Partitioning“) teoremą, kompiuterinė sistema gali užtikrinti tik 2 dalykus iš 3 – darnos, pasiekiamumo ir padalijimo [RW12, 317]. Visgi, kalbant apie sistemas, palaikančias ACID transakcijas, iškyla daug problemų su geru pasiekiamumu, jeigu sistema yra padalinta. SQL duomenų bazių sistemos pilnai įgyvendina ACID principus transakcijoms. Duomenys yra nedalūs, visada nuoseklūs. Vis dėlto, problemos atsiranda siekiant padalijimo.

NoSQL atveju ši problema sprendžiama kiek sumažinant reikalavimus duomenų nuoseklumui, siekiant geresnio pasiekiamumo ir padalijimo. Taip atsirado sistemos, veikiančios BASE (angl. „Basically available, Soft-state, Eventualy consistent“) principu, kuris reiškia, kad duomenų bazė neturi tradicinių transakcijų mechanizmo, o apribojimai pritaikomi duomenų modeliui, siekiant gauti geriau padalijimui pritaikytas schemas. Kitaip sakant, apie nuoseklumą galvojama kaip apie „galiausiai įvyksiantį“ (angl. „eventualy consistent“). [MH13] Tokiu būdu, NoSQL duomenų bazėse duomenys ir paskirstyti ir beveik visada pasiekiami, tačiau gali būti atvejų, kada jie yra ne visai nuoseklūs. Taip pat, mažesnį dėmesį skiriant duomenų vientisumui, galima greičiau vykdyti operacijas.

NewSQL padalinimo problemą sprendžia kiek kitaip. Šios duomenų bazių valdymo sistemos kaip ir SQL išlaiko ACID transakcijas, o savo ruožtu padalijimą pagerina, naudodamos blokiniuose autonomiškus komponentus, dirbdamos su replikomis arba talpindamos duomenis operatyviojoje atmintyje [Pok14].

4. PLEČIAMUMAS IR DUOMENŲ ORGANIZAVIMAS

Plečiamumas – vienas iš pagrindinių dalykų, išryškėjantis kalbant apie SQL ir NoSQL arba NewSQL duomenų bazių sistemas ir stipriai įtakojantis jų greitaveiką, ypač esant dideliam duomenų kiekiui. SQL pasižymi tuo, kad jos plečiamumas gana ribotas – tokios duomenų bazių sistemos plečiamos „vertikaliai“. Tai reiškia, kad norint padidinti jų dydį, reikia pridėti į sistemą papildomų techninės įrangos komponentų, tokių kaip procesorius, operatyvioji atmintis, kietieji diskai. Tokiose sistemose procesoriai (ar jų branduoliai) tarpusavyje dalinasi tiek operatyviąja atmintimi, tiek kietaisiais diskais, tad komponentai yra glaudžiai susiję [Cat10]. Ir nors teoriškai darbui su dideliu duomenų kiekiu galima naudoti daugiau techninės įrangos resursų, gali atsirasti kitokių problemų, pvz.: duomenų siuntimo pralaidumas, duomenų bazės veikla sugedus tam tikriems komponentams ir pan.

Kalbant apie NoSQL ar NewSQL, šioms duomenų bazių sistemoms būdingas „horizontalus“ plečiamumas. Tai reiškia, kad duomenis galima paskirstyti per skirtingus serverius, kurie turi atskirus nepriklausomus komponentus [Cat10]. Tokiu būdu atsiranda galimybė paskirstyti operacijas skirtingiems serveriams padidinant pralaidumą, o kartu ir operacijų vykdymo spartą. Taip pat, replikų pagalba padalinti duomenis. Kadangi SQL plečiamumas grindžiamas fizinių komponentų papildymu, o ne programiniu sprendimu, šiame skyriuje didesnis dėmesys bus skirtas konkretesniems NoSQL ir NewSQL atvejams.

4.1. „Raktas-reikšmė“ DBVS

Nagrinėjamos duomenų bazių valdymo sistemos – „Voldemort“, „Riak“, „Redis“, „Scalaris“.

Visos iš jų įgyvendina plečiamumą paskirstydamos „hašuotus“ raktus po skirtingus serverius. Savo ruožtu „Scalaris“ taip pat įgalina paskirstymą naudodama raktų aibes, kas suteikia geresnį operacijų paskirstymą ir didesnę spartą dirbant su duomenų aibėmis.[Cat10] „Riak“ operacijų spartą gerina sukurdama vadinamus sąryšius tarp duomenų, pagal kuriuos taip pat galima atlikti paiešką. Be to, naudoja „map-reduce“ mechanizmą atlikti operacijas paraleliai. [Bas16]

„Voldemort“, „Riak“ duomenys laikomi operatyviojoje atmintyje arba kietajame diske. Kitose duomenų bazių sistemose duomenys saugomi operatyviojoje atmintyje, naudojant kietąjį diską kaip priemonę saugoti atsargines kopijas arba naudojant replikavimąsi ir atstatymą iš replikų apsieinant be atsarginių kopijų. [Cat10] Operatyviosios atminties panaudojimas taip pat sudaro prielaidas greitesniam duomenų bazių veikimui.

Visų nagrinėjamų duomenų bazių replikos kuriamos asinchroniškai. Tai reiškia, kad serverių resursai tuo pat metu gali būti skirti atlikti kitoms operacijoms, tačiau taip pat reiškia, kad

duomenys ne visada gali būti nuoseklūs skirtingose replikose. Vis dėlto, „Scalaris“ prieš užbaigdama operaciją, privalo atnaujinti daugumą kopijų [ZIB16]. Tuo būdu, naudojant šią duomenų bazių valdymo sistemą užtikrinamas duomenų nuoseklumas. „Voldermort“, „Riak“ problema sprendžiama patikrinant didžiąją daugumą replikų [Cat10], t. y. laikoma, kad duomenys naujausi, jei paėmus didelę dalį replikų, visose reikšmės sutampa.

„Scalaris“ palaiko ACID transakcijų mechanizmą [ZIB16], dėl savo replikų atnaujinimo prieš užbaigiant operaciją funkcionalumo. Vis dėl to skirtingai nei dauguma kitų NoSQL duomenų bazių valdymo sistemų, ši DBVS labiau koncentruojasi ties nuoseklumu ir padalijimu, todėl šiek tiek kenčia pasiekiamumas.

Tam, kad vienam vartotojui skaitant duomenis, o kitam rašant, skaitantysis nepamatytų nenuoseklių ir dalinai redaguotų duomenų „Voldemort“ ir „Riak“ naudoja „konkuruojančių“ operacijų valdymui skirtą MVCC (multi-versijų konkuravimo kontroliavimo) mechanizmą. Jo principas remiasi tuo, kad vartotojas mato tam tikro laiko duomenų bazės duomenis, o kitų naudotojų pakeitimus gali pamatyti tik jų transakcijai jau įvykus. Duomenų versijos kontroliuojamos „Dynamo“ [PT08] naudojamu vektorinio laikrodžio mechanizmu. Jo pagalba galima nurodyti, kokią versiją norima atnaujinti. Kitos nagrinėjamos duomenų bazių valdymo sistemos naudoja „optimistinio užrakinimo“ mechanizmą, kuris neleidžia atlikti operacijos, kol neįvyko kita. [Cat10] Šis būdas kiek lėtesnis, kadangi vienas naudotojas būna priverstas kartoti savo operaciją, tačiau greitesnis nei „pesimistinis užrakinimas“, dažnai naudojamas SQL duomenų bazių valdymo sistemose, kuris reiškia, kad užrakinama lentelė ar eilutė, kas kartais gali virsti amžinu užrakinimu (angl. „deadlock“), t. y. nė viena konkuruojanti operacija nesibaigia.

„Voldemort“, „Riak“, „Scalaris“ automatiškai ieško neveikiančių mazgų ir bando juos pataisyti. Šiam tikslui „Riak“ naudoja „Gossip“ protokolą, kur duomenų bazės valdymo sistemos mazgai tarpusavyje komunikuoja pranešdami apie savo būseną ar atnaujinimų atsiradimą atsitiktinai pasirinktam mazgui. [Cat10] Tokiu būdu užtikrinama teisinga sistemos būsena – apie klaidas sužinoma automatiškai. Mazgų pridėjimas „Voldemort“ ar „Riak“ atveju gali būti vykdomas nepertraukiant darbo – sistema prisitaiko automatiškai, „Redis“ – reikalingas sistemos išjungimas. [Cat10]

4.2. Dokumentų DBVS

Dokumentų duomenų bazių valdymo sistemų nagrinėjami atstovai – „SimpleDB“, „CouchDB“, „MongoDB“, „Terrastore“.

Tokių sistemų duomenų bazės neturi schemos, o tik atributus, dokumentų sąrašus („SimpleDB“ – domenus, „MongoDB“ – kolekcijas ir kt.) ir tų sąrašų indeksus. Tai reiškia, kad tokios sistemos neturi atlikti veiksmų, užtikrinančių duomenų struktūrą, todėl operacijas gali atlikti

kiek greičiau. Šiek tiek skiriasi nagrinėjamų atstovių duomenų modelis, pvz. „SimpleDB“ nepalaiko dokumentų įterptų kituose dokumentuose. Siekiant spartesnės paieškos naudojami indeksai – „SimpleDB“ ir „MongoDB“ dokumentų sąrašai indeksuojami automatiškai, o „CouchDB“ – nurodomi naudotojo. [Cat10]

Priešingai nei „raktas-reikšmė“ duomenų bazių valdymo sistemos, dokumentų DBVS naudoja ne tik pirminio rakto indeksus. Galima paieška pagal įvairius dokumentų parametrus. Taip pat „MongoDB“ ir „Terrastore“ suteikia galimybę atlikti ir kiek sudėtingesnes užklausas naudojant predikatus. [Goo16]

Visos nagrinėjamos sistemos išskyrus „SimpleDB“ naudoja „map-reduce“ mechanizmą, leidžiantį vykdyti operacijas skirtinguose serveriuose paraleliai. [Cat10] Tai taip pat padidina šių duomenų bazių sistemų greitaveiką.

„SimpleDB“ duomenys saugomi „Amazon“ serveriuose („S3“). „MongoDB“ ir „CouchDB“ – kietuosiuose diskuose, o „Terrastore“ – operatyviojoje atmintyje. [Cat10]

Dokumentų duomenų bazių sistemos nepalaiko ACID transakcijų. Remiamasi BASE principu, t. y. apie nuoseklumą galvojant kaip apie galiausiai įvyksiantį. „Konkuruojančių“ operacijų problema „CouchDB“ sprendžiama MVCC mechanizmu, o „MongoDB“ ir „Terrastore“ – „optimistiniu užrakinimu“ ir atominėmis operacijomis laukams, t. y. jos yra paprastos (pvz. didinimas vienetu), tad nereikalauja grįžtamojo ryšio su serveriu, kas taip pat didina jų greitaveiką. „MongoDB“ laukus galima atnaujinti tik tada, jeigu jų reikšmė yra naujausia (t. y. duomenys nėra pasenę). Padalinimo funkcionalumą gali įgyvendinti visos (įskaitant „CouchDB“, tačiau tam jai reikia „CouchDB Lounge“ programinio kodo).[Cat10]

Dokumentai šiose duomenų bazių sistemose gali būti paskirstyti per kelis serverius, tačiau pats plečiamumas pasiekiamas skirtingai. Visos iš nagrinėjamų DBVS plečiamumą gali įgyvendinti skaitant replikas, tačiau „SimpleDB“ ir „CouchDB“ atveju duomenys gali būti pasenę, kadangi replikos kuriamos asinchroniškai. Taip pat, šios duomenų bazių valdymo sistemos leidžia tik skaityti iš replikų, o ne įrašyti naujus duomenis – iš vienos pusės, skaitymo operacijos greitos, tačiau rašymo – kiek lėtesnės. Savo ruožtu „MongoDB“ ir „Terrastore“ replikas naudoja mazgų atstatymui įvykus klaidoms, o plečiamumas pasiekiamas duomenų padalijimu skirtinguose serveriuose (angl. „sharding). Be to jos įgyvendina duomenų rašymą į skirtingas replikas. Taip yra dėl padalijimo funkcionalumo, kuris suskaido duomenų bazę į smulkesnes dalis, bei dėl atominių operacijų. [Cat10]

4.3. Stulpelių DBVS

Šiame poskyryje nagrinėjami duomenų bazių valdymo sistemų atstovai – „HBase“, „HyperTable“, „Cassandra“.

Stulpelių duomenų bazių valdymo sistemos perėmė pagrindinius aspektus iš „Google BigTable“. Jų DB duomenų modelis sudarytas iš eilučių ir stulpelių, o pastarųjų plečiamumas remiasi eilučių ir stulpelių padalinimu per kelis serverius: eilutės padalinamos skirtingiems mazgams naudojant „sharding“ mechanizmą pagal pirminį raktą; stulpeliai paskirstyti skirtingiems mazgams naudojant „stulpelių grupes“ (tai būdas sugrupuoti susijusius stulpelius kartu). [Cat10] „Cassandra“ įgyvendina funkcionalumą automatiškai pridėti naujus mazgus į klasterį, o atsiradus klaidoms, automatiškai jas sprendžia. [ASF16]

Duomenys nagrinėjamų sistemų duomenų bazėse saugojami operatyviojoje atmintyje, o iš ten kompaktiškai palaipsniui perkeliami į kietuosius diskus. [Cat10] Tai leidžia pasiekti didelę duomenų apdorojimo spartą, kadangi nereikia nuolat naudoti lėtesnių kietųjų diskų. „Cassandra“ duomenys be minėtų grupavimo būdų stulpeliais dar gali būti grupuojami į „super stulpelius“, kuriuos sudaro keli stulpeliai [ASF16] (SQL duomenų bazėje tai būtų kažkas panašaus į vaizdą iš kelių lentelių). Šiuo būdu galima greičiau pasiekti taip sugrupuotus duomenis iš karto.

Visos iš jų vykdo replikų kūrimą, tačiau „HyperTable“ jis sinchroniškas, o kitose asinchroniškas. Kalbant apie „konkuruojančių“ operacijų organizavimą, „HBase“ naudoja „optimistinio užrakinimo“ mechanizmą, o „Cassandra“ ir „HyperTable“ – MVCC. „Cassandra“ DBVS atveju duomenys ne visada nuoseklūs, tačiau jei reikalingi būtinai naujausi duomenys, kaip anksčiau nagrinėtose duomenų bazių valdymo sistemos (pvz. „Voldemort“) tai galima pasiekti skaitant didžiąją dalį replikų. [Cat10]

„HBase“ palaiko atomines operacijas eilutėms [Cat10], todėl didesnė sparta pasiekiama būtent dirbant su jomis, o „Cassandra“ – atomines rašymo operacijas stulpeliams. [ASF16]

4.4. NewSQL DBVS

Nagrinėjami NewSQL duomenų bazių valdymo sistemų pavyzdžiai: „MySQL Cluster“, „VoltDB“, „Clustrix“ ir „NuoDB“.

Skirtingai nei NoSQL sistemų duomenų bazės, NewSQL turi griežtą schemą, vykdomos ACID transakcijos. Galima pastebėti, kad taip pat NewSQL duomenų bazių sistemos pasiekia panašų plečiamumą kaip NoSQL, išskyrus atvejus, kai jungiamos lentelės, arba naudojamos transakcijos, apimančios daug serverio mazgų. Tačiau NoSQL jungimo galimybės apskritai neturi. [Cat10]

Pirmosios trys DBVS atstovės automatiškai paskirsto duomenis dalimis per serverius, t. y. remdamosi „shared-nothing“ architektūra. „NuoDB“ duomenų padalijimą įgyvendina dinamiškai saugodama duomenis ir jų kopijas skirtinguose serveriuose. Visos taip pat kuria duomenų replikas, skirtas atstatymui. „MySQL Cluster“ ir „NuoDB“ paskirsto duomenis geografiškai [Nuo16], t. y. tose vietose, kur jie naudojami daugiausiai, kas paspartina operacijų atlikimą atitinkamiems

naudotojams. „VoltDB“ suteikia galimybę naudoti replikavimąsi siekiant padidinti greitaveiką, kai didžioji dauguma operacijų yra skaitymas. „Clustrix“ automatiškai paskirsto duomenis po serverius. Taip pat automatizuotas ir naujų mazgų pridėjimas, atstatymas klaidų atveju. [Cat10]

„VoltDB“ duomenų bazėje SQL užklausos įprastai vykdomos per apsirašytas procedūras, kur viena procedūra atitinka vieną transakciją, t. y. „VoltDB“ užtikrina, kad procedūra bus ACID. Jei duomenys padalinti taip, kad transakcija galėtų būti atlikta viename mazge, tada nereikalingas „užrakinimas“ (procedūra vykdoma automatiškai, paraleliai leidžiant naudotojui toliau vykdyti užklausas), o kartu dingsta ir kito naudotojo laukimas, kol atliekama transakcija. [Vol16] Dėl duomenų balansavimo serveriams ir eilučių versijų saugojimo „Clustrix“ taip pat išvengia „užrakinimo“ [Clu16].

„MySQL Cluster“ duomenų bazėje duomenys laikomi diskuose arba operatyvioje atmintyje, kas leidžia organizuoti darbą su duomenų baze realiu laiku. Šiuo aspektu dar labiau pažengusi „VoltDB“, kuri suprojektuota taip, kad duomenų bazė tilptų serverių RAM atmintyje. Taip pat, indeksai ir įrašymo struktūra suprojektuota darbui su operatyviąja atmintimi, todėl operacijos vykdomos sparčiau, nes beveik neatsiranda perteklinių veiksmų darbui su buferiais, podėliu. [Vol16] Visgi greitaveika smarkiai suprastės, jei virtuali atmintis perpildys operatyviąją atmintį, todėl tam išvengti reikia gero planavimo. Savo ruožtu „Clustrix“ naudoja SSD diskus, kas taip pat leidžia pasiekti gerą našumą, lyginant su HDD diskais. [Cat10] „NuoDB“ duomenis laiko diskuose, tačiau greitaveikos pagerinimui gali naudoti operatyviąją atmintį laikinų lentelių saugojimui. [Nuo16]

4.5 Apibendrintai

SQL duomenų bazių sistemos išlaiko aukštą duomenų nuoseklumą ir vykdo ACID transakcijas, tačiau joms būdingas tik „vertikalus“ plečiamumas. Kol duomenų srautas nėra didelis, užklausos gali būti vykdomos sąlyginai sparčiai, o duomenys yra nuoseklūs, tačiau padidėjus duomenų kiekiui, atsiranda problemų norint tuos duomenis paskirstyti, kadangi pagal CAP teoremą, SQL duomenų bazių sistemos palaiko duomenų nuoseklumą ir pasiekiamumą, aukodamos padalijimą.

NoSQL būdingas „horizontalus“ plečiamumas, tačiau jos neturi schemos, duomenys daugumoje duomenų bazių gali būti nenuoseklūs, be to šios sistemos nepalaiko arba silpnai palaiko ACID transakcijas. NoSQL DBVS aukodamos šiuos dalykus stipriai pagerina greitaveiką. Darbui su duomenimis naudoja replikas, duomenų padalijimą. Dėl šių savybių geba operacijas vykdyti paraleliai, paskirstyti duomenis pagal tai, kurioje pasaulio vietoje jie naudojami. Mazgų pridėjimas ir palaikymas gana paprastas, dažnai automatizuotas.

Teoriškai, NewSQL duomenų bazių sistemos gali pasiekti NoSQL prilygstantį plečiamumą ir greitaveiką, net ir vykdydamos ACID transakcijas. Tai įgyvendinama vadovaujantis „shared-nothing“ architektūra, duomenis laikant operatyviojoje atmintyje ar SSD diskuose. Vis dėlto, galimas spartos praradimas, jei duomenų bazė suprojektuota nekorektiškai (pvz. virtuali atmintis perpildo realią), arba vykdomos transakcijos, kurios apima keletą serverių.

1 lentelė. Pagal plečiamumą ir duomenų organizavimą nagrinėtų DBVS apibendrinta lentelė.

DBVS	**Rūšis**	**Antriniai indeksai**	**Konk. op.**	**Laikmena**	**Replikos**	**Transakcijos**	**Map-reduce**
Voldemort	NoSQL „raktas-reikšmė“	Nėra	Užrakinimas	RAM/ BDB	Asinchroniškos	BASE	Ne
Riak	NoSQL „raktas-reikšmė“	Ryšiai tarp duomenų	MVCC	RAM/ Diskas	Asinchroniškos	BASE	Taip
Redis	NoSQL „raktas-reikšmė“	Nėra	Užrakinimas	RAM	Asinchroniškos	BASE	Ne
Scalaris	NoSQL „raktas-reikšmė“	Nėra	Užrakinimas	RAM	Sinchroniškos	ACID	Taip
SimpleDB	NoSQL dokumentų	Yra	Nėra	S3	Asinchroniškos	BASE	Ne
CouchDB	NoSQL dokumentų	Yra	MVCC	Diskas	Asinchroniškos	BASE	Taip
MongoDB	NoSQL dokumentų	Yra	Užrakinimas	Diskas	Asinchroniškos	BASE	Taip
Terrastore	NoSQL dokumentų	Yra	Užrakinimas	RAM	Sinchroniškos	BASE	Taip

DBVS	Rūšis	Antriniai indeksai	Konk. op.	Laikmena	Replikos	Transakcijos	Map-reduce
HBase	NoSQL stulpelių	Nėra	Užrakinimas	HDFS	Asinchroniškos	Ne ACID, nuoseklūs duomenys	Taip
HyperTable	NoSQL stulpelių	Dalinai	MVCC	HDFS	Sinchroniškas	Ne ACID, nuoseklūs duomenys	Taip
Cassandra	NoSQL stulpelių	Dalinai	MVCC	Diskas	Asinchroniškos	BASE	Taip
MySQL Cluster	NewSQL	Yra	Užrakinimas	Diskas/ RAM	Sinchroniškos	ACID	Ne
VoltDB	NewSQL	Yra	Laiko štampai	RAM	Sinchroniškos	ACID	Iš dalies
Clustrix	NewSQL	Yra	Užrakinimas	Diskas (SSD)	Sinchroniškos	ACID	Ne
NuoDB	NewSQL	Taip	MVCC	Diskas	Sinchroniškos	ACID	Ne

5. OPERACIJŲ SPARTA

NoSQL ir NewSQL duomenų bazių valdymo sistemų pasirinkti sprendimai, susiję su plečiamumu ir duomenų organizavimu padeda pasiekti didesnę spartą dirbant su dideliais duomenimis, bent jau taip turėtų būti. Klausimas kyla, ar iš tiesų taip yra. Atsakymui į šį klausimą rasti pasitelkiami 3 autorių ar jų grupių tyrimai, kuriuose vykdomi eksperimentai, siekiant nustatyti kaip sparčiai atliekamos operacijos naudojant skirtingas DBVS. Aptariant tyrimus kartu pademonstruota, kokiais metodais gali būti lyginama jų sparta.

5.1. Pirmasis tyrimas

Pirmasis nagrinėjamas tyrimas aprašytas straipsnyje „Solving Big Data Challenges for Enterprise Application Performance Management“ [RSJG+12] šešių autorių grupės. Autoriai pasitelkia metodą, kurio principas yra ištirti, kaip sparčiai vykdomos operacijos (op./s) ir koks jų gaišties laikas, vykdydami operacijas tam tikrais scenarijais (nurodyta žemiau) bei žiūrint, kaip šie rodikliai keičiasi naudojant daugiau nei vieną serverį. Nagrinėjamos šešios populiarios atviro kodo duomenų bazės valdymo sistemos: keturios – NoSQL ir dvi – NewSQL. NoSQL atstovės: „Apache Cassandra“, „Apache HBase“, „Project Voldemort“ ir „Redis“. NewSQL atstovės: „VoltDB“ ir „MySQL Cluster“. Tyrimas aktualus tuo, kad eksperimentiškai palyginamos duomenų bazės, kurios buvo aptartos ankstesniame skyriuje.

Duomenų bazių sistemų sparta testuojama naudojant „Yahoo! Cloud Serving Benchmark“ (YCBS) įrankį. Tai karkasas skirtas įvertinti rakto-reikšmės duomenų saugyklas. Jo pagalba galima sugeneruoti apkrovą, kuri apibrėžiama kaip CRUD (liet. sukurti, skaityti, atnaujinti, ištrinti) operacijų paskirstymas aibei įrašų. [RSJG+12]

Pagrindinis tyrimo matas – įvykdytų operacijų skaičius per nustatytą laiko tarpą. Pasirinkta laiko trukmė bandymui – 600 sekundžių. Nagrinėjami penki bandymų scenarijai (žr. lentelę žemiau), apimantys įvairius atvejus, kaip gali būti naudojama duomenų bazių sistema. Kiekvienas bandymas atliekamas po 3 kartus ir išvedamas vidurkis.

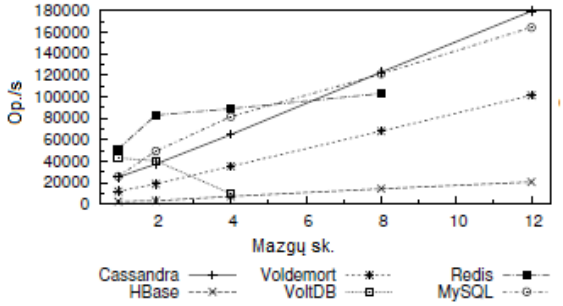
2 lentelė. Tyrime nagrinėjamų DBVS operacijų vykdymo scenarijai.

Bandymo identifikatorius	Skaitymo operacijų skaičius (%)	Skenavimo operacijų skaičius (%)	Įterpimo operacijų skaičius (%)
R (read)	95	0	5
RW	50	0	50
W (write)	1	0	99
RS	47	47	6
RSW	25	25	50

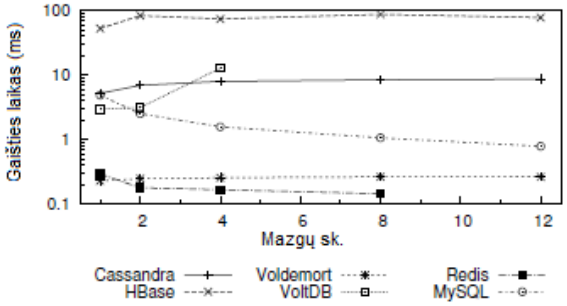
Duomenys sudaryti iš 75 baitų įrašų. Įrašą sudaro raktas, sudarytas iš skaičių ir raidžių ir esantis 25 baitų ilgio, bei 5 reikšmės laukai, kurių kiekvieno ilgis po 10 baitų.

Bandymams atlikti naudojami 2 klasteriai: klasteris M, skirtas darbui su duomenimis, kai jie laikomi atmintyje, ir klasteris D, skirtas darbui su duomenimis, kai jie yra kietuosiuose diskuose. Šiame darbe nagrinėjami bandymai tik su pirmuoju klasteriu, kadangi jį naudojant nagrinėjamos visos 6 duomenų bazių sistemos (kitu klasteriu nagrinėjamos tik 3). Klasterio M specifikacija – 16 Linux mazgų, kurių kiekvienas turi 2 Intel Xeon 4 branduolių procesorius, 16 GB RAM atminties, 2 74 GB talpos kietuosius diskus. Serveriai sujungti vienu Gigabit Ethernet komutatoriumi.

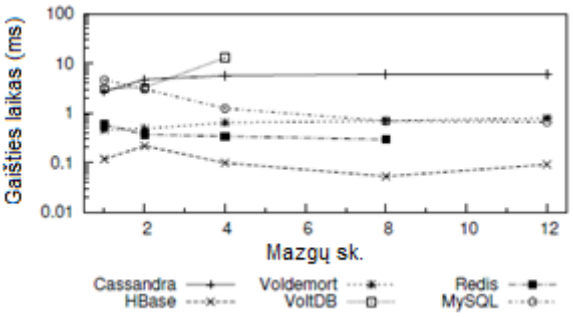
5.1.1. Bandymas R



6 pav. Operacijų sparta. Šaltinis: [RSJG+12].



7 pav. Gaišties laikas skaitymo op. Šaltinis: [RSJG+12].



8 pav. Gaišties laikas rašymo op. Šaltinis [RSJG+12].

Šis bandymas yra labiausiai koncentruotas į skaitymo operacijas (95% skaitymo operacijų, 5% rašymo). Panašus duomenų bazių sistemos veikimas būdingas daugumai socialinės informacijos svetainių, kur skaitymo operacijos užima didžiausią dalį. [RSJG+12]

Kaip matoma iš grafiko, itin didelę spartą esant vienam mazgui pademonstravo „Redis“ (virš 50000 op./s) ir „VoltDB“. Tai rodo, kad jei duomenų padalijimo nėra, šios duomenų bazių sistemos veikia greičiausiai. Dvigubai lėčiau veikė „Cassandra“ ir „MySQL Cluster“. Dar dvigubai lėčiau –„Voldemort“. Mažiausią spartą naudojant vieną serverį pademonstravo „HBase“.

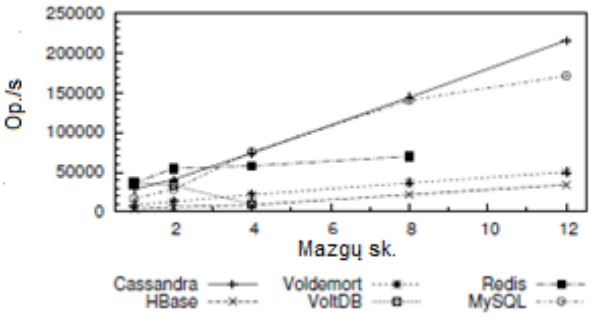
Pastebima, kad „Cassandra“, „Voldemort“ ir „HBase“ sparta didėja tolygiai didinant mazgų skaičių. Naudojant 12 mazgų, sparčiausiai operacijas atliko „Cassandra“ ir „MySQL Cluster“.

Sunkumų iškilo atliekant bandymus su „VoltDB“ ir „Redis“. Nors „VoltDB“ parodė itin aukštą spartą esant vienam mazgui, padidinus mazgų skaičių, sparta sumažėjo. Tai veikiausiai galima paaiškinti netinkamu konfigūravimu YCBS įrankiui [RSJG+12]. Tai bus pademonstruota kitame poskyryje. „Redis“ savo ruožtu nepavyko pritaikyti integruoto padalijimo (tam tikslui panaudota „Jedis“ biblioteka), tačiau laukiamos spartos gauti nepavyko (taip pat kaip ir išnaudoti visų 12 mazgų).

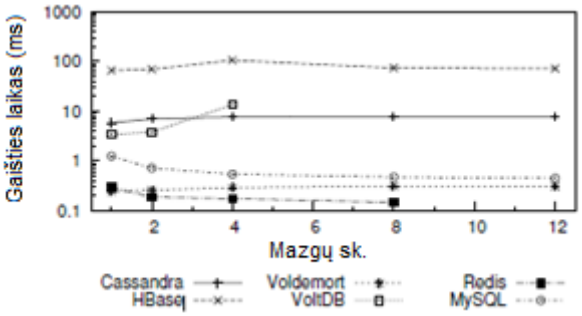
Duomenų bazių sistemų gaišties laikas skaitymo operacijoms didinant mazgų skaičių išlieka gana stabilus, visgi pati trukmė skiriasi. Nors „Cassandra“ operacijos atliekamos sparčiausiai, jai būdingas kiek didesnis gaišties laikas lyginant su kitomis sistemomis. Atitinkamai lėčiausiai veikusi „HBase“ pademonstravo ir didžiausią gaišties laiką. Geriausią rezultatą pademonstravo „Redis“.

Rašymo operacijų gaišties laikuose atsiranda kiek didesnių skirtumų. Didžiausias jis „Cassandra“ DBVS. Visgi lyginant su skaitymo operacijomis, jis lieka gana stabilus (apie 10ms). Įdomu, kad „HBase“ gaišties laikas čia jau mažiausias. Akivaizdu, kad ši duomenų bazė aukoja skaitymo operacijas, kad sumažintų gaišties laiką rašymo operacijoms.

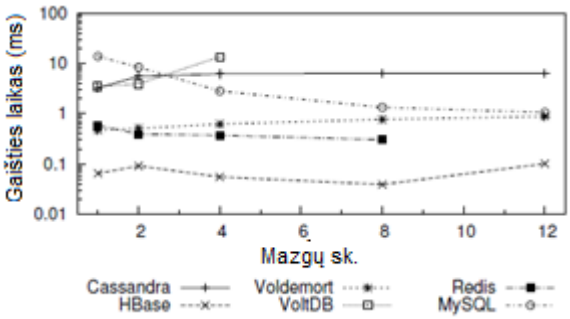
5.1.2. Bandymas RW



9 pav. Operacijų sparta. Šaltinis: [RSJG+12].



10 pav. Skaitymo op. gaišties laikas. Šaltinis: [RSJG+12].



11 pav. Rašymo op. gaišties laikas. Šaltinis: [RSJG+12].

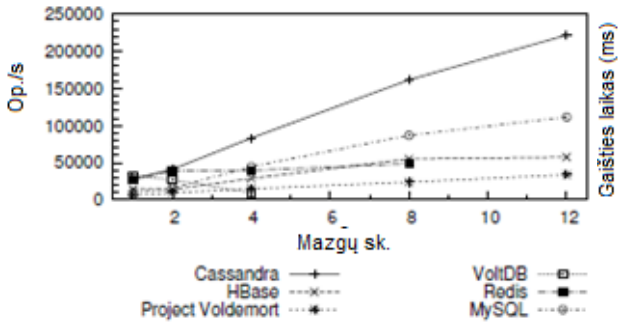
Bandymo operacijos sudarytos iš 50% skaitymo ir 50% rašymo operacijų. Tai apibūdinama kaip sistema, turinti didelę rašymo operacijų apkrovą [RSJG+12].

Didžiausią spartą vienam mazgui pasiekė „VoltDB“, kurios sparta tik šiek tiek sumažėjo lyginant su ankstesniu bandymu. „Redis“ – antroje vietoje, kurios sparta lyginant su ankstesniu bandymu sumažėjo apie 20%. „Cassandra“ sparta padidėjo apie 10%, o „HBase“ net apie 40%. Savo ruožtu „Voldemort“ sparta sumažėjo 33%.

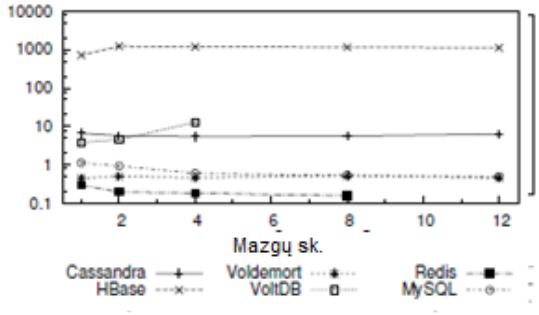
Didinant mazgų skaičių vėl pastebimas tolygus spartos didėjimas „Cassandra“, „Voldemort“ ir „HBase“ duomenų bazių sistemoms. „MySQL Cluster“ iki 8 mazgų sparta gali lygintis su „Cassandra“, tačiau vėliau jos spartos augimas mažėja. Sparčiausia ir šiame bandyme jau minėta „Cassandra“.

Kalbant apie gaišties laiką skaitymo operacijoms, jis išlieka daugmaž toks pat kaip bandyme R. Ryškesnis skirtumas matomas tik „MySQL Cluster“, kur laikas gana stipriai sumažėjo. Taip pat ir rašymo operacijoms, laikas išlieka panašus kaip bandyme R, išskyrus „HBase“, kurios laikas sumažėja daugmaž perpus ir „MySQL Cluster“, kurios gaišties laikas išauga dvigubai.

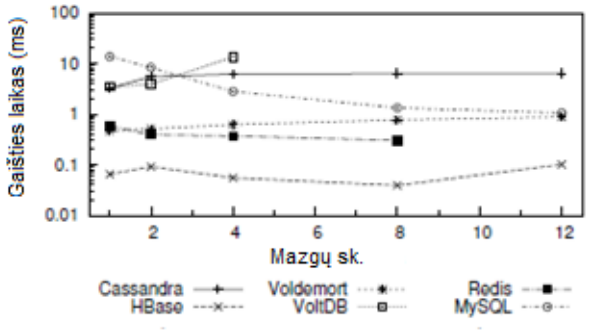
5.1.3. Bandymas W



12 pav. Operacijų atlikimo sparta. Šaltinis: [RSJG+12].



13 pav. Skaitymo op. gaišties laikas. Šaltinis [RSJG+12].



14 pav. Rašymo op. gaišties laikas. Šaltinis: [RSJG+12].

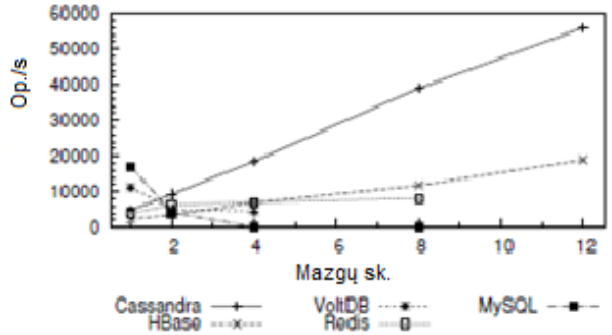
Šio bandymo operacijas sudaro 99% rašymo operacijų ir 1% skaitymo.

Bandymo rezultatai panašūs į RW. Skirtumas tas, kad visos duomenų bazių sistemos parodė kiek sumažėjusią spartą išskyrus „Cassandra“ (2% prieaugis esant 12 mazgų) ir „HBase“, kurios sparta padidėjo kiek ženkliau, kas patvirtina šių duomenų bazių efektyvumą atliekant rašymo operacijas.

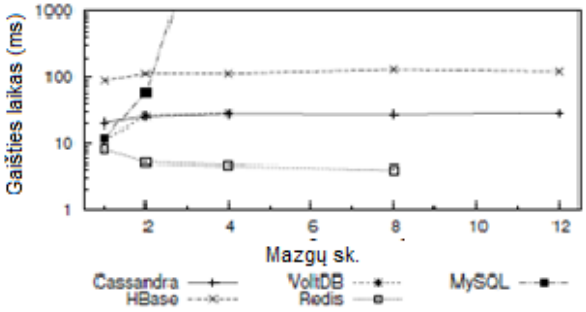
Didžiausi pasikeitimai skaitymo gaišties laikuose matomi „HBase“, kur jo reikšmė siekia net 1 s ir „Voldemort“, kur laikas lyginant su ankstesniais bandymais padvigubėja.

Rašymo gaišties laikas visose duomenų bazių valdymo sistemose kiek padidėjo, išskyrus „Voldemort“, kur jis liko panašus į ankstesnių bandymų rezultatus. Drastiškai padidėjo „HBase“ rezultatas (20 kartų), tačiau vis tiek reikšmė mažiausia iš visų DBVS.

5.1.4. Bandymas RS



15 pav. Operacijų sparta. Šaltinis: [RSJG+12]



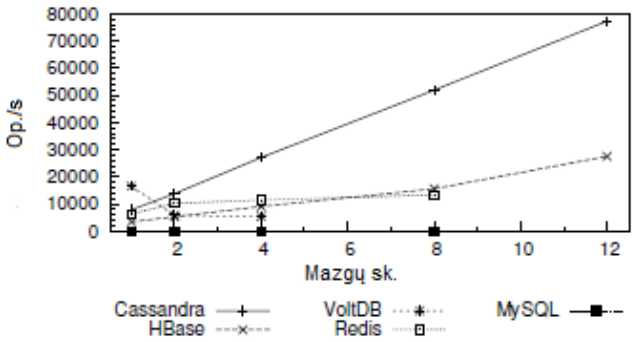
16 pav. Skaitymo op. gaišties laikas. Šaltinis: [RSJG+12]

Šio bandymo operacijas sudarytos iš 47 % skaitymo ir skenavimo bei 6 % rašymo operacijų. Kadangi naudotas YCBS papildinys skirtas „Voldemort“ DBVS nepalaikė skenavimo operacijų, sistema nebuvo tiriama.

Šį kartą geriausią rezultatą operacijų spartos aspektu vienam mazgui pasiekė „MySQL Cluster“. Visgi sparta drastiškai krito naudojant daugiau nei vieną mazgą. Tai rodo, kad ši duomenų bazių sistema neveikia gerai, kai duomenys padalinti ir vykdomos skenavimo operacijos. Visgi tai taip pat galima paaiškinti ir nevisiškai tinkamu YCBS pritaikymu. Tas pats pastebima ir „VoltDB“, tačiau tai matoma viso tyrimo metu, todėl priežastis galimai kita, kas vėliau atsispindės kitame tyrime. „Cassandra“ ir „HBase“ vėl pastebimas tolygus spartos didėjimas didinant mazgų skaičių. Kaip ir ankstesniuose bandymuose, taip ir čia „Cassandra“ veikia sparčiausiai.

Didžiulis gaišties laikas „MySQL Cluster“ didinant mazgų skaičių patvirtina, kad skenavimo operacijos šioje duomenų bazėje itin lėtos. Kitų DBVS laikas stabilus, iš kurių mažiausias – „Redis“.

5.1.5. Bandymas RSW



17 pav. Operacijų sparta. Šaltinis: [RSJG+12].

Bandymo metu vykdomos operacijos: 25 % skaitymo, 25 % skenavimo, 50 % rašymo.

Rezultatai yra panašūs į bandymo RS rezultatus, išskyrus „MySQL Cluster“. Šios duomenų bazių sistemos sparta itin maža, vos 20 op./s, o pridėjus daugiau mazgų – dar mažesnė. Savo ruožtu, „Cassandra“ ir „HBase“ operacijas atlieka dvigubai sparčiau, lyginant su RS bandymu. Tai galima paaiškinti sumažėjusiu skenavimo operacijų skaičiumi. „VoltDB“ pasiekia didžiausią spartą esant vienam mazgui.

5.1.6. Rezultatai ir išvados

Sparčiausia praktiškai visuose bandymuose naudojant padalijimą per kelis mazgus buvo „Cassandra“. Jai taip pat būdingas tolygus spartos didėjimas pridedant naujus mazgus. Spartai pasiekti veikiausiai padeda naudojamas MVCC mechanizmas (vietoj „užrakinimo“), o plečiamumas automatizuotas. „Cassandra“ kiek sparčiau veikė vykdant rašymo operacijas, o ne skaitymo. Tai taip pat galima paaiškinti tuo, kad šios operacijos stulpeliuose yra atominės. Visgi, gaišties laikas sąlyginai didelis.

„HBase“ – kita stulpeliais grįsta NoSQL duomenų bazių valdymo sistema, priešingai nei „Cassandra“ parodė mažiausią spartą. Vienas aspektų, dėl ko sparta lėtesnė – „užrakinimų“ naudojimas. Vis dėl to, jos plečiamumas taip pat efektyvus – didinant mazgų skaičių tolygiai didėja ir sparta. „HBase“ pademonstravo gana mažą gaišties laiką rašymo operacijoms, ypač kai skaitymo operacijų intensyvumas aukštas. Visgi gaišties laikas skaitymo operacijoms buvo kur kas didesnis nei kitoms DBVS.

Kaip ir „Cassandra“ bei „HBase“ „Voldemort“ taip pat parodė tolygų spartos augimą plečiant sistemą. Taip pat, ši DBVS pademonstravo žemą bei gana tolygų gaišties laiką pridedant naujus mazgus. Spartos požiūriu ši duomenų bazių valdymo sistema buvo tarp anksčiau minėtų duomenų bazių valdymo sistemų atstovių.

NewSQL atstovė „MySQL Cluster“ parodė gerą spartą, kai nenaudojamos skenavimo operacijos. Gerą spartą galima paaiškinti operatyviosios atminties naudojimu saugant duomenis. Skenavimo atveju, ši duomenų bazė dirbo itin lėtai (didinant mazgų skaičių pasiekdama sekundės ir didesnį gaišties laiką). Visgi tai gali būti paaiškinama nebūtinai pačios sistemos yda, bet ir YCBS netinkamu suderinamumu. Galima pastebėti, kad ši duomenų bazė nėra taip gerai plečiama – pridedant mazgų matoma, kad gautas spartos prieaugis mažėja.

„Redis“ parodė labai gerus rezultatus esant vienam mazgui. Vis dėlto iškilo sunkumų siekiant padalinti duomenis pridedant naujų mazgų (tam pasitelkta biblioteka „Jedis“), kas galimai įtakojo ne itin aukštą spartos padidėjimą. Taip pat šiai duomenų bazių valdymo sistemai nepavyko išnaudoti visų 12 mazgų. Aukštą spartą veikiausiai pasiekiama dėl operatyviosios atminties naudojimo duomenims saugoti. Taip pat ši duomenų bazių sistema parodė žemą gaišties laiką.

Didžiausi sunkumai iškilo tiriant „VoltDB“. Ši duomenų bazių valdymo sistema parodė labai gerus spartos rezultatus esant vienam mazgui, tačiau nepavyko gauti jokio spartos išaugimo pridėjus naujų mazgų.

Pasitelkdami metodą, kuriuo tiriama DBVS operacijų vykdymo sparta bei gaišties laikas skirtingais veikimo scenarijais bei esant skirtingam mazgų skaičiui autoriai pademonstravo, kokios sistemos, kokias atvejais yra sparčiausios. Šiam tikslui pasiekti panaudotas YCBS įrankis, generavęs duomenis bei apkrovą, kuris padėjo pasiekti aiškius rezultatus, nepaisant tam tikrų problemų kai kurioms duomenų bazių valdymo sistemoms.

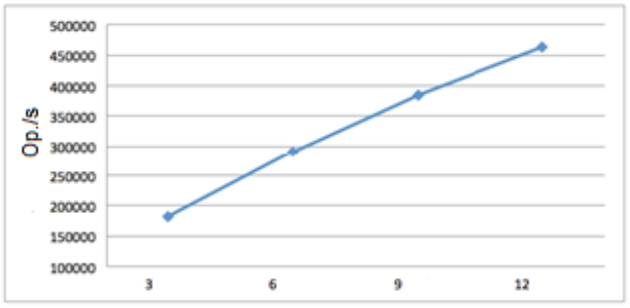
5.2. „VoltDB“ tyrimas

Antrasis tyrimas atliktas vieno iš „VoltDB“ komandos narių A. Rogerso kaip atsakas anksčiau nagrinėtam tyrimui, aprašytas straipsnyje „VoltDB in-memory database achieves best-in-class results, running in the cloud, on the YCSB Benchmark“ [Rog14]. Jame pademonstruota, kad „VoltDB“ yra gerai plečiama ir gali rodyti labai gerus spartos rodiklius.

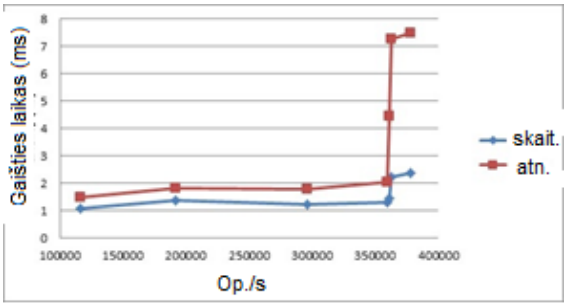
Testams atlikti naudotas tas pats YCBS įrankis. Atlikti 3 bandymai naudojant standartines YCBS apkrovas: A – 50 % skaitymo, 50 % atnaujinimo operacijų, B – 95 % skaitymo, 5 % atnaujinimo operacijų, E – 95 % skenavimo, 5 % įterpimo operacijų.

Bandymams atlikti panaudotos virtualios mašinos, kurios veikė serveryje, kurio parametrai: Intel Xeon procesoriai (viso 32 branduoliai), 2 SSD diskai po 320 GB atminties, ir kiekvienai virtualiai mašinai skirta 60 GB operatyvios atminties.

5.2.1. Bandymas A



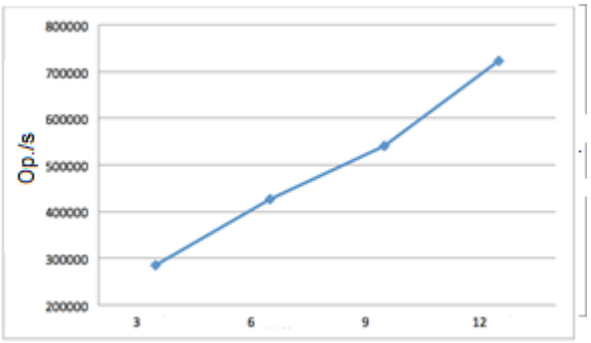
18 pav. Operacijų sparta pagal mazgų sk. [Rog14].



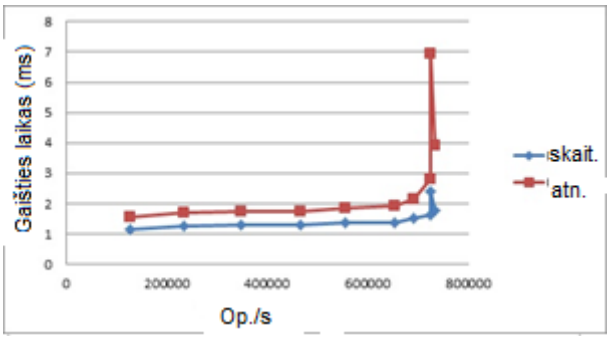
19 pav. Gaišties laikas pagal op./s. [Rog14].

Kaip matoma iš pirmos diagramos, „VoltDB“ pasiekė didelę spartą (12 mazgų net virš 450000 op./s). Spartos didėjimas pridedant naujų mazgų beveik visiškai tolygus. Antroje diagramoje matyti kaip kinta gaišties laikas priklausomai nuo operacijų vykdymo spartos. Pastebima, kad laikas gana mažas iki kol pasiekiama 35000 op./s – tada jis stipriai išauga. Tai ypač pastebima atnaujinimo operacijų gaišties laiko pasikeitime.

5.2.2. Bandymas B



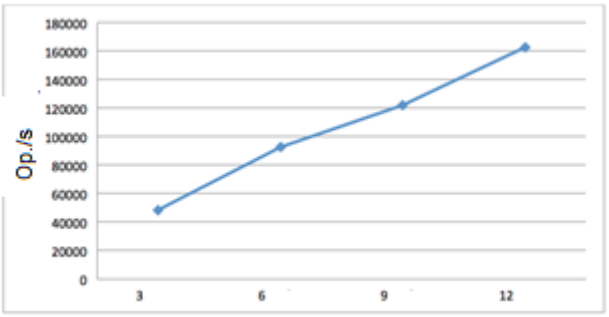
20 pav. Operacijų sparta pagal mazgų sk. Šaltinis: [Rog14].



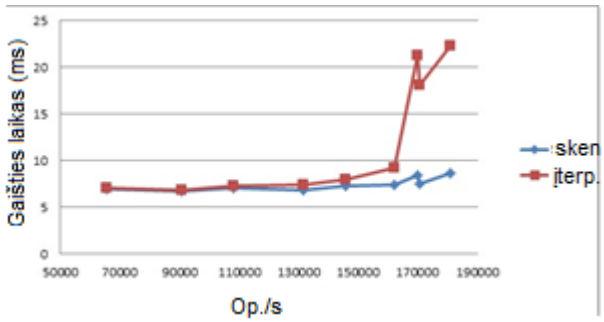
21 pav. Gaišties laikas pagal op./s. Šaltinis: [Rog14].

Antrajame bandyme dominuoja skaitymo operacijos (95 %). Iš grafiko matoma, kad dėl šios priežasties stipriai išauga operacijų sparta. Kaip ir pirmame bandyme taip ir šiame pastebimas beveik tolygus spartos augimas pridedant naujų mazgų. Kalbant apie gaišties laiką, didelių skirtumų lyginant su bandymu A nepastebima.

5.2.3. Bandymas E



22 pav. Operacijų sparta pagal mazgų sk. Šaltinis: [Rog14].



23 pav. Gaišties laikas pagal op./s. Šaltinis: [Rog14].

Bandyme E dominuoja skenavimo operacijos. Kaip matoma iš grafiko, „VoltDB“ atlikdama šias operacijas veikia kur kas lėčiau lyginant su kitais bandymais. Visgi ir čia pastebimas gana tolygus spartos augimas pridedant naujų mazgų. Gaišties laikas savo ruožtu didesnis nei kituose bandymuose. Taip pat pastebima tendencija, kad ir čia pasiekus maksimalią operacijų spartą, stipriai išauga gaišties laikas.

5.2.4. Apibendrinimas

Šie eksperimentai parodo, kad „VoltDB“ taip pat sėkmingai kaip ir NoSQL duomenų bazių sistemos gali būti plečiama. Sistemos parodyti rezultatai tiek spartos, tiek gaišties laiko požiūriu atrodo geri. Sparčiausiai „VoltDB“ dirbo atlikdama skaitymo operacijas, o lėčiausiai – skenavimo.

5.3. Trečiasis tyrimas

Trečiasis nagrinėjamas tyrimas aprašytas straipsnyje „A performance comparison of SQL and NoSQL databases“ [LM13], atliktas mokslininkų iš Oklando universiteto Naujojoje Zelandijoje. Tyrimas šiek tiek skirtingas nuo ankstesnių – jame pasitelkiamas metodas, kuriuo nagrinėjama ne tai, kaip duomenų bazių sistemos veikia plečiamos, o kiek laiko užtrunka atlikti tam tikrą skaičių operacijų. Nagrinėjamos 7 duomenų bazių sistemos – NoSQL atstovės „MongoDB“, „RavenDB“, „CouchDB“, „Cassandra“, „Hypertable“, „Couchbase“, ir SQL atstovė – „Microsoft SQL Server Express“.

Šiame darbe nagrinėjamos 4 tyrime vykdytos CRUD operacijos trimis atvejais:

- Skaitymas, t. y. rakto-reikšmės poroje reikšmės perskaitymas pagal nurodytą raktą,
- Rašymas, kuris apima C ir U operacijas (įterpimą ir atnaujinimą), t. y. jei raktas naujas, įterpiama reikšmė, o jei toks raktas jau egzistuoja, reikšmė atnaujinama,
- Trynimas, t. y. rakto-reikšmės poros ištrynimas pagal nurodytą raktą.

5.3.1. Skaitymas

3 lentelė. Skaitymo op. atlikimo laikas. Šaltinis: [LM13].

Duomenų bazė	Operacijų skaičius					
	10	50	100	1000	10000	100000
MongoDB	8	14	23	138	1085	10201
RavenDB	140	351	539	4730	47459	426505
CouchDB	23	101	196	1819	19508	176098
Cassandra	115	230	354	2385	19758	228096
Hypertable	60	83	103	420	3427	63036
Couchbase	15	22	23	86	811	7244
MS SQL Express	13	23	46	277	1968	17214

Lentelėje pavaizduoti skaitymo operacijų rezultatai – kiek laiko (ms) reikėjo, kad būtų įvykdytas nurodytas skaičius operacijų. Geriausius rezultatus parodė „Couchbase“. Panašius rezultatus parodė ir „MongoDB“ bei „MS SQL Express“, ypač esant mažesniam operacijų skaičiui. Lėčiausiai tiek atliekant mažai, tiek daug operacijų veikė „RavenDB“, lyginant su „Couchbase“ atliekant 100000 operacijų, net 6 kartus lėčiau.

Nors „MongoDB“ ir „Couchbase“ esant dideliam operacijų skaičiui veikė apytiksliai dvigubai sparčiau už SQL DBVS atstovę, verta pastebėti, kad kitos NoSQL duomenų bazės parodė kur kas prastesnius rezultatus.

5.3.2. Rašymas

4 lentelė. Rašymo op. atlikimo laikas. Šaltinis: [LM13].

Duomenų bazė	Operacijų skaičius					
	10	50	100	1000	10000	100000
MongoDB	61	75	84	387	2693	23354
RavenDB	570	898	1213	6939	71343	740450
CouchDB	90	374	616	6211	67216	932038
Cassandra	117	160	212	1200	9801	88197
Hypertable	55	90	184	1035	10938	114872
Couchbase	60	76	63	142	936	8492
MS SQL Express	30	94	129	1790	15588	216479

Antrojo tyrimo rezultatai vėl pademonstravo, kad „Couchbase“ ir šias operacijas atlieka sparčiausiai. Įdomu pastebėti, kad jos sparta beveik nesulėtėjo, lyginant su „MongoDB“ ar ypač su „CouchDB“ ir „MS SQL Express“. Savo ruožtu „Cassandra“ patvirtino, kad gali greitai vykdyti rašymo operacijas, pademonstruodama trečią rezultatą ir lyginant su skaitymo operacijomis esant dideliam skaičiui pagreitėdama net 2,5 karto. Nors SQL atstovė stipriai sulėtėjo, vis tiek veikė greičiau nei „RavenDB“ ir „CouchDB“.

5.3.3. Trynimas

5 lentelė. Trynimo op. atlikimo laikas. Šaltinis: [LM13].

Duomenų bazė	Operacijų skaičius					
	10	50	100	1000	10000	100000
MongoDB	4	15	29	235	2115	18688
RavenDB	90	499	809	8342	87562	799409
CouchDB	71	260	597	5945	67952	705684
Cassandra	33	95	130	1061	9230	83694
Hypertable	19	63	110	1001	10324	130858
Couchbase	6	12	14	81	805	7634
MS SQL Express	11	32	57	360	3571	32741

Šiame tyrime skirtinai nuo anksčiau nagrinėtų tiriama ir ištrynimo operacijų sparta. Kaip ir ankstesniuose bandymuose stabilumą išlaikė ir didžiausią spartą parodė „Couchbase“ DBVS. „MongoDB“ vėl parodė antrą geriausią rezultatą, o „MS SQL Server“ sugrįžo į trečią vietą. „Cassandra“ vėl parodė panašius rezultatus kaip ir atliekant rašymo operacijas, kas rodo akivaizdų jos polinkį į duomenų modifikavimo operacijas. Atitinkamai „RavenDB“ ir „CouchDB“ vėl parodė gana prastus rezultatus.

5.3.4. Apibendrinimas

Metodas, naudotas šiame tyrime padėjo parodyti, kad jei lyginamos operacijos nepadalijant duomenų ir neplečiant sistemų, SQL duomenų bazių sistemos gali veikti labai greitai lyginant su NoSQL. Šiuo atveju „Microsoft SQL Server Express“ DBVS parodė trečią greičiausią rezultatą atliekant skaitymo ir trynimo operacijas bei 5 rezultatą rašant. Įdomu, kad kai kurios DBVS kaip „CouchDB“ ar „RavenDB“ pademonstravo stipriai prastesnius rezultatus.

Atliekant bandymus akivaizdus „Couchbase“ dominavimas, kuri veikė žymiai sparčiau nei likusios duomenų bazių valdymo sistemos. Be to, ši DBVS pademonstravo gana stabilius ir nekintančius rezultatus, nesvarbu kokios operacijos buvo atliekamos.

Verta atkreipti dėmesį, kad „Cassandra“ kaip ir pirmame tyrime pademonstravo geresnius rezultatus atliekant rašymo operacijas lyginant su skaitymo. Visgi čia jau ji veikė lėčiau už kai kurias kitas DBVS. Viena iš greitesnių – „MongoDB“. Nors ši duomenų bazių valdymo sistema duomenų saugojimui nenaudoja operatyviosios atminties, tačiau duomenys yra indeksuoti, palaikomos atominės operacijos, kas veikiausiai ir lemia gerą greitaveiką.

Sudėtinga paaiškinti tokią prastą „CouchDB“ spartą. Vienas aspektų, dalinai įtakojęs prastą spartą galėtų būti tai, kad duomenų indeksavimas jose nėra automatiškas, todėl atliekant bandymus duomenys buvo neindeksuoti ir savaime suprantama lėčiau randami.

6. PRAKTINIS CRUD OPERACIJŲ SPARTOS PALYGINIMAS

Be minėtų mokslininkų grupių atliktų tyrimų, darbe atliktas praktinis eksperimentas, kurio metu naudotas dar kitoks greitaveikos tyrimo metodas – išmatuotas vidutinis CRUD operacijų atlikimo laikas pasirinktoms duomenų bazių sistemoms, tiriant konkrečias užklausas, o ne generuotą apkrovą. Nagrinėjamos trys DBVS – SQL atstovė „Microsoft SQL Server“, NoSQL atstovė „MongoDB“ ir NewSQL atstovė „NuoDB“.

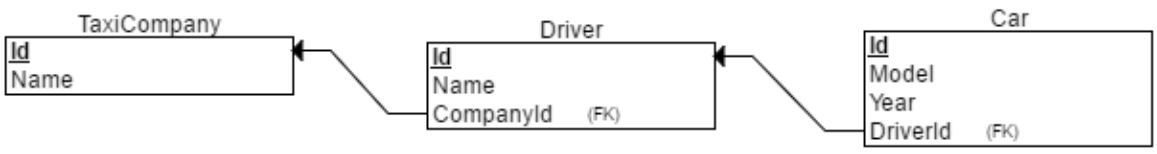
Serverio, kuriame atliktas tyrimas, parametrai:

- Operacinė sistema „Windows 10 Education“,
- Procesorius „Intel Core i7-3612QM“,
- 8 GB operatyviosios atminties,
- 500 talpos GB kietasis diskas.

Bandymams atlikti parašyta programa naudojant .NET karkasą, kuria kreipiamasi į minėtas duomenų bazių valdymo sistemas. Programos pagalba matuojama, kiek laiko atliekamos nagrinėjamos CRUD operacijos.

„Microsoft SQL Server“ ir „MongoDB“ pasirinktos todėl, nes jos nagrinėtos ankstesniuose skyriuose, kur pademonstravo gerus rezultatus. Taip pat tai reiškia, kad galima palyginti gautus rezultatus su ankstesniais tyrimais. NewSQL atveju, dėl tos priežasties planuota nagrinėti „VoltDB“ duomenų bazių valdymo sistemą, tačiau jai reikalingas serveris, kuris naudoja „Linux“ operacinę sistemą. Todėl pasirinkta „NuoDB“.

Eksperimentui atlikti, sugalvotas pavyzdys – duomenys apie taksi, t. y. įmones, vairuotojus, automobilius. Tai SQL ir NewSQL atstovių atveju atitinka lenteles (žr. reliacinę schemą žemiau), o „MongoDB“ – kolekcijas. Tiriamų SQL ir NewSQL duomenų bazių duomenys saugomi 32 bitų sveikųjų skaičių formatu (identifikatoriai, automobilio metai, išoriniai raktai) bei 60 simbolių ilgio eilučių formatu (kiti nepaminėti laukai). „MongoDB“ duomenys saugomi BSON formatu.



24 pav. Taksi duomenų bazės reliacinė schema.

Eksperimento metu nagrinėti 6 CRUD operacijų atvejai:

1. Duomenų įterpimas,
2. Duomenų atnaujinimas pagal raktą,
3. Duomenų trynimas pagal raktą,

4. Duomenų skaitymas pagal raktą,
5. Duomenų skaitymas pagal atributą,
6. Duomenų skaitymas pagal atributus apimant kelis duomenų rinkinius.

Kiekvienam atvejui išnagrinėti atlikta po 3 bandymus. Bandymų metu atlikta po 1010 operacijų. Kiekvienos operacijos spartos vidurkis gaunamas atmetus pirmąsias 10 kiekvieno bandymo operacijų (t. y. tam, kad būtų atmestas pirmųjų operacijų atlikimo laikas, kuris yra galimai lėtesnis dėl prisijungimo prie duomenų bazės ar kitų priežasčių).

6.1. Duomenų įterpimas

Pirmoji nagrinėjama operacija – įterpimas. Bandymo metu įterpiami duomenys apie 10 taksi kompanijų, 500 vairuotojų (kiekvienai įmonei po 50) ir 500 automobilių (kiekvienam vairuotojui po vieną). Gauti rezultatai pademonstruoti lentelėje žemiau.

6 lentelė. Vidutinis duomenų įterpimo operacijos vykdymo laikas.

Microsoft SQL Server	MongoDB	NuoDB
4,86 ms	0,21 ms	31,62 ms

Matoma, kad kaip ir ankstesniame skyriuje nagrinėtame tyrime, taip ir čia „MongoDB“ parodė itin aukštą spartą bei trumpiausią laiką įterpiant duomenis. Ji įterpimo operaciją vidutiniškai atliko net 23 kartus greičiau nei „Microsoft SQL Server“. Lyginant su ankstesniu tyrimu pastebimas didžiulis skirtumas. Vis dėl to, nors tai paaiškinti gana sudėtinga, tačiau pastebima ta pati tendencija, kad NoSQL atstovė veikė sparčiau. Verta pastebėti, kad itin žemą spartą parodė „NuoDB“, kas rodo, kad bent jau nesant duomenų padalijimui, ji įterpia duomenis gana lėtai, lyginant su kitomis DBVS.

6.2. Duomenų atnaujinimas

Kita nagrinėjama operacija – duomenų atnaujinimas pagal nurodytą raktą. Atliekamos užklausos trims duomenų grupėms paduodant atsitiktinį raktą iš jau egzistuojančių. Gauti rezultatai pateikti lentelėje žemiau.

7 lentelė. Vidutinis duomenų atnaujinimo operacijos vykdymo laikas.

Microsoft SQL Server	MongoDB	NuoDB
4,8 ms	0,31 ms	30,37 ms

Rezultatai lyginant su įterpimu skiriasi nežymiai. Vėl patį geriausią vidutinišką operacijos atlikimo laiką pasiekė „MongoDB“. „NuoDB“ pagreitėjo maždaug 1 ms, tačiau laikas vis tiek gana prastas lyginant su kitomis tirtomis duomenų bazių valdymo sistemomis.

6.3. Duomenų trynimas

Trečioji nagrinėta operacija – duomenų trynimas pagal nurodytą raktą. Gauti rezultatai parodyti lentelėje žemiau.

8 lentelė. Vidutinis duomenų trynimo operacijos vykdymo laikas.

Microsoft SQL Server	MongoDB	NuoDB
4,96 ms	0,29 ms	29,7 ms

Gauti rezultatai vėl panašūs į ankstesniuose dviejuose bandymuose matytus rezultatus – eilės tvarka vėl išsirikiuoja „MongoDB“, „Microsoft SQL Server“ ir „NuoDB“.

6.4. Duomenų skaitymas nr. 1

Ketvirtoji nagrinėta operacija – duomenų skaitymas nurodant pirminį raktą. Gauti rezultatai pateikiami lentelėje žemiau.

9 lentelė. Vidutinis duomenų skaitymo nr. 1 operacijos vykdymo laikas.

Microsoft SQL Server	MongoDB	NuoDB
4,32 ms	0,25 ms	0,28 ms

Šį kartą gauti rezultatai iš esmės skiriasi. „NuoDB“ veikusi itin lėtai atlikdama duomenų modifikavimo operacijas pademonstravo beveik niekuo nenusileidžiantį „MongoDB“ vidutinį operacijos atlikimo laiką, t. y. operacijos atliekamos daugiau nei 100 kartų greičiau. Tai rodo, kad ši DBVS didžiausią dėmesį skiria duomenų skaitymui, o ne jų modifikavimui. Šiek tiek mažesnį laiką lyginant su ankstesniais bandymais parodė „Microsoft SQL Server“, tačiau pokytis nedidelis.

6.5. Duomenų skaitymas nr. 2

Penktoji tyrime nagrinėta operacija – duomenų skaitymas pagal atributus. Konkrečiau – taksi įmonių duomenys buvo gaunami pagal panašų vardą naudojant reguliarias išraiškas, vairuotojų duomenys – nurodant įmonės raktą, automobilių duomenys – nurodant modelį taip pat pagal reguliarias išraiškas. Gauti rezultatai pateikti lentelėje žemiau.

10 lentelė. Vidutinis duomenų skaitymo nr. 2 operacijos vykdymo laikas.

Microsoft SQL Server	MongoDB	NuoDB
4,61 ms	0,53 ms	1,03 ms

Pastebima, kad „MongoDB“ ir „NuoDB“ operacijas atliko dvigubai lėčiau, kadangi buvo ieškomi duomenys nenaudojant indeksuotų laukų. Savo ruožtu „Microsoft SQL Server“ sulėtėjo

nežymiai. Vis dėlto, ji vis tiek veikė kur kas lėčiau nei kitos dvi nagrinėjamos duomenų bazių valdymo sistemos.

6.6. Duomenų skaitymas nr. 3

Šeštoji nagrinėta operacija – duomenų skaitymas pagal atributus apimant kelių rinkinių (lentelių ar kolekcijų) duomenis. Konkrečiau, randami visi nurodytos įmonės vairuotojai, kurių automobilis yra naujesnis nei nurodyti metai. Kadangi „MongoDB“ nesukurta tam, kad atliktų skaitymo operacijas keliose kolekcijose iš karto, gauti atitinkamus rezultatus jai atliktos dvi skaitymo operacijos – pirma randami įmonės vairuotojai, po to – automobiliai, tenkinantys metų sąlygą. Reiškia, skaičiuojant vidurkį, „MongoDB“ operacijų atlikimo laiko dalinys dvigubai didesnis nei kitoms DBVS, o dalmuo toks pat.

11 lentelė. Vidutinis duomenų skaitymo nr. 3 operacijos vykdymo laikas.

Microsoft SQL Server	MongoDB	NuoDB
7,15 ms	0,65 ms	33,8 ms

Kaip ir buvo galima tikėtis, visos duomenų bazių sistemos parodė didesnius vidutinius operacijų atlikimo laikus. Vis dėl to, jei SQL ir NoSQL atstovės sulėtėjo ne itin smarkiai, „NuoDB“ kaip ir atlikdama įterpimo, atnaujinimo ar trynimo operacijas, vėl parodė gana prastą vidutinį operacijos atlikimo laiką. Tai rodo, kad ši DBVS ne itin greitai veikia, kai reikalingas duomenų apjungimas ir išrinkimas iš kelių lentelių.

6.7. Apibendrinimas

Apibendrinus visus bandymus galima teigti, kad sparčiausiai operacijas atliko „MongoDB“. Taip pat pastebima, kad jos užfiksuoti atlikimo laikai yra stabilūs ir panašūs visose atliktose operacijose. Tai patvirtina faktą, kad NoSQL gali pasiekti labai aukštą spartą – o šiuo atveju, net ir nesant duomenų padalijimui. Lėčiau už „MongoDB“ (virš 20 kartų) pasirodė „Microsoft SQL Server“. Jos atliekamų operacijų vidutinis laikas taip pat gana stabilus nepriklausomai nuo operacijos pobūdžio. Lėčiausiai įterpimo, atnaujinimo ir ištrynimo operacijas atliko „NuoDB“, stipriai nusileidusi tiek SQL, tiek NoSQL atstovėms. Vis dėl to, kiek netikėtus lyginant su modifikavimo operacijomis rezultatus ši DBVS parodė atlikdama skaitymo operacijas, kur ji veikė tik šiek tiek lėčiau už „MongoDB“ bei kur kas greičiau už „Microsoft SQL Server“. Iš to galima spręsti, kad ši duomenų bazių valdymo sistema „aukoja“ duomenų modifikavimo operacijų spartą siekdama kuo geresnės skaitymo operacijų spartos. Vis dėl to reikia pastebėti, kad „NuoDB“ labai sparčiai atliko tik paprastas skaitymo operacijas pagal raktą ar kitus atributus, tačiau parodė gana

prastą laiką, kai reikalingas lentelių apjungimas, kas rodo, kad sistema suprojektuota ne šiam tikslui.

Praktiniam tyrimui pasirinktas metodas papildo anksčiau nagrinėtus tyrimus, parodydamas, kaip sparčiai gali būti vykdomos konkrečios CRUD operacijos. Kadangi vykdoma ne generuota apkrova, o iš anksto apibrėžtos užklausos, lengva pastebėti, kokiais atvejais tirtos DBVS veikia sparčiau. Pavaizduota, kokios galimos nagrinėtų SQL, NoSQL ir NewSQL duomenų bazių valdymo sistemų stiprybės: „MongoDB“ – itin aukšta sparta ir stabilumas nepriklausomai nuo operacijos pobūdžio, „MS SQL Server“ – stabilumas ir gana aukšta sparta lyginant su NewSQL atstove, „NuoDB“ – itin aukšta sparta skaitant reikšmes pagal raktą ar kitus atributus. Taip pat silpnybės: „MongoDB“ – didesnio operacijų skaičiaus poreikis norint apjungti kolekcijas, „NuoDB“ – žema sparta atliekant modifikavimo operacijas ar skaitant duomenis apjungiant lenteles.

REZULTATAI IR IŠVADOS

Gauti darbo rezultatai:

1. Aptarta, kas yra SQL, NoSQL ir NewSQL duomenų bazių sistemos. Nurodytos pagrindinės jų savybės, esminiai skirtumai lyginant tarpusavyje. Remiantis db-engines.com sudarytu populiarumo reitingu, nustatytos rinkoje populiariausios duomenų bazių valdymo sistemų atstovės.
2. Sudarytas kriterijų rinkinys, apimantis esmines duomenų bazių sistemų savybes bei leidžiantis palyginti duomenų bazių sistemas bei įvertinti, kaip jų pasirinkti sprendimai gali įtakoti greitaveiką.
3. Pagal sudarytą kriterijų rinkinį, įvertintos ir palygintos pasirinktos SQL, NoSQL, NewSQL duomenų bazių sistemos. Aptarta, kaip tam tikri jų pasirinkti sprendimai gali įtakoti greitaveiką. Nagrinėjant konkrečias DBVS atstoves parodyta, kaip naudojant šį kriterijų rinkinį galima tirti bet kokią kitą duomenų bazių valdymo sistemą.
4. Nagrinėjant kitų autorių atliktus eksperimentus bei darbe vykdytą eksperimentą išskirti metodai, kuriais galima tirti DBVS greitaveiką bei pavaizduota, kaip tai atlikti. Nustatyta, kokios DBVS kokiais atvejais veikia sparčiausiai. Pateikti samprotavimai, kodėl gauti tokie spartos rezultatai. Taip pat tarpusavyje palyginti skirtingų tyrimų rezultatai, aptarta, kuo nagrinėti metodai skiriasi.
5. Atliekant eksperimentą, parašyta programa bei sukurtos trys duomenų bazės. Naudojant programą gauti tirtų DBVS CRUD operacijų vykdymo laikai. Iš vykdymo laikų apskaičiuoti vidurkiai, kurie buvo palyginti bei įvertinti.

Darbo metu sudarytas kriterijų rinkinys buvo panaudotas tiriant įvairias duomenų bazių valdymo sistemas. Juo remiantis galima nustatyti, kokiais naudojimo atvejais, kuri gali būti efektyvesnė. Apibendrinus darbo rezultatus gautus lyginant ir vertinant DBVS pagal skirtingus kriterijus galima daryti tokias išvadas:

1. SQL duomenų bazės palaiko ACID transakcijas, tačiau dėl šios priežasties aukoja duomenų padalijimo galimybę. NewSQL palaiko ACID transakcijas kartu įgyvendindama ir duomenų padalijimą. Tam pasiekti pasitelkiami įvairūs architektūriniai sprendimai. NoSQL dažniausiai nepalaiko ACID transakcijų, dėl to, kad būtų geriau plečiamos. Šios sistemos dažnai remiasi BASE principais, tad jose duomenys ne visada nuoseklūs.
2. SQL duomenų bazės plečiamos tik „vertikaliai“, nes nėra duomenų padalijimo. NoSQL ir NewSQL būdingas „horizontalus“ plečiamumas.
3. NoSQL ir NewSQL sprendimai sudaro galimybes pagerinti greitaveiką įgyvendinant paralelių operacijų vykdymą, duomenų padalijimą, geografinį duomenų paskirstymą, replikų naudojimą, operatyviosios atminties naudojimą duomenims saugoti, MVCC ar

„optimistinio užrakinimo“ naudojimą konkuruojančioms operacijoms spręsti. NoSQL greitaveiką gerina ir ACID atsisakymu.

4. SQL ir NewSQL duomenys saugomi lentelėse, todėl galimas jų apjungimas. Taip pat, jose duomenis galima indeksuoti ne tik pagal pirminius laukus, kas paspartina paiešką ieškant ne pagal raktą. Antriniai indeksai palaikomi ne visose NoSQL sistemose.
5. NoSQL duomenų saugojimo formatas įvairus, todėl skirtingais atvejais efektyviau naudoti skirtingas sistemas:
   - Siekiant gauti tam tikro atributo duomenų aibę tinkama stulpelių DBVS,
   - Norint greitai pasiekti nedidelę porciją duomenų, pravartu naudoti rakto-reikšmės DBVS,
   - Saugojant duomenis tokiu formatu kaip XML, JSON ar pan. patogu naudoti dokumentų duomenų bazių valdymo sistemas,
   - Grafų duomenų bazės patogios siekiant pereiti reikšmės.
6. Nagrinėjant DBVS pagal metodą, kurio esmė išmatuoti DBVS spartą ir operacijų gaišties laiką, vykdant operacijas skirtingais apkrovos scenarijais bei naudojant skirtingą mazgų skaičių, gauta, kad NoSQL atstovių „Cassandra“, „Voldemort“ ir „HBase“ plečiamumas efektyvus ir tolygus. Tolygus plečiamumas parodytas ir „VoltDB“ atveju. Labai didelę spartą pademonstravo „Cassandra“ ir „VoltDB“. Kita NewSQL atstovė „MySQL Cluster“ taip pat parodė aukštus rezultatus, tačiau jos plečiamumas ne toks tolygus.
7. Kitas metodas, pagal kurį nagrinėtos DBVS remiasi laiko kiekiu tam tikram CRUD operacijų skaičiui atlikti. Šiuo metodu atliktas tyrimas parodė, kad NoSQL išties gali pasiekti itin aukštą spartą, iš kurių sparčiausios nagrinėtos – „Couchbase“ ir „MongoDB“. Vis dėl to, kai kurios NoSQL atstovės veikė gana lėtai ir kur kas lėčiau už nagrinėtą SQL atstovę.
8. Trečias metodas, kurį pasitelkiant atliktas eksperimentas, remiasi konkrečių CRUD operacijų vykdymu bei vidutinių vykdymo laikų nustatymu. Eksperimentas patvirtino didelę „MongoDB“ spartą. Taip pat parodyta, kaip nagrinėtų DBVS sparta skiriasi priklausomai nuo operacijos pobūdžio. NewSQL atstovės atveju sparta drastiškai kinta skirtingais atvejais – skaitant pasiekiami rezultatai nedaug skyrėsi nuo „MongoDB“, tačiau rašant – kur kas prastesni už SQL atstovės rezultatus. Tad pastebima, kad kai kurios DBVS aukoja spartą atlikdomas vienos rūšies operacijas, kad sparčiau atliktų kitas.

ŠALTINIAI

[ASF16] The Apache Software Foundation. Apache Cassandra 2.0 Product documentation. [Žiūrėta 2016-05-24]. Prieiga per internetą: <http://docs.datastax.com/en/cassandra/2.0/cassandra/gettingStartedCassandraIntro.html>

[Bas16] Basho Technologies, Inc. Complex Query Support. [Žiūrėta 2016-05-24]. Prieiga per internetą: <http://basho.com/products/riak-kv/complex-query-support/>

[Cat10] R. Cattell. Scalable SQL and NoSQL Data Stores. [Žiūrėta 2016-04-12]. Prieiga per internetą: <http://www.cattell.net/datastores/Datastores.pdf>

[Clu16] Clustrix, Inc. Clustrix Documentation. Concurrency Control. [Žiūrėta 2016-05-24]. Prieiga per internetą: <http://docs.clustrix.com/display/CLXDOC/Concurrency+Control>

[DBE15] DB-Engines. DB-Engines Ranking. [Žiūrėta 2016-05-23]. Prieiga per internetą: <http://db-engines.com/en/ranking>.

[Goo16] Google, Inc. terrastore – Developers_Guide.wiki. [Žiūrėta 2016-05-24]. Prieiga per internetą: <https://code.google.com/archive/p/terrastore/wikis/Developers_Guide.wiki>

[HJ11] R. Hecht, S. Jablonski. NoSQL Evaluation. [Žiūrėta 2016-04-13]. Prieiga per internetą: <http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6138544&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D6138544>.

[Ima16] Imanuel. 13 NewSQL Databases. 2016. [Žiūrėta 2016-05-24]. Prieiga per internetą: <http://www.predictiveanalyticstoday.com/newsql-databases/>

[LM13] Y. Li, S. Manoharan. A performance comparison of SQL and NoSQL databases. [Žiūrėta 2016-05-03]. Prieiga per internetą: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6625441>.

[MH13] A B M Moniruzzaman, S. A. Hossain. NoSQL Database: New Era of Databases for Big data Analytics - Classification, Characteristics and Comparison. [Žiūrėta 2016-05-11]. Prieiga per internetą: <http://arxiv.org/ftp/arxiv/papers/1307/1307.0191.pdf>.

[Nuo16] NuoDB, Inc. NuoDB Documentation. [Žiūrėta 2016-05-24]. Prieiga per internetą: <http://doc.nuodb.com/Latest/Default.htm>.

[PT08] Paper Trail. Consistency and availability in Amazon's Dynamo. [Žiūrėta 2016-05-24]. Prieiga per internetą: <http://the-paper-trail.org/blog/consistency-and-availability-in-amazons-dynamo/>.

[Pok14] J. Pokorný. How to Store and Process Big Data: Are Today's Databases Sufficient? Knygoje: K. Saeed, V. Snášel. Computer Information Systems and Industrial Management. CISIM, Ho Chi Minh City, 2014, p. 5 – 10.

[Rog14] A. Rogers. VoltDB in-memory database achieves best-in-class results, running in the cloud, on the YCSB Benchmark. [Žiūrėtas 2016-05-24]. Prieiga per internetą: <https://voltdb.com/blog/voltdb-memory-database-achieves-best-class-results-running-cloud-ycsb-benchmark>.

[RSJG+12] T. Rabl, M. Sadoghi, H. Jacobsen, S. Gomez-Villamor, V. Muntes-Mulero, S. Mankovskii. Solving Big Data Challenges for Enterprise Application Performance Management. [Žiūrėta 2016-06-02]. Prieiga per internetą: <http://vldb.org/pvldb/vol5/p1724_tilmannrabl_vldb2012.pdf>.

[RW12] E. Redmond, J. R. Wilson. Seven Databases in Seven Weeks. A Guide to Modern Databases and the NoSQL Movement. Pragmatic Bookshelf, Raleigh, 2012. p. 3, 317

[UC14] B. Upadrasta, A. Chungath. NoSQL, NewSQL, or RDBMS: How To Choose. [Žiūrėta 2016-03-11]. Prieiga per internetą: <http://www.informationweek.com/big-data/big-data-analytics/nosql-newsql-or-rdbms-how-to-choose/a/d-id/1297861>.

[Vol16] VoltDB, Inc. VoltDB Documentation. [Žiūrėta 2016-05-24]. Prieiga per internetą: <https://docs.voltdb.com/UsingVoltDB/IntroHowVoltDBWorks.php>.

[ZIB16] Zuse Institute Berlin. Users and Developers Guide. Scalaris. [Žiūrėta 2016-05-24.]. Prieiga per internetą: <https://github.com/scalaris-team/scalaris/blob/master/user-dev-guide/main.pdf>.

SANTRUMPOS

ACID – angl. „Atomicity, Consistency, Isolation, Durability“, liet. vientisumas, darna, izoliuotumas, patvarumas. Tai parametrų rinkinys, užtikrinantis, kad duomenų bazių transakcijos įvykdomos patikimai.

BASE – angl. „Basically available, Soft-state, Eventualy consistent“. Tai transakcijų vykdymo principas, kuris reiškia, kad duomenų bazė neturi tradicinių transakcijų mechanizmo, o apribojimai pritaikomi duomenų modeliui, siekiant gauti geriau padalijimui pritaikytas schemas.

BSON – angl. „Binary JSON“. Tai apsikeitimo duomenimis formatas.

CAP – angl. „Consistency, Availability, Partitioning“, liet. darna, pasiekiamumas, padalijimas. Tai teorema, teigianti, kad kompiuterinė sistema gali užtikrinti tik 2 iš šių 3-jų minėtų dalykų.

CRUD – angl. „Create, Read, Update, Delete“. Tai operacijų rinkinys, reiškiantis duomenų sukūrimą, skaitymą, atnaujinimą ir trynimą.

DB – duomenų bazė.

DBVS – duomenų bazių valdymo sistema.

HDD – angl. „Hard Drive Disk“, liet. kietasis diskas.

YCBS – „Yahoo! Cloud Serving Benchmark“ – karkasas skirtas įvertinti rakto-reikšmės duomenų saugyklas.

JSON – angl. „Javascript Object Notation“. Tai duomenų apsikeitimo formatas, paremtas žmogui skaitomu tekstu ir susidedantis iš atributų bei jų reikšmių.

MS – „Microsoft“ organizacija.

MVCC – angl. „Multi-version Concurrency Control“. Tai būdas spręsti „konkuruojančių“ operacijų problemą, kurio principas toks, kad vartotojas mato tam tikro laiko momentu esančią duomenų bazės būseną ir pakeitimus mato tik įvykus transakcijai.

NoSQL – angl. „Not only SQL“. Duomenų bazių rūšis, pasižyminti didesniu lankstumu už SQL.

RAM – angl. „Random Access Memory“, liet. operatyvioji atmintis.

SQL – angl. „Structured Query Language“. Tai užklausų kalba skirta rašyti užklausas SQL duomenų bazėms.

SSD – angl. „Solid State Drive“. Tai duomenų laikymo įrenginys.

XML – angl. „Extensible Markup Language“ yra duomenų aprašymo kalba