# Didelių duomenų skaidymo našumo analizė rakto-reikšmės ir reliacinėse duomenų bazėse

**Dominykas Černovas, Vasilij Savin**

Vilniaus universitetas, Matematikos ir informatikos fakultetas, Informatikos institutas, Didlaukio g. 47, Vilnius, Lietuva
*dominykas.cernovas@mif.stud.vu.lt, vasilij.savin@mif.vu.lt*

**Santrauka.** Didėjant e. komercijos sistemų apkrovoms, duomenų bazių našumas ir mastelio plėtimas tampa kritiniu veiksniu užtikrinant sklandžią naudotojų patirtį. Šiame darbe sprendžiama problema – kuri duomenų bazių valdymo sistema yra efektyvesnė aukšto srauto scenarijuose: rakto–reikšmės tipo *Redis Cluster* ar atmintyje veikianti reliacinė *PostgreSQL* su Citus plėtiniu. Tyrimo rezultatai rodo, kad *Redis Cluster* pasiekia žymiai didesnį pralaidumą net ir tada, kai *PostgreSQL* veikia vienodomis atminties sąlygomis, tačiau *PostgreSQL* pasižymi stabilesne delsa esant mažesnėms apkrovoms.

**Raktiniai žodžiai:** duomenų skaidymas, sharding, Redis Cluster, PostgreSQL, Citus, JMeter.

## 1 Įvadas

Augant didžiųjų duomenų apimtims, tradicinės duomenų bazių architektūros susiduria su našumo iššūkiais. Duomenų skaidymas tampa vienu iš esminių metodų, leidžiančiu horizontaliai plėsti sistemas ir užtikrinti stabilų jų veikimą. Šio tyrimo objektas – *Redis Cluster* ir atmintyje veikiančio, išskaidyto *PostgreSQL* palyginimas realiame pirkinių krepšelio scenarijuje. Nors *Redis Cluster* ir *PostgreSQL* našumo tyrimai yra paplitę mokslinėje literatūroje, išskaidytų šių sistemų tiesioginio palyginimo rasti nepavyko. Todėl šis tyrimas užpildo šią spragą pateikdamas pagrįstą abiejų architektūrų įvertinimą identiškomis sąlygomis.

Mokslinėje literatūroje duomenų bazių valdymo sistemų našumo tema yra plačiai nagrinėjama. Almeida ir kt. [4] tyrimas parodė, kad Redis ir Memcached NoSQL duomenų bazės pasižymi mažesne delsa ir geresniu pralaidumu. Tačiau šiame darbe buvo lyginamos atmintyje veikiančios rakto–reikšmės duomenų bazės (Redis ir Memcached) su tradicinėmis diską


DOI: https://doi.org/10.15388/LMITT.2026.7

naudojančiomis reliacinėmis duomenų bazėmis. Dėl esminio duomenų saugojimo terpės (RAM ir diskas) skirtumų toks palyginimas gali neatspindėti vien tik duomenų bazių valdymo sistemų architektūros skirtumų našumui.

Basalla ir kt. [5] nustatė, kad padidėjusi sistemos delsa e. komercijos platformose tiesiogiai koreliuoja su naudotojų pirkinių krepšelio apleidimu, patvirtindami, jog našumo optimizavimas šiame kontekste turi praktinę reikšmę. Todėl šiame darbe siekiama palyginti sistemas vienodomis atminties sąlygomis, kad architektūros poveikis būtų įvertintas tiksliau.

**Tyrimo tikslas** - praktiškai įvertinti ir palyginti *PostgreSQL* bei *Redis Cluster* našumą vykdant pirkinių krepšelio operacijas.

## 2 Teorinis pagrindimas

Pagrindinis skirtumas tarp reliacinės ir NoSQL duomenų bazės yra tai, kad reliacinėje duomenų bazėje duomenys yra saugomi iš anksto aprašytoje lentelėje, kuri turi savo struktūrą ir pirminį (unikalų) raktą. Rakto–reikšmės (angl. *key-value*) duomenų bazės renka, gauna ir saugo duomenis kaip rakto ir reikšmės porų grupes, o kiekvienas duomenų įrašas yra pažymėtas unikaliu raktu ir susijusia reikšme. Šis duomenų bazės tipas dažnai taikomas sistemose, kur būtina greitai apdoroti didelį duomenų kiekį trumpalaikėse užklausose, pavyzdžiui, elektroninės prekybos krepšelių valdymui.

Reliacinės duomenų bazės turi griežtai laikytis ACID transakcijų, kurios užtikrina duomenų vientisumą ir patikimumą, tačiau kiekviena operacija turi būti užregistruota, kas gali sumažinti atsako laiką kai operacijų skaičius yra didelis [6]. Tuo tarpu Redis atsisako ACID transakcijų ir remiasi BASE principu: pagrindiniu prieinamumu, minkšta būsena ir galutiniu nuoseklumu. Kadangi Redis leidžia naudoti atmintyje (angl. *in-memory*) laikomus duomenis, jų gavimas trunka milisekundes [7].

Sistemų plėtimo galimybes apibrėžia AKF skalės kubas (angl. *AKF Scale Cube*) [8], kuris suteikia pagrindą suvokimui, kaip didelio masto sistemos gali būti plečiamos trimis kryptimis. Modelis išskiria X ašį (horizontalus dubliavimas), Y ašį (funkcinė dekompozicija) ir **Z ašį (duomenų skaidymas)**, kuri apibrėžia duomenų skaidymą pagal įvairius duomenų segmentus.

Atsižvelgiant į šiuos teorinius principus ir siekiant praktiškai įvertinti abiejų technologijų našumą, tolesnėje darbo dalyje aprašomas eksperimentas, kuriame lyginamas *Redis Cluster* ir *PostgreSQL* pralaidumas bei delsa simuliuojant pirkinių krepšelio scenarijų.

# 3 Eksperimento aprašymas

## 3.1 Eksperimento metodika

Pradinėje tyrimo fazėje buvo sukurta .NET 10 API sąsaja, turėjusi suvienodinti prieigą prie Redis ir *PostgreSQL*. Tačiau atliekant pirminius testus su didelėmis apkrovomis (virš 1000 lygiagrečių gijų), nustatyta, kad aplikacijos sluoksnis tapo sistemos „butelio kakleliu“ (angl. *bottleneck*). API nespėdavo apdoroti visų gaunamų užklausų, todėl duomenų bazių resursai likdavo neišnaudoti, o gauti rezultatai neatspindėjo realaus duomenų bazės potencialo.

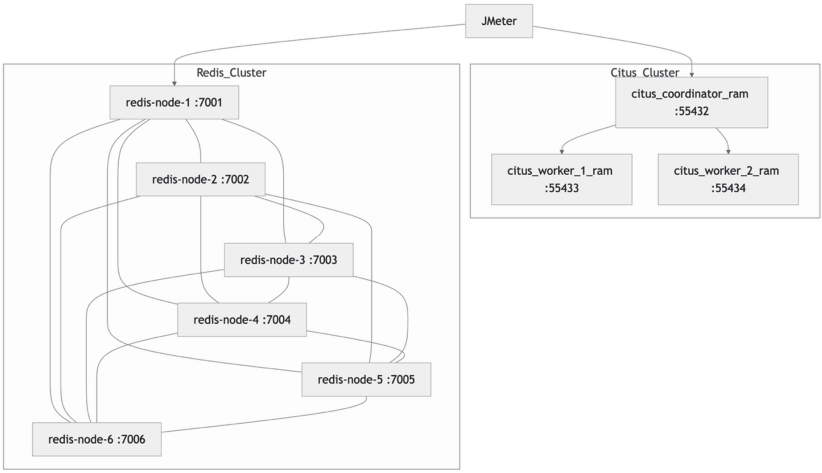
Siekiant eliminuoti šį ribojimą ir išmatuoti maksimalų duomenų bazių pajėgumą, eksperimento architektūra buvo pakeista:

1. **Tiesioginis kreipimasis.** Atsisakyta tarpinio API sluoksnio, JMeter (5.6.3 versija) [3] testavimo įrankį jungiant tiesiogiai prie duomenų bazių *Groovy* programavimo kalbos pagalba.
2. **Optimizavimas.** Užklausų vykdymui buvo naudojamas *JMeter JSR223 Sampler* komponentai su *Groovy* programavimo kalba. Norint išgauti maksimalų našumą ir minimalų testavimo įrankių paklaidą, *Groovy* programavimo kalba parašyti komponentai buvo kešuojami panaudojant JMeter suteikiama „*Compilation Cache*“ galimybe. Tai leido išvengti perkompiliavimo kiekvienos užklausos metu, taip užtikrinant, kad JMeter netaptų nauju ribojančiu faktoriumi.

## 3.2 Infrastruktūra ir duomenų bazių konfigūracija

Eksperimento vykdymo aplinka buvo parengta naudojant OpenNebula virtualizacijos platformą. Duomenų bazės buvo leidžiamos virtualiose mašinose (VM), naudojančių Ubuntu 24.04 LTS operacinę sistemą. Abi duomenų bazės buvo sukonfigūruotos dirbti klasterio režimu Docker (29.2.1 versija) konteineriuose, naudojant RAM atmintį duomenų saugojimui. Docker konfigūracijos schemą galima matyti 1 pav. Docker infrastruktūra

- ***PostgreSQL* (16 versija) (Citus:12.1).** Buvo naudojamas vienas koordinatoriaus mazgas ir keli darbiniai mazgai. Konfigūracijoje nustatytas padidintas maksimalių jungčių skaičius (*max_connections=300*) **i**r išjungtas priverstinis sinchronizavimas (*fsync=off*), optimizuojant rašymo operacijas. [2]
- ***Redis Cluster* (8.6 versija).** Sukonfigūruotas 6 mazgų klasteris. Mazgai tarpusavyje bendravo per vidinį Docker tinklą.




**1 pav.** Docker infrastruktūra.

## 3.3 Užklausų generavimas

Eksperimento metu abiem sistemoms buvo siunčiamos identiškos pirkinių krepšelio operacijos, kurios buvo parinktos pagal vidutinį krepšelio pirkinių skaičių vieno apsipirkimo metu. Atliekamos operacijos atspindi tipinį vartotojo elgesį formuojant ir valdant pirkinių krepšelį [1]. Krepšelio valdymo operacijos:

1. **Krepšelio inicijavimas**
   a. Krepšelio sukūrimas
2. **Prekių valdymas**
   a. Prekės pridėjimas
   b. Papildomos prekės pridėjimas
   c. Prekės kiekio atnaujinimas
   d. Prekės pašalinimas
3. **Krepšelio peržiūra**
   a. Krepšelio turinio peržiūra
4. **Pirkimo užbaigimas**
   a. Apsipirkimas (krepšelio ištrynimas po užsakymo patvirtinimo)

Siekiant visapusiškai įvertinti duomenų bazių stabilumą ir jų ribas, eksperimentas buvo suskaidytas į kelis etapus kaip pavaizduota 1 lentelėje, pa-

gal lygiagrečių naudotojų (gijų) skaičių. Kiekvienas etapas atspinti skirtingą sistemos (duomenų bazės) užimtumą, o paskutinis etapas skirtas patikrinti ar sistema sugeba atsistatyti po stresinės (kritinės) apkrovos.

**1 lentelė.** Testavimo etapai.

| Etapas | Pavadinimas | Naudotojų skaičius | Trukmė | Tikslas |
|---|---|---|---|---|
| **1.** | Žema apkrova | 200 | 5 min. | Sistemos stabilumo patikra ir bazinių našumo metrikų fiksavimas. |
| **2.** | Vidutinė apkrova | 500 | 5 min. | Normalių veikimo sąlygų simuliacija, pralaidumo stebėjimas. |
| **3.** | Aukšta apkrova | 1 000 | 10 min. | Perėjimas prie intensyvaus darbo režimo, pradedamas stebėti delsos augimas. |
| **4.** | Labai aukšta apkrova | 2 000 | 10 min. | Sistemos elgsenos stebėjimas prie didelio intensyvumo, artėjimas prie pralaidumo ribų. |
| **5.** | Ekstremali (stresinė) apkrova | 5 000 | 10 min. | Siekiama nustatyti maksimalų pralaidumą ir fiksuoti sistemos lūžio taškus bei klaidų lygį. |
| **6.** | Vidutinė apkrova | 500 | 5 min. | Stebima, kaip greitai sistema grįžta į stabilią būseną (2 etapas) po kritinės perkrovos. |

## 4 Rezultatų analizė

Eksperimento duomenys, gauti po API sluoksnio pašalinimo ir skriptų optimizavimo, atskleidė tikrąjį abiejų sistemų pralaidumą.

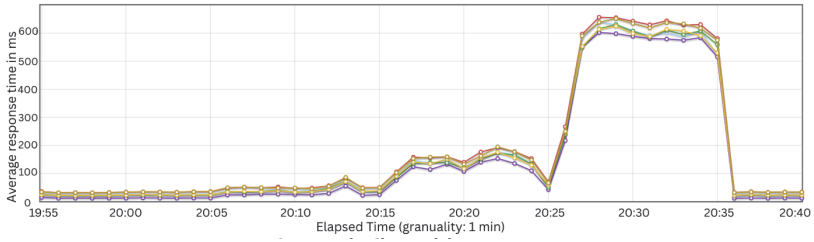
### 4.1 *Redis Cluster* – testavimo rezultatai

Pagrindiniai rezultatai:

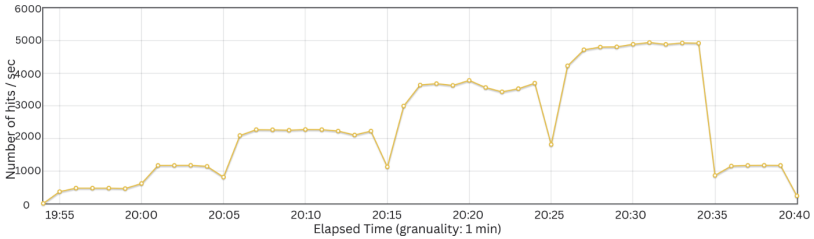
- *Redis Cluster* prisotinamas (angl. *saturated*) esant ~2 000 lygiagrečių naudotojų.
- Optimalus pralaidumas – 3 pav. matome, kad pralaidumas yra ~3 600 užklausų per sekundę (req/s).
- Pereinant nuo 2 000 iki 5 000 lygiagrečių naudotojų, pralaidumas

auga tik ~36 % (papildomai ~1 300 req/s), tačiau 2 pav. delsa padidėja ~3,5 karto – nuo ~200 ms iki ~700 ms. Tai patvirtina, kad 2 000 lygiagrečių naudotojų šiame testavimo plane yra optimalus apkrovos taškas.

- Pastebėti du staigūs pralaidumo kritimo momentai (20:15 ir 20:25), kurie greičiausiai susiję su interneto nestabilumu, JMeter šiukšlių surinkimo pauzėmis arba *Redis Cluster* perbalansavimo įvykiais.




**2 pav.** Redis Cluster delsa.



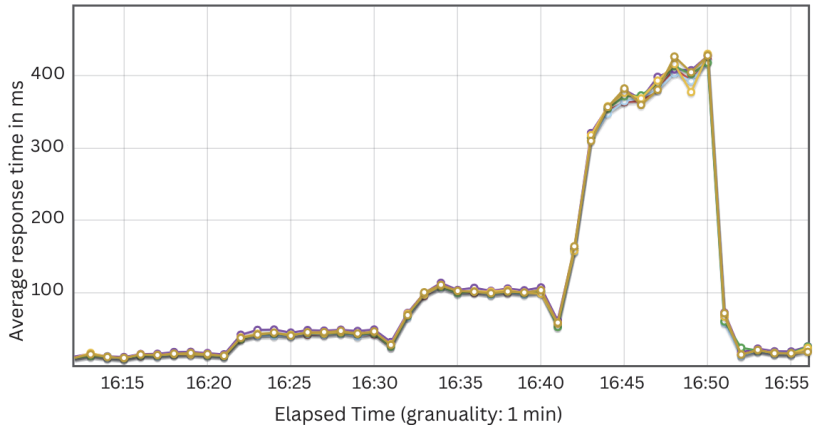

**3 pav.** Redis Cluster pralaidumas.

## 4.2 *PostgreSQL* (Citus) – testavimo rezultatai

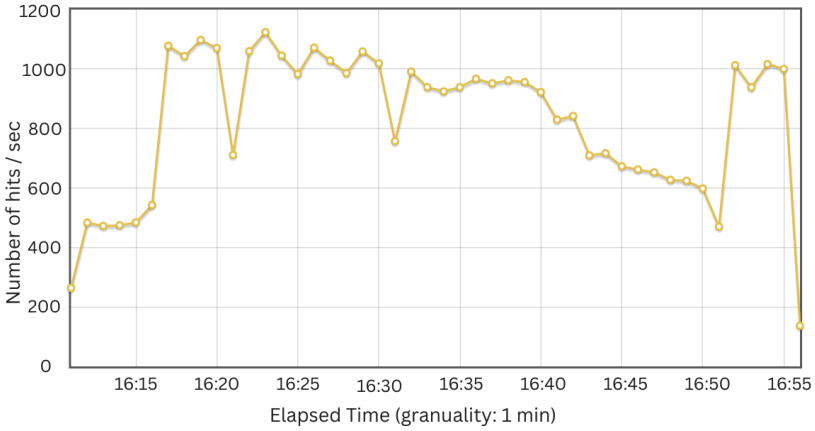
Pagrindiniai rezultatai:

- *PostgreSQL* prisotinamas (angl. *saturated*) anksčiau nei *Redis Cluster*. Jau esant 1 000 lygiagrečių naudotojų.
- Optimalus pralaidumas – 5 pav. matome, kad pralaidumas yra ~1 000 užklausų per sekundę (req/s). Tai yra tik ~28 % *Redis Cluster* pralaidumo (~3 600 req/s).
- Prie 5 000 naudotojų *PostgreSQL* išlaikė mažesnę delsą (~400 ms) 4 pav. nei *Redis Cluster* (~600 ms) 2 pav. tačiau tai buvo pasiekta žymiai mažesnio pralaidumo kaina. Sistema sugebėjo apdoroti žymiai mažiau užklausų per tą patį laiko tarpą.

- Nuo ~16:40 pralaidumas pradeda nuosekliai mažėti nuo ~950 req/s iki ~600 req/s – sistema pradeda kaupti neapdorotų užklausų eilę ir jungtys laukia atsilaisvinančių resursų.
- Kaip matoma 4 pav. po stresinės apkrovos sumažinimo iki stabilaus naudotojų kiekio *PostgreSQL* grįžo į savo bazinį lygį (~30 ms), panašiai kaip *Redis Cluster*, tai rodo gerą atsigavimo gebėjimą.




**4 pav.** PostgreSQL delsa.




**5 pav.** PostgreSQL pralaidumas.

### 4.3 Palyginimas

Atlikus abiejų sistemų apkrovos testavimą, šiame skyriuje pateikiamas apibendrintas *Redis Cluster* ir *PostgreSQL* našumo metrikų palyginimas. Žemiau esančioje 2 lentelėje pateikiama pagrindinių našumo metrikų suvestinė, apimanti užklausų skaičių ir sėkmingumą, pralaidumą ir delsą.

**2 lentelė.** Redis Cluster ir PostgreSQL našumo metrikų palyginimas.

| Metrika | Redis Cluster | PostgreSQL (Citus) |
|---|---|---|
| **Užklausų skaičius** | 6 722 964 | 2 275 603 |
| **Klaidų skaičius** | 1,53 % (103 043) | 0 % |
| **Vidutinis pralaidumas** | 2 475,41 req/s | 841,60 req/s |
| **Prisotinimo riba** | ~2 000 naudotojų | ~1 000 naudotojų |
| **Optimalus pralaidumas** | ~3 600 req/s | ~1 000 req/s |
| **Delsa (500 naudotojų)** | ~30 ms | ~30 ms |
| **Delsa (5 000 naudotojų)** | ~625 ms | ~400 ms |

## 5 Išvados

Apibendrinant atliktą *Redis Cluster* ir *PostgreSQL* našumo analizę pirkinių krepšelio scenarijuje, galima teigti, kad pasirinkta testavimo metodika leido objektyviai įvertinti abiejų sistemų architektūrinius skirtumus. Gauti rezultatai leidžia daryti šias pagrindines išvadas:

1. Atlikus eksperimentą nustatyta, kad *Redis Cluster* architektūra yra žymiai efektyvesnė aukšto pralaidumo scenarijuose net ir tada, kai palyginimas atliekamas vienodomis atminties sąlygomis – tai paneigia prielaidą, kad *PostgreSQL* našumo apribojimai kyla tik dėl diskų naudojimo, o ne dėl architektūrinių skirtumų.
2. Nustatyta, kad *PostgreSQL* delsos pranašumas ekstremalios apkrovos metu yra klaidinantis rodiklis – jis atsiranda dėl apdorojamų užklausų skaičiaus sumažėjimo, o ne dėl tikrojo sistemos efektyvumo padidėjimo. Todėl vertinant duomenų bazių našumą nerekomenduojama remtis vienu rodikliu.
3. Abiejų sistemų atsigavimo po stresinės apkrovos greitis yra panašus, kas rodo, kad nei *Redis Cluster*, nei *PostgreSQL* su Citus nėra jautresni trumpalaikiams apkrovos pakilimams – tai svarbus stabilumo kriterijus renkantis technologiją e. komercijos sistemoms.

## Literatūra


[1] Dynamic Yield. eCommerce Statistics and Benchmarks by Industry. XP² Benchmark Portal. Mastercard. 2025. [Tinkle]. Available: https://marketing.dynamicyield.com/benchmarks

[2] Citus Data. Citus 12.1 Documentation. Citus 12.1 documentation. Microsoft. 2023. [Tinkle]. Available: https://docs.citusdata.com/en/v12.1

[3] The Apache Software Foundation. Apache JMeter User Manual. JMeter documentation. The Apache Software Foundation. 2024. [Tinkle]. Available: https://jmeter.apache.org/usermanual/index.html

[4] D. Almeida, M. Lopes, L. Saraiva, M. Abbasi, P. Martins, J. Silva, P. Váz. Performance Comparison of Redis, Memcached, MySQL, and PostgreSQL: A Study on KeyValue and Relational Databases. Iš: *2023 Second International Conference On Smart Technologies For Smart Nation (SmartTechCon).* 2023, pp. 902–907. [Tinkle]. Available: https://doi.org/10.1109/SmartTechCon57526.2023.10391649

[5] M. Basalla, J. Schneider, M. Luksik, R. Jaakonmäki, J. V. Brocke. On Latency of E-Commerce Platforms. *Journal of Organizational Computing and Electronic Commerce*. 2021, vol. 31, no. 1, pp. 1-17. [Tinkle]. Available: https://doi.org/10.1080/10919392.2021.1882240

[6] M. Diogo, B. Cabral, J. Bernardino. Consistency Models of NoSQL Databases. Future Internet, vol. 11, no. 2, Art. no. 43, 2019. [Tinkle]. Available: https://doi.org/10.3390/fi11020043

[7] V. Sharma, M. Dave. Sql and nosql databases. International Journal of Advanced Research in Computer Science and Software Engineering, vol. 2, no. 8, pp. 20-27, Aug. 2012. [Tinkle].

[8] M. L. Abbott, M. T. Fisher. The AKF Scale Cube. AKF Partners tinklaraštis. [Tinkle]. Available: https://akfpartners.com/growth-blog/scale-cube