Kvaliteta usluge u ne-relacijskim distribuiranim bazama podataka ograničenih resursa

Kordaso, Ivan

Master's thesis / Diplomski rad

2016

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:677273

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-03-29**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

zir.nsk.hr

Image not found or type unknown

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

KVALITETA USLUGE U NE-RELACIJSKIM DISTRIBUIRANIM BAZAMA PODATAKA OGRANIČENIH RESURSA

Diplomski rad

Ivan Kordaso

Osijek, 2016.

Sadržaj

1. UVOD 1

1.1. Zadatak diplomskog rada 2

2. TEORIJSKA OSNOVA 3

2.1. Osnovni pojmovi 3

2.1. Ne-relacijske baze podataka 6

2.2. Klasifikacija ne-relacijskih baza podataka 9

2.2.1. Baze podataka s ključ-vrijednost modelom podataka 9

2.2.2. Baze podataka s dokument modelom podataka 10

2.2.3. Baze podataka sa stupčastim modelom podataka 11

2.2.4. Baze podataka s graf modelom podataka 12

2.3. Primjeri ne-relacijskih baza podataka 13

2.3.1. Redis 13

2.3.2. MongoDB 17

2.3.3. Hbase 19

2.3.4. Cassandra 21

2.3.5. Neo4j 22

2.4. Yahoo! Cloud Service Benchmark (YCSB) 22

2.5. ARM arhitektura 24

3. ISTRAŽIVANJE 26

3.1. Cilj i metodologija istraživanja 26

4. ANALIZA REZULTATA MJERENJA 32

4.1. Uporedba *Cubieboard2* i *HP650* rezultata mjerenja – *Redis* baza podataka 32

4.1.1. Scenarij opterećenja „A“ 33

4.1.2. Scenarij opterećenja „B“ 34

4.1.3. Scenarij opterećenja „C“ 35

4.1.4. Scenarij opterećenja „D“ 36

4.1.5. Scenarij opterećenja „F“ 37

4.1.6. Propusnost svih scenarija opterećenja 38

4.2. Usporedba rezultata mjerenja *Redis*, *Hbase* i *MongoDB* baza podataka 39
4.2.1. Scenarij opterećenja „A“ 39
4.2.2. Scenarij opterećenja „B“ 40
4.2.3. Scenarij opterećenja „C“ 41
4.2.4. Scenarij opterećenja „D“ 42
4.2.5. Scenarij opterećenja „F“ 43
4.2.6. Propusnost svih scenarija opterećenja 44
5. ZAKLJUČAK 45
LITERATURA 47
SAŽETAK 51
ABSTRACT 52
ŽIVOTOPIS 53
PRILOZI 54

1. UVOD

Velike količine podataka, velik broj korisnika i računarstvo u oblaku mijenjaju način na koji se razvijaju i stvaraju mnoge aplikacije danas. Relacijske su baze podataka mnogo godina dominirale tržištem a tijekom vremena su se pojavljivale manje ili više uspješne alternative. U posljednje vrijeme, zbog sve veće potrebe za spremanjem sve veće količine podataka o korisnicima, proizvodima, uslugama i sl., uz omogućavanje većeg broja pristupa pohranjenim podacima te uz smanjenje vremena obrade i dohvata tih podataka, moguće je pratiti razvoj baza podataka novije generacije. Zadnjih nekoliko godina primjetan je porast korištenja tzv. ne-relacijskih ili NoSQL baza podataka, koje postupno dobivaju svoj udio na tržištu.

Primarni razlozi za nastanak ovih baza su prvenstveno razlike između relacijskog modela podataka i podatkovnih struktura koje koriste moderni programski jezici te činjenica da prikupljanje i procesiranje vrlo velikih količina podataka u tvrtkama koje koriste relacijske baze podataka postaje skuplje u financijskom i vremenskom kontekstu. Razni oblici fizičkih i virtualnih računala su cijenom i performansama vrlo konkurentni, a većina relacijskih sustava za upravljanje bazama podataka, koji su trenutno prisutni na tržištu, nije dizajnirana imajući na umu skalabilnost i performanse na ovoj razini, ili pak uopće nije dizajnirana za rad u oblaku.

Zbog visokih zahtjeva za performansama i skalabilnošću, uzrokovanih velikim brojem podataka i korisnika u oblak okruženju, sve se veći broj organizacija i tvrtki odlučuje za korištenje ne-relacijskih baza podataka. Dodatno, ne-relacijske baze podataka podržavaju različite modele podataka te su stoga prikladan izbor pri bržem i jednostavnijem razvoju i održavanju aplikacija, a mnoge od njih su dizajnirane za rad u distribuiranim okruženjima.

Upravo je kvaliteta usluge u ne-relacijskim distribuiranim bazama podataka tema istraživanja koje će se provesti u svrhu ovog diplomskog rada. U radu će biti prikazana teorijska osnova o ne-relacijskim bazama podataka, ključni pojmovi važni za njihovo razumijevanje, klasifikacija ne-relacijskih baza podataka te cilj, metodologija, tijek i rezultati samog istraživanja.

1.1. Zadatak diplomskog rada

Zadatak ovog diplomskog rada bio je provesti istraživanje o kvaliteti usluge u ne-relacijskim distribuiranim bazama podataka ograničenih resursa. Kao ne-relacijske distribuirane baze podataka odabrane su baza podataka *Redis*, kao baza podataka s ključ-vrijednost modelom podataka, *Hbase* kao baza podataka sa stupčastim modelom podataka te *MongoDB* kao baza podataka s dokument modelom podataka.

Za računalne sustave ograničenih resursa korišteni su *Cubieboard2*, tzv. računalo na jednoj pločici (engl. *SBC – single board computer*) te prijenosno osobno računalo *HP650*. Svojstva koja su se mjerila kako bi se prikazala kvaliteta usluge su propusnost baza (engl. *throughput*) te vrijeme odgovara na zahtjev (engl. *request response time*). Navedena svojstva mjerila su se pomoću *YCSB* alata te proizvoljnim testiranjem opterećenja na bazi, koja je oponašala stvarni sustav kontrole pristupa.

2. TEORIJSKA OSNOVA

2.1. Osnovni pojmovi

Baza podataka je skup podataka koji su pohranjeni i organizirani tako da mogu zadovoljiti zahtjeve korisnika. To je skup međusobno povezanih podataka, pohranjenih zajedno, uz isključenje bespotrebne zalihosti (redundancije), koji mogu zadovoljiti različite primjene. Podaci su pohranjeni na način neovisan o programima koji ih koriste. Prilikom dodavanja novih podataka, mijenjanja i pretraživanja postojećih podataka primjenjuje se zajednički i kontrolirani pristup. Podaci su strukturirani tako da služe kao osnova za razvoj budućih primjena [1].

Sustav za upravljanje bazama podataka (engl. *Database Management System*) je programski sustav koji omogućuje upravljanje bazom podataka. Korisnik ili korisnički program postavlja zahtjev za obavljanjem neke operacije s podacima, a sustav za upravljanje bazama podataka ga analizira, provjerava, optimizira, transformira u niz operacija koje je potrebno obaviti na fizičkoj razini, obavlja operacije i vraća rezultat. Najvažnije zadaće sustava za upravljanje bazama podataka su zaštititi bazu podataka od neovlaštenog korištenja (engl. *data security*), spriječiti narušavanja pravila integriteta (engl. *data integrity*), osigurati obnovu podataka u slučaju uništenja (engl. *recovery*), spriječiti štetne interferencije korisnika u višekorisničkim sustavima (engl. *concurrency*), omogućiti korištenje rječnika podataka (podaci o podacima), optimizirati sve funkcije i obavljati ih efikasno koliko je to moguće [1].

Podaci u bazi su logički organizirani u skladu s nekim modelom podataka. Model podataka je skup koncepata za opis podataka na visokoj razini apstrakcije pri čemu su skriveni mnogi detalji fizičke pohrane podataka [1]. Sustavi za upravljanje bazama podataka omogućuju korisniku da pohrani podatke u skladu s modelom podataka. Model podataka je skup konceptualnih alata za opis podataka, veza među njima, semantike podataka i ograničenja konzistentnosti podataka. On pruža mogućnost opisa sheme baze podataka na fizičkoj, logičkoj i eksternoj razini. Modeli podataka su formalni sustavi koji se koriste kod modeliranja baza podataka a oni mogu biti relacijski, objektno orijentirani, objektno-relacijski, polustrukturirani, modeli za pohranu i upravljanje nizovima i serijama itd. [1]. Oni predstavljaju temelj na kojem se gradi programska potpora ili informacijski sustav i kao takvi moraju biti stabilni. Pri odabiru modela potrebno je voditi računa o njegovoj primjenjivosti u nekom području i o njegovim ograničenjima. Od 70-tih godina 20. st. pa sve do danas, relacijski je model bio dominantan s bazama podataka poput *Oraclea*, *MySQL*-a i *Microsoft SQL* servera te s drugim bazama podataka koje su pratile istu arhitekturu [2]. Relacijski model podataka zasnovan je na matematičkom pojmu relacije. U njemu

se podaci i veze medu podacima prikazuju “pravokutnim” tablicama, tzv. relacijama [3]. Iako se očekivalo da će relacijski model podataka zamijeniti neki drugi modeli podataka (npr. objektni model podataka), on još uvijek prevladava na tržištu. Da su još uvijek najkorištenije relacijske baze podataka vidljivo je na slici 2.1., na kojoj je prikazano 10 trenutno najpopularnijih baza podataka [4]. Prva tri mjesta zauzimaju relacijske baze podataka a od petnaest najkorištenijih baza, čak su devet relacijske baze podataka.

305 systems in ranking, June 2016

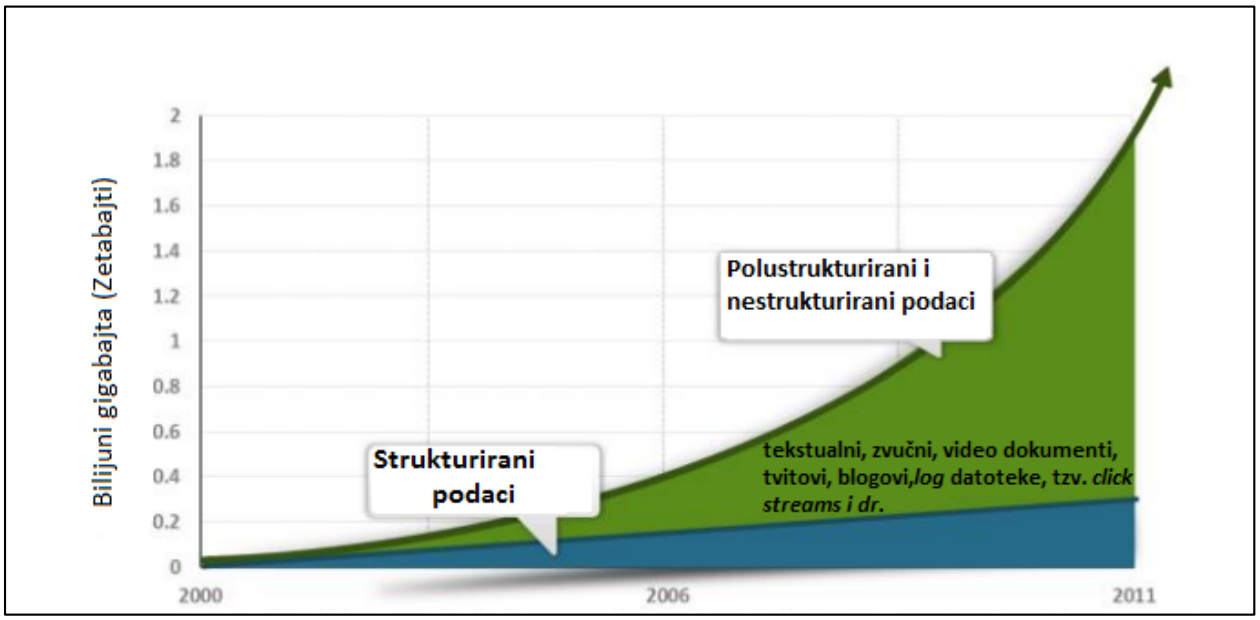
Rank Jun 2016	Rank May 2016	Rank Jun 2015	DBMS	Database Model	Score Jun 2016	Score May 2016	Score Jun 2015
1.	1.	1.	Oracle	Relational DBMS	1449.25	-12.78	-17.11
2.	2.	2.	MySQL	Relational DBMS	1370.13	-1.69	+91.78
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1165.81	+22.99	+47.76
4.	4.	↑ 5.	MongoDB	Document store	314.62	-5.60	+35.57
5.	5.	↓ 4.	PostgreSQL	Relational DBMS	306.60	-1.01	+25.70
6.	6.	6.	DB2	Relational DBMS	188.57	+2.61	-10.12
7.	7.	↑ 8.	Cassandra	Wide column store	131.12	-3.38	+22.21
8.	8.	↓ 7.	Microsoft Access	Relational DBMS	126.22	-5.35	-20.27
9.	↑ 10.	9.	SQLite	Relational DBMS	106.78	-0.48	-1.19
10.	↓ 9.	10.	Redis	Key-value store	104.49	-3.74	+9.00
11.	11.	↑ 14.	Elasticsearch	Search engine	87.41	+1.10	+17.32
12.	12.	↑ 13.	Teradata	Relational DBMS	73.84	+0.09	+0.75
13.	13.	↓ 11.	SAP Adaptive Server	Relational DBMS	71.68	+0.21	-16.49
14.	14.	↓ 12.	Solr	Search engine	64.07	-1.55	-17.18
15.	15.	15.	HBase	Wide column store	52.99	+1.15	-8.71

Sl. 2.1. Prikaz 15 najpopularnijih baza podataka u lipnju 2016. godine [5]

No, u novije vrijeme sve više tvrtki razmišlja o alternativama relacijskim bazama podataka a razlog tomu su pojava sve veće količine podataka s kojima se moraju nositi njihove aplikacije te pojava potrebe za smanjenjem troškova poslovanja. Tehnologija relacijskih baza podataka sve se teže nosi sa sve većim brojem korisnika i sve većom količinom podataka [2].

Big data je pojam koji opisuje velike količine podataka, kako strukturirane tako i nestrukturirane, koji preplavljuju poslovanje različitih organizacija na dnevnoj bazi. Njihova važnost leži u tome što se njihovom analizom može doći do uvida u informacije koje će voditi do boljih odluka te do boljih strateških i poslovnih poteza [4]. Pojmom *big data* označuju se velike količine podataka s kojima je veoma teško ili gotovo nemoguće raditi uz pomoć standardnih alata ili relacijskih baza podataka. U hrvatskome se jeziku često upotrebljava engleski naziv *big data*, a u optjecaju je i nekoliko hrvatskih prijevoda toga naziva kao što su veliki podaci, golemi podaci, velika količina podataka i sl. Prikupljanje te velike količine podataka postaje sve jednostavnije a javlja se i mogućnost pristupanja podacima preko trećih strana kao što su to npr. Facebook, Twitter

i sl. Osobni podaci korisnika, podaci ovisni o lokaciji, graf-orijentirani podaci, podaci generirani od strane korisnika, podaci generirani od strane sustava i podaci generirani u stvarnom vremenu samo su neki od primjera koji upućuju na veliku promjenu i povećanje količine podataka koji se prikupljaju[2]. Stoga i ne čudi što su programeri prepoznali rastuću vrijednost utjecaja ovih podataka i odlučili ih iskoristiti kako bi poboljšali postojeće aplikacije ali i kako bi razvili nove. Primjena podataka kontinuirano mijenja prirodu „života na internetu“ što uključuje *online* komunikaciju, *online* kupovinu, *online* oglašavanje, hobije, te upravljanje odnosima. Aplikacije koje ne zadovoljavaju zahtjeve trenutnih velikih tržišnih trendova brzo zaostaju za onima koje ih zadovoljavaju.



Slika 2.2. Rast strukturiranih te polustrukturiranih i nestrukturiranih podataka od 2000. do 2011. godine [2]

Osim količine podataka koja ubrzano raste, mijenja se i priroda podataka. Prikupljaju se različiti tipovi podataka koji zahtijevaju fleksibilnu bazu podataka u koju će se na jednostavan način inkorporirati bilo koji tip podatka. Stoga, baza podataka mora imati sposobnost učinkovite pohrane i vrlo brzog pristupa novim vrstama podataka kao što su polustrukturirani i nestrukturirani podatci. Na slici 2.2. vidljiv je porast polustrukturiranih i nestrukturiranih podataka od 2000. godine pa nadalje.

NoSQL baze podataka zadovoljavaju rastuće trendove za pohranu, obradu i pronalaženje podataka pružajući fleksibilan model podataka koji mapira zahtjev organizacije i pojednostavljuje komunikaciju između aplikacije i baze podataka, a rezultat toga je manja količina pisanja koda, rjeđe ispravljanje pogrešaka a i lakše održavanje aplikacije.

Važnost koncepta *Big data* ne leži u količini podataka već u njegovoj namjeni. Prikupljene je podatke moguće analizirati kako bi se došlo do informacija koje će omogućiti uštedu

troškova, uštedu vremena, razvoj novih proizvoda i donošenje pametnih odluka [2]. Kada se velike količine podataka kombiniraju sa snažnom analitikom moguće je utvrditi uzroke kvarova, problema i nedostataka u gotovo realnom vremenu, odraditi ponovno izračunavanje rizika u svega nekoliko minuta, otkriti prijevare prije nego na ikakav način utječu na tvrtku i dr.

Relacijske baze podataka imaju vrlo slabe mogućnosti usvajanja novih vrsta podataka zbog toga što ovise o statičkim shemama i zbog strogog pristupa prema modeliranju podataka. Google, Amazon, Facebook i Linkedln su među prvim organizacijama i tvrtkama koje su otkrile niz ograničenja koje nosi tehnologija relacijskih baza podataka u radu s velikim brojem podataka i velikim brojem korisnika. Zbog njihove inicijative za potražnjom alternativnih rješenja, nekoliko se razvojnih tvrtki počelo baviti s razvojem baza podataka s drugačijim modelom upravljanja podacima. Kao rezultat, stvorene su ne-relacijske baze podataka [2]. Danas se ne-relacijske baze podataka veoma brzo razvijaju i uvode u poslovanje brojnih internetskih tvrtki i drugih poduzeća. Na slici 2.1. vidljivo je da su jedine baze koje konkuriraju relacijskim bazama podataka na tržištu zapravo ne-relacijske baze podataka i većina njih (*MongoDB*, *Cassandra*, *Redis*) bilježi porast popularnosti od lipnja 2015. godine do danas, što ide u korist tezi da se ne-relacijske baze podataka sve više koriste [5].

Sve se više ne-relacijske baze podataka smatraju boljim izborom u usporedbi s relacijskim bazama te sve više organizacija i tvrtki smatra da se zahtjevi za performansama i skalabilnosti zbog velikog broja podataka i korisnika u oblak okruženju mogu uspješno ispuniti korištenjem ne-relacijskih baza podataka. One podržavaju različite modele podataka te su stoga prikladan izbor pri bržem i jednostavnijem razvoju i održavanju aplikacija, a mnoge od njih su dizajnirane za rad u distribuiranim okruženjima (računalni klaster ili računalni oblak) [6].

2.1. Ne-relacijske baze podataka

Izraz NoSQL prvi je put iskoristio, 1998. godine, Carlo Strozzi za njegovu još uvijek relacijsku bazu podataka koja je ime dobila zbog činjenice da ne koristi SQL jezik, nego koristi *shell skripte* (tekstualne datoteke s naredbama koje omogućuju različite operacije u UNIX sustavima) i standardne UNIX *pipelines* (mehanizam u UNIX sustavima pomoću kojeg je moguće preusmjeriti izlaz iz jednog programa na ulaz u drugi program) za postavljanje upita nad podacima. Izraz NoSQL se u današnjem značenju počeo upotrebljavati tek 2009. godine a nastao je kao jedan od produkata konferencije koja se održavala u San Franciscu, kada ga je Eric Evans, zaposlenik Rackspacea, iskoristio kao termin za trenutni porast korištenja ne-relacijskih baza podataka [7].

Razvojni timovi imaju utjecaj na odabir tehnologija koje moraju zadovoljavati neke određene potrebe. Jedan od razloga prelaska na ne-relacijske baze podataka je porast broja istovremenih pristupa globalnih korisnika aplikacijama putem interneta i mobilnih uređaja. Također, jedan je od razloga i velika količina podataka koji se skupljaju i obrađuju te je postalo nužno skupljati različite vrste strukturiranih i nestrukturiranih podataka a njihova je upotreba postala sastavni dio koji obogaćuje aplikacije. Uz to, u današnje vrijeme s pojavom tehnologija na oblaku, aplikacije koriste troslojnu internet arhitekturu koja je javna ili privatnu tehnologiju na oblaku koja podržava velik broj korisnika i veliku količinu informacija [2].

Ne-relacijske baze podataka su baze podataka za koje većinom vrijedi da nisu relacijske, distribuirane su, rade na principu otvorenog koda (engl. *open source*) i horizontalno su skalabilne [8]. Da je baza podataka ne-relacijska znači da više nije bazirana na relacijskom modelu baze podataka, odnosno podaci u njoj se više ne organiziraju u dvodimenzionalne tablice koje se nazivaju relacijama, stoga je princip djelovanja ne-relacijskih baza podataka, u usporedbi s relacijskim bazama, potpuno drugačiji [8]. Ne-relacijske baze podataka su distribuirane što znači da se podaci pohranjuju i da se njima upravlja različitim uređajima, stoga se ti podaci mogu replicirati. Distribuirana baza podataka (DBP) je skup logički povezanih baza podataka razmještenih u različitim čvorovima računalne mreže (LAN, MAN, WAN). Distribuirani sustav za upravljanje bazama podataka (DSUBP) je programski sustav koji upravlja distribuiranom bazom podataka na takav način da je distribuiranost sustava transparentna prema korisnicima. Distribuirani sustav za upravljanje bazama podataka obuhvaća određen broj lokalnih sustava za upravljanje bazama podataka. Svaki lokalni sustav za upravljanje bazama podataka predstavlja jedan čvor (engl. *site, node*) distribuiranog sustava. Postoji dvosmjerna veza između svaka dva čvora, tj. svaki čvor može direktno ili indirektno komunicirati sa svakim čvorom. Čvorovi distribuiranog sustava za upravljanje bazama podataka ne dijele zajedničke fizičke komponente (disk, memorija, procesor). Čvorovi su sposobni obavljati transakcije koje zahtijevaju isključivo lokalni pristup podacima (lokalne transakcije), ali i transakcije koje zahtijevaju pristup podacima iz različitih čvorova (globalne transakcije). Čvorovi posjeduju određeni stupanj lokalne autonomije. Baza podataka je distribuirana ako podržava barem jednu globalnu aplikaciju [1].

Važan dio postupka oblikovanja distribuirane baze podataka je određivanje načina distribucije podataka. Podaci se smještaju u čvorove u kojima se najčešće koriste i tako se minimalizira mrežni promet. Proces oblikovanja distribucije odvija se kroz sheme fragmentacije i alokacije. Shema fragmentacije je podjela baze podataka u disjunktni skup fragmenata koji obuhvaćaju sve podatke u bazi podataka uz zadovoljenje pravila da se baza podataka može rekonstruirati iz tih fragmenata bez gubitka informacija. Relacije mogu biti razdijeljene u

fragmente horizontalno ili vertikalno ili horizontalno i vertikalno. Shemom alokacije opisuje se koji je fragment pridružen kojem čvoru distribuiranog sustava. Alokacija je zapravo stupanj replikacije fragmenta odnosno broj čvorova u kojima je fragment alociran. S obzirom na stupanj replikacije fragmenta moguće je razlikovati particionirane ili nereplicirane baze podataka u kojima je svaki od fragmenata alociran u točno jednom čvoru, tj. stupanj replikacije svakog fragmenta jednak je broju jedan. Također, postoje i potpuno replicirane baze podataka u kojima je svaki od fragmenata alociran u svim čvorovima. Svaki čvor sadrži repliku cijele baze podataka, tj. stupanj replikacije svakog fragmenta jednak je *n*, odnosno broju čvorova u distribuiranim sustavima za upravljanje bazama podataka. Postoje još i parcijalno replicirane baze podataka. Parcijalno replicirane baze podataka nisu niti particionirane niti potpuno replicirane tj. svaki od fragmenata može biti alociran u jednom, više ili u svim čvorovima [1]. Prednosti repliciranih baza podataka su povećanje dostupnosti, tj. ako je čvor u kojem je pohranjena kopija fragmenta nedostupan, sustav može pristupiti kopiji fragmenta u nekom drugom čvoru. Također, smanjuje se volumen prijenosa podataka odnosno aplikacije mogu pristupati lokalno podacima koji se često koriste u više čvorova. Još jedna prednost repliciranih baza očituje se u paralelnom obavljanju dijelova istog upita odnosno upit koji obrađuje fragment može se dekomponirati, te se svaki dio upita može obavljati nad jednom od kopija fragmenta. Nedostatak repliciranih baza podataka je problem konzistentnosti kopija istog elementa jer sustav mora osigurati konzistentnost svih kopija. Operacija pisanja (unos, brisanje, izmjena) jedne kopije fragmenta mora se propagirati prema svim čvorovima u kojima je taj fragment alociran. Također, veći broj operacija koje treba obaviti u većem broju čvorova može uzrokovati smanjenje dostupnosti i povećanje broja potpunih zastoja pri sinkronoj replikaciji ili narušavanje konzistentnosti pri asinkronoj replikaciji [1].

Da ne-relacijske baze podataka rade na principu otvorenog koda znači da bilo tko može imati uvid i besplatan pristup kodu, može ga mijenjati i na temelju njega samostalno sastavljati novi kod. Horizontalna skalabilnost znači da se performanse sustava povećavaju dodavanjem novih poslužitelja [8]. Za razliku od ne-relacijskih baza, relacijske baze podataka su vertikalno skalabilne, odnosno brzina same baze podataka može se povećati npr. ugradnjom boljeg procesora u poslužitelj, ugradnjom više memorije i sl.

Definicija ne-relacijskih baza podataka može se proširiti s još nekim svojstvima. Kod ne-relacijskih baza podataka shema podataka ne postoji ili je fleksibilna sa slabim ograničenjima. Također, čvorovi su neovisni jedan od drugoga, podaci se lako repliciraju a sustav pruža jednostavno API sučelje (sučelje za programiranje aplikacija) [8]. Jedna od razlika između relacijskih i ne-relacijskih baza podataka je ta što se relacijske baze podataka temelje na ACID svojstvima dok mnogi sustavi ne-relacijskih baza podataka ne podržavaju sva ACID svojstva već

samo neka od njih. Naime akronim ACID predstavlja skup poželjnih svojstava za transakcije u bazi podataka koji uključuje atomarnost (engl. *atomicity*), konzistentnost (engl. *consistency*), izolaciju (engl. *isolation*) i trajnost (engl. *durability*). Atomarnost je svojstvo koje nalaže da transakcije moraju slijediti pravilo „sve ili ništa“ u značenju da će se ili sve promjene do kojih je došlo transakcijom pohraniti ili se neće pohraniti nijedna [9]. Svojstvo konzistentnosti znači da se transakcije uvijek odvijaju konzistentno i da uvijek moraju dovesti bazu podataka u konzistentno stanje [9]. Ako se promjena dogodila na jednoj bazi u određenom čvoru, doći će do sinkronizacije podataka i u ostalim čvorovima. Svojstvo izolacije stvara iluziju da se svaka transakcija provodi sama. Ovo svojstvo osigurava da transakcije koje se izvršavaju u isto vrijeme nemaju utjecaja jedna na drugu [9]. Svojstvo trajnosti nalaže da jednom kad se transakcija uspješno završila, promjene do kojih je došlo nad podacima moraju ostati čak i u slučaju naknadnih kvarova [9]. Jedan od osnovnih razloga za brzinu ne-relacijskih baza podataka i za prednost koju imaju s aspekta performansi sustava je upravo taj što ih većina ne podržava sva ACID svojstva jer iako korisna, ACID svojstva donose dodatno opterećenje na sustav za upravljanje bazom podataka.

2.2. Klasifikacija ne-relacijskih baza podataka

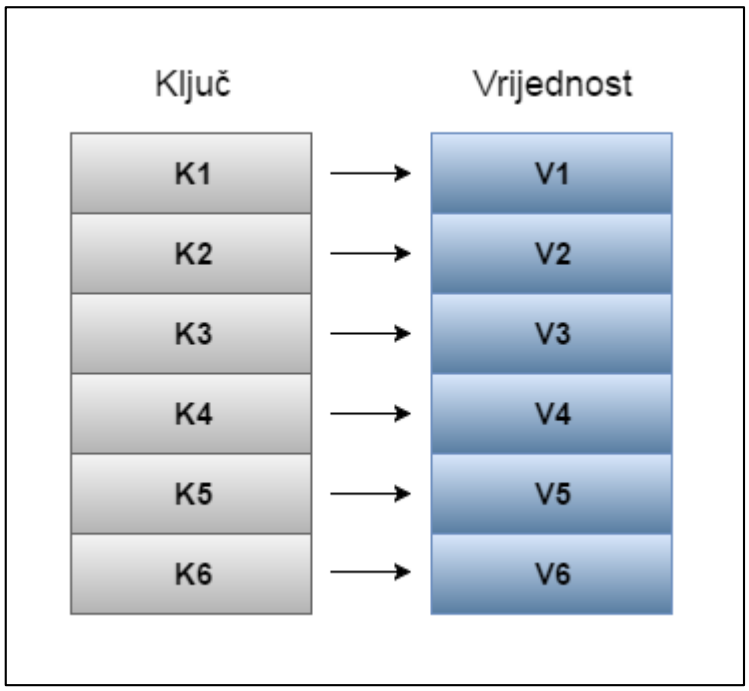
Ne-relacijske baze podataka mogu se klasificirati prema kriteriju modela podataka te je tada moguće razlikovati baze podataka s [2, 6, 10]:

- ključ-vrijednost modelom podataka (engl. *key-value model*),
- dokument modelom podataka (engl. *document model*),
- stupčastim modelom podataka (engl. *column-family model*, *wide-column model*) i
- graf modelom podataka (engl. *graph model*).

2.2.1. Baze podataka s ključ-vrijednost modelom podataka

Baze podataka s ključ-vrijednost modelom podataka su najjednostavnije ne-relacijske baze podataka [2]. Pojednostavljen dijagram ključ-vrijednost modela podataka prikazan je na slici 2.3. Baze podataka s ključ-vrijednost modelom podataka su baze u kojima su podaci spremljeni u parovima ključ-vrijednost. Svi podaci unutar baze podataka su spremljeni na ovaj način pri čemu vrijednost koja se pohranjuje uz pojedini ključ može biti bilo koji tip podatka [6]. Ovaj model je izuzetno koristan za prikazivanje ne strukturiranih polimorfnih podataka, jer baza podataka ne zahtijeva specifični model, odnosno shemu podataka između parova ključ-vrijednost [10]. Ovakvi sustavi imaju iznimno visoke performanse pri čitanju podataka s obzirom na to da ovakve baze

podataka koriste jedinstveni identifikator za dohvaćanje vrijednosti. Baze podataka ove vrste su *Redis, Riak, Berkeley DB, Memcached HamsterDB, Amazon DynamoDB* i *Project Voldemort i* dr. Baze podataka s ključ-vrijednost modelom podataka korisne su za uži skup aplikacija koje vrše upit nad podacima uz pomoć samo jedne vrijednosti ključa. Prednosti ovih sustava su njihova skalabilnost i performanse, koje mogu biti visoko optimizirane zbog jednostavnosti pristupa podacima [10].

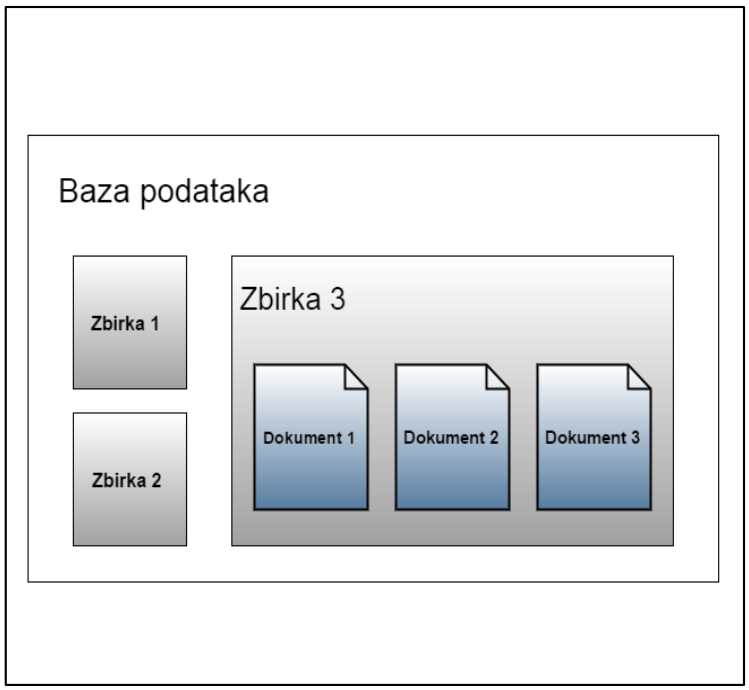


Sl. 2.3. Dijagram ključ-vrijednost modela podataka

2.2.2. Baze podataka s dokument modelom podataka

Baze podataka s dokument modelom podataka temelje se na radu s različitim vrstama dokumenata (XML, JSON, BSON i dr.), uključujući i binarne dokumente. One su nalik bazama podataka s ključ-vrijednost modelom podataka pri čemu je vrijednost dokument. U ove se baze podaci pohranjuju u formatu ključ-dokument, pri čemu je ključ identifikator podatka [6]. Dokumenti sadrže jedno ili više polja, gdje svako polje sadrži upisanu vrijednost (znakovni niz, datum, binarni broj i sl.). Svaki zapis i njemu pridruženi podaci obično su pohranjeni zajedno u jednom dokumentu. To pojednostavljuje pristup podacima i smanjuje ili čak eliminira potrebu za pridruživanjem i složenim transakcijama. U bazi podataka s dokument modelom podataka, model podataka je dinamičan, tj. svaki dokument može sadržavati različita polja. Ova fleksibilnost može biti osobito korisna pri modeliranju nestrukturiranih i polimorfnih podataka. Također, navedena fleksibilnost olakšava rad na aplikaciji tijekom njena razvoja, kao npr. pri postupku dodavanja

novih polja. U ovim bazama dohvat podataka odvija se na temelju bilo kojeg polja u dokumentu [10]. Dokumenti se mogu dodatno indeksirati ovisno o svojoj strukturi, ali nije nužno da imaju istu strukturu ili da su dokumenti unutar jedne baze podataka istoga formata. Također, ove baze podataka podržavaju dinamičke podatke koji se mogu mijenjati u bilo kojem vremenu. One podržavaju složene strukture podataka i omogućavaju lakše ispravljanje pogrešaka [2]. Dijagram dokument modela podataka prikazan je na slici 2.4. Predstavnici ove skupine baza podataka su *MongoDB, CouchDB, SimpleDB* i dr. Za baze podataka s dokument modelom podataka može se reći da su baze podataka opće namjene jer su korisne za različite aplikacije zbog fleksibilnosti modela podataka, sposobnosti za provođenje upita na bilo kojem polju i prirodnog mapiranja odnosno pretvaranja dokumenata kao modela podatka u objekte u suvremenim programskim jezicima [9].

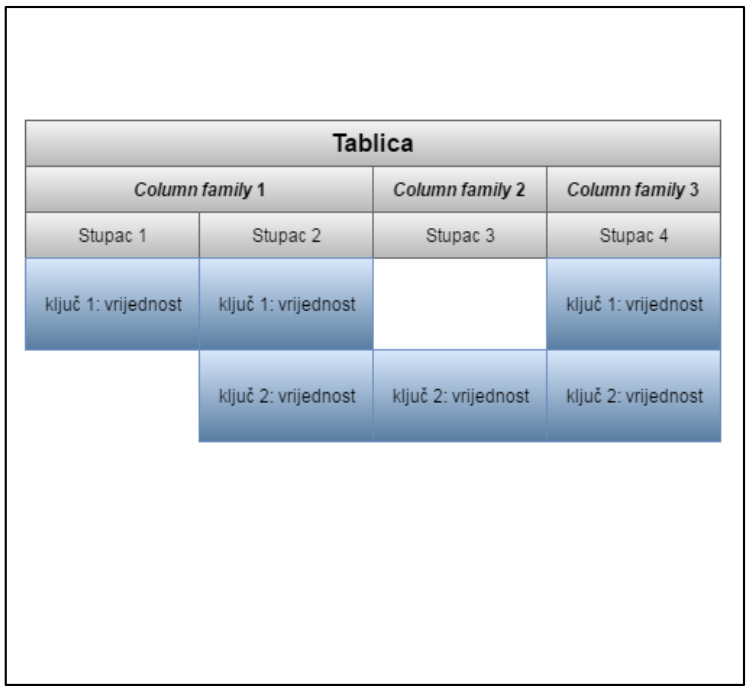


Sl. 2.4. Dijagram dokument modela podataka

2.2.3. Baze podataka sa stupčastim modelom podataka

Baze podataka sa stupčastim modelom podataka ili baze podataka sa skupinama stupaca, temelje se na *BigTable* modelu podataka predloženom od Googlea a nastale su kao prošireni model baza podataka s ključ-vrijednost modelom podataka. Ključ je vrsta identifikatora, a vrijednosti su podatkovne mape, koje sadržavaju skupine (stupce) vezanih podataka [6]. Ove baze podataka pohranjuju podatke u obliku stupova, što omogućuje brže učitavanje određenih stupaca u memoriji i omogućuje izradu proračuna svih vrijednosti u stupcu. Također, ove su baze optimizirane za upite

u velikim skupovima podataka [2]. Svaki zapis može se razlikovati u broju stupaca koji se pohranjuju a stupci mogu biti ugniježđeni unutar drugih stupaca što se naziva „super stupcima“ (engl. *super columns*) [10]. „Super stupci“ mogu sadržavati druge stupce, ali ne i druge „super stupce“. Kod baza podataka sa stupčastim modelom podataka pojavljuje se i pojam „obitelji“ stupaca (engl. *column family*). To je struktura koja može sadržavati neograničen broj redova a sastoji se od strukture mape s ključem i vrijednosti. Na slici 2.5. prikazan je dijagram stupčastog modela podataka. Najpoznatiji predstavnici ove grupe baza podataka su *Cassandra*, *Hbase*, *Hypertable*, *Amazon SimpleDB* i dr. Baze podataka sa stupčastim modelom podataka, kao i baze podataka s ključ-vrijednost modelom podataka, korisne su za uži skup aplikacija koje vrše upit nad podacima uz pomoć samo jedne vrijednosti ključa. Također, njihove glavne prednosti su visoka skalabilnost i performanse [10].

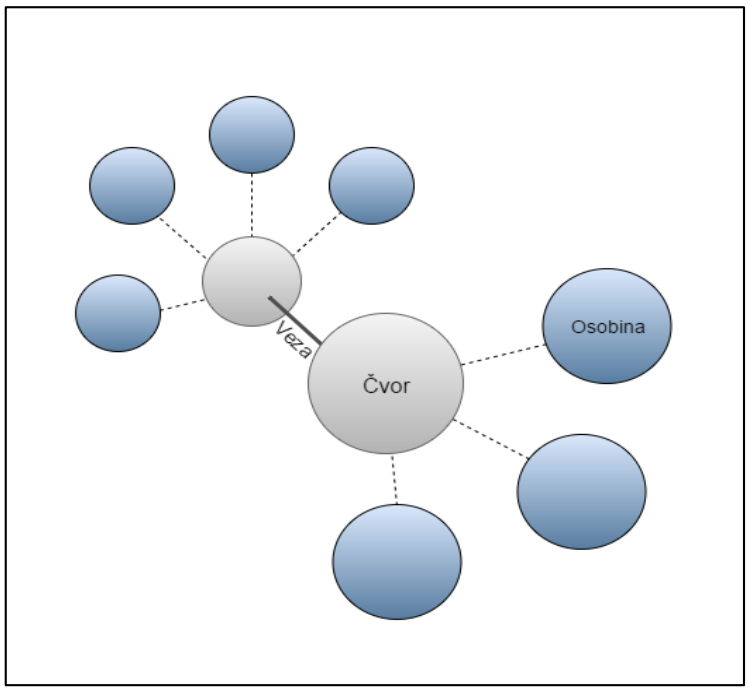


Sl. 2.5. Dijagram stupčastog modela podataka

2.2.4. Baze podataka s graf modelom podataka

Baze podataka s graf modelom podataka služe za pohranu podataka u formi usmjerenog grafa. Entiteti i relacije među njima se također nazivaju čvorovi i bridovi. Ovaj model podataka omogućuje postavljanje upita koji se temelje na vezama između čvorova [6]. Na slici 2.6. prikazan je dijagram graf modela podataka. Baza podataka s graf modelom podataka je baza čije su sheme i/ili instance modelirane kao (usmjereni) graf, ili generalizacija graf podatkovne strukture, pri čemu je manipulacija podacima izražena graf-orijentiranim operacijama i tipovima i ima

integritetska ograničenja prikladna strukturi grafa [6]. Iako su poprilično kompleksne, baze podataka s graf modelom podataka imaju široku primjenu za različite aplikacije. Njihova glavna prednost je što olakšavaju modeliranje odnosa između entiteta u aplikaciji [10]. Najznačajniji predstavnici baza podataka ove skupine su *Neo4j, AllegroGraph, OrientDB, InfiniteGraph* i dr. Baze podataka s graf modelom podataka su korisne u slučajevima kada su odnosi između entiteta temelji aplikacije, kao što je to slučaj kod društvenih mreža [10].



Sl. 2.6. Dijagram graf modela podataka

2.3. Primjeri ne-relacijskih baza podataka

U narednim potpoglavljima bit će predstavljene i opisane najpoznatije ne-relacijske baze podataka i to: *Redis* kao najpoznatiji predstavnik baza podataka s ključ-vrijednost modelom podataka, *MongoDB* kao predstavnik baza podataka s dokument modelom podataka, *Cassandra* i *Hbase* kao predstavnici baza podataka sa stupčastim modelom podataka te *Neo4J* kao predstavnik baza podataka s graf modelom podataka.

2.3.1. Redis

Redis (*REmote DIctionary Server*) je jednostavna i izrazito brza baza podataka s ključ-vrijednost modelom podataka otvorenog koda. *Redis* je tzv. *in-memory* skladište za pohranu podataka što znači da se prvenstveno oslanja na pohranu podataka u radnoj memoriji [11]. Za *Redis* se općenito može reći da je napredna baza podataka s ključ-vrijednost modelom podataka

koja se u posljednje vrijeme sve više koristi. Koristi se u aplikacijama kao što su Twitter, GitHub, Weibo, Pinterest, Snapchat, Craiglist, Digg, StackOverflow, Flickr i u mnogim drugima.

Redis je do 2013. godine sponzorirao VMware, od 2013. godine Pivotal, a od 2015. godine pa nadalje sponzorira ju Redis Labs [12]. Napisao ju je talijan Salvatore Sanfilippo. Napisana je u jeziku ANSI C a podržava master-slave replikaciju. Njezine glavne prednosti su brzina izvršenja, jednostavna konfiguracija i pokretanje, svakodnevni razvoj i nadogradnja i atomske operacije. Licencirana je od strane BSD-a (*Berkeley Software Distribution*).

Redis nije obično ključ-vrijednost „skladište" podataka već je zapravo poslužitelj- struktura podataka koja podržava različite vrste vrijednosti. Dok se u tradicionalnim ključ-vrijednost spremištima podataka ključevi znakovnog niza (engl. *string keys*) vežu za vrijednosti znakovnog niza (engl. *string values*), u *Redisu* vrijednost nije ograničena na jedan jedini znakovni niz već se može nositi s mnogo kompleksnijim strukturama podataka. *Redis* je poslužitelj sljedećih struktura podataka: znakovni nizovi (engl. *strings*), skupovi (engl. *sets*), sortirani skupovi (engl. *sorted sets*), *hashevi* i liste (engl. *lists*) [13].

Znakovni nizovi predstavljaju najosnovniju vrstu vrijednosti u *Redisu*. Oni su binarno sigurni što znači da mogu sadržavati bilo koju vrstu podataka (veličine do 512 MB) [14]. Znakovni nizovi se mogu koristiti kao brojači pomoću naredbi: INCR, DECR, INCRBY. Budući da *Redis* podržava atomske operacije, ukoliko dva klijenta konkurentno pristupaju istom ključu *Redis* će to znati riješiti te će uspješno odgovoriti na zahtjev i jednom i drugom klijentu. Također, znakovne nizove moguće je koristiti kao vektore slučajnog pristupa pomoću naredbi GESTRANGE i SETRAGNE. Uz to, pomoću naredbi GETBIT i SETBIT, moguće je šifrirati mnogo podataka u malom prostoru ili stvoriti tzv. *Bloom Filters*, prostorno-efikasne strukture podataka koje se temelje na vjerojatnosti [14].

Skupovi u *Redisu* su neuređene zbirke jedinstvenih elemenata. *Redis* ne dopušta ponavljanje elemenata u skupovima što znači da nije potrebna provjera postoji li neki element već u skupu ili ne jer *Redis* to samostalno odrađuje. Maksimalan broj elemenata je 2^{32}-1, tj. ukupno 4294967295 elemenata po skupu. Skupovi su korisni za operacije nad podacima kao što su unija, naredbom SUNION, ili presjek, naredbom SINTER. Pomoću skupova mogu se pratiti jedinstvene informacije, pa se tako primjerice, pomoću naredbe SADD, mogu pratiti jedinstvene IP adrese koje posjećuju određeni blog i sl. Skupovi su korisni za predstavljanje odnosa. Pomoću *Redisa* može se stvoriti sustav označavanja tako da se koriste skupovi za svaku oznaku. Zatim se ID oznake svih objekata, koji imaju određenu oznaku, mogu dodati u skup koji predstavlja tu određenu oznaku, pomoću naredbe SADD. Skupovi se koriste i kako bi se nasumce izdvojili

elementi pomoću naredbi SPOP i SRANDMEMBER. Oni predstavljaju idealnu strukturu za društvene mreže [14].

Sortirani skupovi slično kao i skupovi, predstavljaju uređenu zbirku jedinstvenih elemenata. Razlika između skupova i sortiranih skupova leži u tome što se svakom elementu u sortiranom skupu priključuje rezultat na osnovu kojeg se rangiraju elementi od najmanjeg prema najvećem rezultatu. Iako su elementi jedinstveni, rezultati se mogu ponavljati. Svakom elementu se pridružuje samo jedan rezultat. Maksimalan broj elemenata po skupu je 2^{32}-1 tj. ukupno 4294967295 elemenata. Sa sortiranim skupovima moguće je dodati, ukloniti, ili ažurirati elemente na veoma brz način (u vremenu proporcionalnom logaritmu broja elemenata). Uz naredbe koje su podržane i kod običnih skupova omogućene su i naredbe pomoću kojih se lako može doći do podataka kao što su npr. prvih deset elemenata prema rezultatu, rangiranje elemenata prema rezultatu i sl. Sortirani se skupovi ponajviše koriste kada je u pitanju nekakvo bodovanje ili rangiranje jer je u veoma kratkom vremenskom roku moguće dobiti podatke o položaju korisnika na osnovu bodova (naredba ZRANGE) ili je primjerice moguće veoma brzo ažurirati rezultate (naredba ZADD). Također, često se koriste i za indeksiranje podataka koji su pohranjeni u bazi. To su najnapredniji tipovi podataka u *Redisu* [14].

Hashevi su veze između polja znakova i vrijednosti znakova što ih čini savršenim tipom podataka za predstavljanje objekata, kao što je npr. korisnik s poljima kao što su ime, prezime, dob itd. Oni upravljaju s nizovima znakova tj. postoji više parova ključ-vrijednost u okviru jednog *hasha*. *Hash* se koristi kada nam je potrebna struktura slična tablicama u bazama podataka. *Hash* s određenim brojem polja se pohranjuje na način da zauzima jako malo prostora pa se tako u relativno malom prostoru mogu pohraniti milijuni objekata. Iako se *hashevi* ponajviše koriste kako bi predstavljali objekte, pomoću njih je moguće pohraniti mnogo elemenata, pa se stoga koriste i za neke druge zadatke. Svaki *hash* također može pohraniti do 2^{32}– 1 parova u polju vrijednosti. Prednost *hasheva* svakako je ušteda memorije. Važno je napomenuti da se manji *hashevi*, odnosno *hashevi* sa samo nekoliko elemenata s manjim vrijednostima, šifriraju na poseban način, odnosno na način koji doprinosi uštedi memorije [14].

Liste su zapravo liste nizova znakova u kojima su elementi sortirani prema redoslijedu dodavanja u listu. Moguće je i dodavati elemente na listu na način da se novi elementi dodaju na početak (na glavu) liste s naredbom LPUSH ili na kraj (rep) liste s naredbom RPUSH. Maksimalan broj elemenata je 2^{32}-1 tj. ukupno 4294967295 elemenata po listi. Elementi se ubacuju u listu i brišu s liste u jednakom vremenu tj. isto je vremena potrebno kada se ubacuje element u listu od 10 i u listu od 10 milijuna elemenata. Pristupanje elementima s početka (glave) i s kraja (repa) liste veoma se brzo odvija dok je pristupanje elementima iz sredine liste sporije. Liste imaju

višenamjensku primjenu pa se tako koriste za kreiranje različitih lista čekanja, za kreiranje vremenskih linija na društvenim mrežama gdje se spomenutom naredbom LPUSH mogu unijeti novi događaji, za kreiranje lista koje nikada ne prelaze određeni broj elemenata već samo pamte najnovije *n* elemente i sl. [14].

Redis također podržava i Bitmape i HyperLogLogove a to su zapravo vrste podataka koje se baziraju na nizovima podataka ali imaju vlastitu semantiku [14].

Za svaku od navedenih struktura podataka postoje posebne naredbe. Pored standardnih naredbi *Redis* radi i s transakcijama. Naredbe MULTI, EXEC, DISCARD i WATCH su osnove transakcija u *Redisu* [15]. One omogućuju izvedbu skupine naredbi u samo jednom koraku i jamče dvije važne stvari. Prvo, sve naredbe u transakciji su serijalizirane i izvode se sekvencijalno. Nemoguće je da zahtjev nekog drugog klijenta poremeti izvršenje izvedbe transakcije. To jamči da će naredbe biti izvršene kao izolirane operacije. Drugo, ili se sve naredbe provode ili se nijedna naredba ne provodi, što čini *Redisove* transakcije atomskim. Transakcije u *Redisu* razlikuju se od transakcija u sustavima za upravljanje relacijskim bazama podataka jer ne postoji tzv. *rollback* (metoda za poništenje transakcije do njihova početka ili do prethodno definirane točke pohrane, engl. *savepoint*). Razlog tome je što do greške u *Redis* naredbama može doći ili kada se koristi pogrešna sintaksa ili ako se pogrešno pozove određena naredba nad pogrešnim tipom podataka. Očekuje se da bi se takve greške trebale uočiti u razvojnoj fazi a ne u fazi produkcije. Drugi razlog je to što je *Redis* pojednostavljen i brz baš zato što ne omogućuje *rollback* [15].

Što se tiče postojanosti podataka, *Redis* podržava dva načina pohrane podataka i to RDB (binarno) i AOF (*Append Only File*) [16]. Oba imaju svoje prednosti i nedostatke a o odabiru se odlučuje zavisno od konkretne upotrebe. U RDB-u podaci se čuvaju u memoriji te se u određenim vremenskim intervalima spremaju na disk. U slučaju bilo kakvih kvarova može doći do gubitka podataka. U AOF-u je moguće odabrati kako će se podaci pohranjivati na disk (npr. svake sekunde, poslije svakog upita, nikad). AOF čuva i sve naredbe pokrenute nad poslužiteljem te podatke, ključeve i *logove* zajedno u jednom dokumentu što omogućuje repliciranje i stvaranje novih poslužitelja samo iz tog jednog dokumenta.

Za komunikaciju s *Redis* poslužiteljem, klijenti koriste RESP protokol, odnosno *Redis Serialization Protocol*. Iako je dizajniran specifično za *Redis* može se koristiti i za druge klijent-poslužitelj softverske projekte. Prednosti RESP-a su jednostavna implementacija, brzo raščlanjivanje i činjenica da je razumljiv ljudima. RESP ima mogućnost serijalizacije različitih tipova podataka kao što su cijeli brojevi, znakovni nizovi i redci (engl. *arrays*). Zahtjevi se šalju od klijenta na *Redisov* poslužitelj kao redovi znakova koji predstavljaju argumente naredbe koja se treba izvršiti. *Redis* odgovara s tipom podataka koji je specifičan za zadanu naredbu. RESP je

binarno siguran i ne zahtijeva obradu rasutih podataka koji se prenose iz jednog procesa u drugi zato što se koristi unaprijed zadana duljina za prenošenje tih podataka [17].

Cijena *Redisove* brzine je nedostatak povezanosti između podataka odnosno referencijalnog integriteta, stoga je potrebna veoma dobra organizacija podataka na razini same aplikacije.

2.3.2. MongoDB

Najjednostavnije rečeno, *MongoDB* je najpoznatija ne-relacijska baza podataka s dokument modelom podataka otvorenog koda. Napisana je u programskom jeziku C++ a licencirana je od strane APGL-a (*Affero General Public License*).

Postoji nekoliko koncepata koji se trebaju razumjeti kako bi se razumjelo funkcioniranje *MongoDB* baze podataka. Prvo, unutar *MongoDB*-a ne mora postojati nijedna a može postojati i više baza podataka od kojih svaka predstavlja „spremište“, na visokoj razini, za sve ostalo. Baza podataka ne mora imati nijednu ili može imati više zbirki, koje su slične zbirkama u relacijskim bazama podataka. Zbirke se ne moraju sastojati od nijednog ili se mogu sastojati od više dokumenata koji se mogu promatrati kao redci u relacijskim bazama podataka. Dokument se može sastojati od nijednog ili više polja koja se mogu promatrati kao stupci u relacijskim bazama podataka. Zbirke se mogu indeksirati a indeksi u *MongoDB* bazi podataka funkcioniraju kao i indeksi u relacijskim bazama podataka. I posljednje, postavljanje upita u bazi *MongoDB* odvija se pomoću pokazivača (engl. *cursor*), a njega je moguće definirati kao pokazivač na skup rezultata kojima se može upravljati (npr. brojanje, preskakanje) prije konkretnog dohvaćanja podataka [18].

Iako su spomenuti koncepti poprilično slični konceptima u relacijskim bazama podataka oni ipak nisu identični. Osnova razlika proizlazi iz činjenice da relacijske baze podataka definiraju stupce na razini tablice dok dokumenti baze podataka definiraju polja na razini dokumenta. Dakle, svaki dokument unutar zbirke može posjedovati svoj jedinstveni skup polja. Kao rezultat toga, zbirke su pojednostavljene u usporedbi s tablicama dok dokumenti imaju mnogo više informacija nego redci [18].

Jedna od prednosti *MongoDB* baze podataka je fleksibilnost odnosno činjenica da ne primjenjuje fiksnu shemu što ju čini mnogo fleksibilnijom od tradicionalnih tablica u relacijskim bazama podataka. U nekim se slučajevima, kao prednost dinamičke sheme može vidjeti nedostatak postavki vezanih uz objektno-orijentirano programiranje i nedostatak potrebe za istim. *MongoDB* ne pruža opciju mapiranja po svojstvu i tipu podatka.

MongoDB je veoma koristan kada su u pitanju zapisi. On nudi mogućnost da se naredba pošalje i vrati odmah, bez čekanja da se zapis potvrdi, što čini zapis veoma brzim. Također, moguća je i kontrola nad zapisima na način da se poštuje trajnost podataka. *MongoDB* se u novijim verzijama pozabavio s problemom koji se tiče trajnosti podataka. U staroj inačici *MongoDB* nije podržavao svojstvo trajnosti odnosno, kada bi došlo do rušenja servera, došlo bi i do gubitka oštećenih podataka. *MongoDB* je po tom pitanju dorađen uz pomoć opcije dokumentiranja (engl. *journaling*) koja dopušta brzi oporavak poslužitelja u slučaju rušenja ili naglog nestanka struje [18]. Još jedna značajka ove baze je automatsko rascjepkavanje (engl. *sharding*), odnosno tablice se distribuiraju na klasteru pomoću tzv. regija a regije se automatski dijele i redistribuiraju paralelno s rastom podataka.

Još jedna opcija koju pruža *MongoDB* je pretraživanje cjelovitog teksta. *MongoDB* podržava petnaest jezika i ima mogućnost određivanja korijena riječi i zaustavnih riječi (engl. *stopwords*). *MongoDB* ne omogućuje transakcije već umjesto njih pruža neke alternative od kojih nijedna nije u potpunosti zadovoljavajuća stoga nije preporučljivo koristiti *MongoDB* u tu svrhu. Pruža bogato sučelje upita s pokazivačima te koncepte „lomljenja“ i dohvaćanja skupova podataka sličnih SQL-u. Postoje doslovno tisuće načina za pretraživanje i specificiranje podataka pomoću upita. Većina se poslova obrade podataka prije verzije 2.2. oslanjala na *MapReduceu*. *MapReduce* je model programiranja i povezane implementacije za obradu i generiranje velikih skupova podataka s paralelno distribuiranim algoritmom na skupu. U verziji 2.2. dodana je veoma korisna mogućnost zvana „okvir sakupljanja“ (engl. *aggregation framework*) ili „cjevovod“ (engl. *pipeline*) stoga se *MapReduce* koristi u veoma rijetkim slučajevima kada su potrebne kompleksne funkcije za sakupljanje. Posebno moćna značajka *MongoDB* baze podataka je to što podržava geoprostorne indekse što znači da je moguće pohraniti ili geoJSON (format za kodiranje različitih geografskih struktura podataka) ili x i y koordinate unutar dokumenata [18].

Repliciranje u bazi *MongoDB* radi na sličan način na koji radi i repliciranje u relacijskim bazama podataka. Sve implementacije produkcije trebale bi biti skupovi replika koji se idealno sastoje od tri ili više poslužitelja koji posjeduju isti podatak. Ako je primarni poslužitelj nedostupan jedan od sekundarnih poslužitelja bit će izabran na njegovo mjesto [18].

Metlife, Expedia, Google, Buzzfeed, EA, Ebay, Bosch i Facebook samo su neke od velikih tvrtki koje za neke od svojih usluga koriste *MongoDB*.

MongoDB koristi *MongoDB Wire Protocol*. To je jednostavni zahtjev-odgovor protokol koji je baziran na *socketima*. Klijenti komuniciraju s bazom podataka putem običnog TCP/IP (*Transmission Control Protocol*) protokola.

2.3.3. Hbase

Hbase je ne-relacijska, distribuirana baza podataka otvorenog koda. Osnovna je karakteristika *Hbase* baze podataka rad s veoma velikim tablicama, odnosno rad s tablicama koje se sastoje od milijardi redaka i milijuna stupaca [19]. *Hbase* je baza podataka sa stupčastim modelom podataka a jedna od njenih najvažnijih prednosti je brz pristup podacima koji su raštrkani, što je od velike pomoći kada se radi o bazi u kojoj je pohranjeno nekoliko terabajta podataka a potrebno je pronaći primjerice 10 najvećih vrijednosti u tablici. Dakle, *Hbase* se koristi kada je potreban nasumični *read/write* pristup velikoj količini podataka u realnom vremenu. *Hbase* ne treba koristiti za male tablice jer za to postoje mnogo bolja rješenja koja će posao odraditi mnogo brže. Nastala je prema Googleovom *Bigtable* sustavu za distribuiranu pohranu strukturiranih podataka. *Bigtable* je distribuirani sustav za pohranu i upravljanje strukturiranim podacima koji je dizajniran kako bi mogao doseći veoma velike veličine [20]. Mnogi Googleovi projekti spremaju podatke u *Bigtable* sustav, uključujući *Google Earth, Google Finance* i web indeksiranje. Te aplikacije postavljaju vrlo različite zahtjeve *Bigtableu*, kako u pogledu veličine podataka tako i u pogledu latencija zahtjeva no unatoč tim raznovrsnim zahtjevima *Bigtable* uspješno pruža fleksibilna rješenje visokih performansi za sve ove Googleove proizvode.

Hbase baza podataka stvorena je 2007. godine u tvrtki Powerset iz San Francisca. Od 2008. godine *Hbase* baza se razvijala kao Hadoopov pod-projekt, a od 2010. godine jedan je od najvažnijih projekata korporacije Apache Software Foundation [21]. Napisana je u programskom jeziku Java a predviđena je za rad na Hadoopu tj. na HDFS-u. Hadoop je besplatan, na programskom jeziku Java zasnovan *framework*, odnosno „okvir“ koji podržava obradu velikih skupova podataka u računalnom okruženju. To je dio projekta Apache pod pokroviteljstvom korporacije Apache Software Foundation. HDFS (*The Hadoop Distributed File System*) je distribuirani datotečni sustav zasnovan na programskom jeziku Java koji pruža podesivu i pouzdanu pohranu podataka a dizajniran je kako bi obuhvatio velike klastere poslužitelja. *Hbase* je licenciran od strane Apachea.

Hbase organizira podatke u tablice [22]. Imena tablica su znakovni nizovi i sastoje se od znakova koji su sigurni za upotrebu u datotečnom sustavu. Unutar tablice, podaci se pohranjuju prema redcima. Redci se identificiraju jedinstveno prema ključu retka. Ključevi redaka nemaju tip podatka već se uvijek tretiraju kao nizovi bajtova. Podaci unutar redova se grupiraju pomoću tzv. „obitelji“ stupaca. „Obitelji“ stupaca su nizovi znakova koji su sigurni za upotrebu u datotečnom sustavu. Oni utječu na fizički raspored podataka pohranjenih u *Hbase* bazi podataka, zbog čega se moraju unaprijed definirati i ne mogu se lako mijenjati. Svi redci u tablici imaju iste „obitelji“

stupaca iako redak ne mora pohranjivati podatke u sve svoje „obitelji“. Podaci unutar „obitelji“ oslovljavaju se prema svojim kvalifikatorima stupaca. Kvalifikatori stupaca se ne moraju odrediti unaprijed i ne moraju biti konzistentni između redaka. Kao i ključevi redaka, kvalifikatori stupaca nemaju određen tip podatka i uvijek se tretiraju kao nizovi bajtova. Kombinacija ključa retka, „obitelji“ stupaca i kvalifikatora stupaca jedinstveno identificira polje. Podaci koji su pohranjeni u polju predstavljaju vrijednost polja. Vrijednosti također nemaju tip podatka i uvijek se tretiraju kao nizovi bajtova. Nad vrijednostima u poljima provodi se verzioniranje (proces pridavanja jedinstvenog imena ili jedinstvenog broja). Nijedan podatak u polju se ne mijenja niti se presnimljuje već se za svaku izmjenu stvaraju nove verzije polja. Vrijednostima se dodjeljuje jedinstveni broj koji je u ovom slučaju vremenska oznaka (engl. *timestamp*) koja označava kada je polje nastalo. Broj verzija vrijednosti sadržanih u polju je konfiguriran za svaku „obitelj“ stupaca. Zadani broj verzija polja je tri [22]. Za razliku od relacijskih baza podataka za spremanje NULL vrijednosti u *Hbaseu* nije potreban prostor za pohranu. *Hbase* podržava atomske operacije. Što se tiče tipova podataka, *Hbase* može pohraniti kao vrijednost sve što se može konvertirati u niz bajtova, npr. znakovni nizovi, brojevi, kompleksni objekti ili čak slike. Jedan od tipova podataka koji *Hbase* podržava su i brojači (engl. *counters*). Kod postavljanja upita SCAN operacija se može izvršiti na određenom rasponu polja s definiranim početnim i završnim ključem retka. Kao rezultat upita GET operacija vraća skup povezanih polja. Podaci su denormalizirani za vrijeme spremanja u bazu podataka. Naredbe PUT i DELETE se prvo spremaju u unutarmemorijsku strukturu zvanu *memstore*. Prije nego se *memstore* ažurira izmjene se pohranjuju u WAL-u (*Write Ahead Log*) kako bi se omogućio oporavak u slučaju „rušenja“ poslužitelja [22].

Neke od značajki *Hbasea* su fleksibilni model podataka, linearna i modularna skalabilnost, dobar i za korištenje jednostavan Java API (*Java Application Programming Interface*), obrada u realnom vremenu, mogućnost praćenja stanja baze preko Ambari ili Ganglia dodatka i dr. *Hbase* je veoma konzistentna baza i omogućuje automatsko rascjepkavanje (engl. *sharding*). Ne podržava dodatne mogućnosti koje nude sustavi za upravljanje relacijskim bazama podataka kao što su klasifikacija stupaca, sekundarni indeksi, transakcije, napredni upiti itd. [22].

Dva su načina pristupanja i komuniciranja s *Hbase* poslužiteljem. Klijenti mogu birati hoće li mu pristupiti s Thrift protokolom ili s REST protokolom. Thrift je softverski okvir koji omogućuje stvaranje međujezičnih veza, točnije on konvertira druge programske jezike pomoću Java klijenta. Kako bi funkcionirali, i Thrift i REST protokol zahtijevaju korištenje takozvanih *Hbase daemona*, odnosno pozadinskih procesa koji će upravljati zahtjevima za uslugama kao što

su primjerice prijenos datoteka i sl., a koji miruje kada je nepotreban [23]. Oni se mogu instalirati s paketima *Hbase* Thrifta i *Hbase* REST-a.

Hbase bazu podataka, za neke od svojih usluga, koriste mnoge korporacije od kojih su najpoznatije svakako Micro, Ebay, Yahoo!, Facebook, RocketFuel, Flurry i dr.

2.3.4. Cassandra

Cassandra je distribuirana baza podataka sa stupčastim modelom podataka otvorenog koda namijenjena za upravljanje velikim količinama strukturiranih podataka preko većeg broja poslužitelja. Nastala je u svrhu rješavanja problema pretraživanja *Inboxa,* odnosno primljenih poruka, na Facebooku, a stvorena je na temelju Amazonovog *Dynamoa* i Googleovog *BigTablea* [24]. Namijenjena je za korištenje u situacijama kada je potrebna visoka skalabilnost i dostupnost bez kompromitiranja performansi. *Cassandru,* za neke od svojih usluga, koriste korporacije kao što su Ebay, Netflix, Sky, The New York Times, Instagram, GitHub, Reddit, Nasa i mnoge druge.

Cassandra je napisana u programskom jeziku Java a licencirana je od strane *Apachea.* Koristi protokol CQL3 (*The Cassandra Query Language*) koji je zapravo primarni jezik za komunikaciju s *Cassandra* bazama podataka. Također podržava i Thrift protokol. Dizajnirana je na način da može upravljati s veoma velikom količinom podataka i s tisućama istodobnih korisnika i operacija u sekundi pa čak i preko nekoliko centara podataka. Jednako je efikasna i u upravljanju s mnogo manjom količinom podataka i korisnika.

Svi čvorovi u *Cassandri* imaju identičnu ulogu i ne postoji koncept nadređenog „*master*“ čvora već svi čvorovi međusobno komuniciraju na istoj razini. Dakle, svi su čvorovi ravnopravni a svaki čvor ima funkcionalnost sustava za upravljanje bazama podataka. Podaci se automatski distribuiraju kroz sve čvorove iz klastera. Ukoliko neki čvor otkaže, kopije podataka su dostupne na drugim čvorovima u klasteru [24].

Neke od osnovnih značajki *Cassandre* su mogućnost pohrane velikih skupova podataka i optimizirana obrada istih, stalna dostupnost te dinamični model podataka. *Cassandra* je visoko skalabilna (linearna skalabilnost), odnosno kapacitet joj je lako proširiv dodavanjem novih čvorova. Ona predstavlja pouzdan sustav koji jamči sigurnost podataka. Ukoliko dođe do otkazivanja rada čvorova oni će biti zamijenjeni bez vremenskih zastoja. Što se tiče trajnosti podataka, ona je također osigurana i to na način da se izmjene najprije dodaju u tzv. *commit logove,* odnosno zapise transakcija, pa tako jednom učinjene izmjene ostaju očuvane čak i ako dođe do otkazivanja hardvera [24].

2.3.5. Neo4j

Neo4j je baza podataka s graf modelom podataka otvorenog koda koja je napisana u programskom jeziku Java a razvijena je od strane Neo Technologya. Većinskim je dijelom licencirana od strane GPL-a (*General Public License*) a neki dijelovi su licencirani od strane AGPL-a (*Affero General Public License*). Zahvaljujući svojoj stabilnosti, izdržljivosti, brzini, skalabilnosti i visokoj dostupnosti *Neo4j* je vodeća baza podataka s graf modelom podataka u svijetu.

Koncept pohrane podataka u *Neo4j* bazu podataka sastoji se od čvorova (engl. *nodes*) koji su zapisi graf podataka, veza (engl. *relationships*) koje povezuju čvorove i osobina (engl. *properties*) koje su imenovane vrijednosti podataka. Podaci se pohranjuju kao osobine čvora a osobine se predstavljaju korištenjem jednostavnih parova imena i vrijednosti. Čvorovi se mogu grupirati korištenjem oznaka, odnosno graf konstrukcija koje se koriste za grupiranje, pa tako svi čvorovi označeni istom oznakom pripadaju istom skupu. Oznake su opcionalne pa tako čvorovi ne moraju biti označeni ili mogu biti označeni s više oznaka. S obzirom na to da se mnogi upiti mogu raditi nad skupovima čvorova umjesto nad cijelim grafom to upite čini efikasnijima i jednostavnijima za kreiranje [25]. Veze su zapisi podataka koji mogu imati vlastita svojstva pri čemu veze između čvorova uvijek imaju smjer i tip.

Osnovna svojstva *Neo4j* baza podataka su mogućnost istinske ACID transakcije, visoka dostupnost, skaliranje do milijardi čvorova i veza, visoka brzina postavljanja upita i deklarativni jezik za postavljanje upita [26]. Za komunikaciju klijenta s poslužiteljem *Neo4j* koristi protokol HTTP/REST. Za neke od svojih usluga *Neo4j* bazu podataka koriste korporacije kao što su Ebay, Walmart, Telenor, Linkedin, Cerved i dr.

2.4. Yahoo! Cloud Service Benchmark (YCSB)

Yahoo! Cloud Service Benchmark (*YCSB*) je alat otvorenog koda koji je kroz zadnjih nekoliko godina prepoznat kao jedno od najboljih rješenja za testiranje i vrednovanje baza podataka, odnosno oblaka računala. Najčešća upotreba *YCSB*-a je ispitivanje više sustava i njihova usporedba pa se tako npr. može instalirati više sustava na istoj hardverskoj konfiguraciji nad kojima će se provesti isti scenariji opterećenja kako bi naposljetku bilo moguće uspoređivati njihove karakteristike.

Sastoji se od *YCSB* klijenta, generatora opterećenja, te seta osnovnih scenarija opterećenja. *YCSB* klijent je Java program za generiranje podataka koji će se dodavati u bazu podataka i za

generiranje operacija, koje čine scenarij opterećenja (engl. *workload*). Izvršitelj scenarija opterećenja upravlja višestrukim nitima (engl. *thread*) klijenta. Svaka nit izvršava sekvencijalnu seriju operacija tako što poziva sloj sučelja baze podataka kako bi se baza podataka učitala (faza učitavanja) i kako bi se izvršio scenarij opterećenja (faza transakcije). Niti reguliraju stopu kojom se reguliraju zahtjevi tako da je moguće direktno kontrolirati ponuđeno opterećenje na bazu podataka. Niti također mjere latenciju i postignutu propusnost svojih operacija te šalju ta mjerenja u statistički modul. Na kraju, statistički modul skuplja mjere i izvještava o prosjecima, o 95 i 99 percentilnim latencijama te daje ili histogram ili vremenske serije latencije [27].

YCSB klijent ima niz svojstava koja definiraju njegov rad. Ta se svojstva dijele na dvije skupine i to na svojstva scenarija opterećenja i svojstva izvršenja. Svojstva scenarija opterećenja su svojstva koja definiraju scenarij opterećenja neovisno o zadanoj bazi podataka ili o samom eksperimentu a to su primjerice spoj čitanja i zapisa, distribucija koja se koristi, veličina i broj polja u zapisu i dr. Svojstva izvršenja su svojstva specifična za određeni eksperiment kao što su npr. baza podataka koja će se koristiti (*Redis*, *Cassandra*, *MongoDB* itd.), svojstva koja će se koristiti kako bi se pokrenula ta baza (npr. *hostnames*), broj niti klijenta itd. Dakle, svojstva scenarija izvršenja su nepromjenjiva i koriste se za mjerenje različitih baza podataka dok se svojstva izvršenja razlikuju od eksperimenta do eksperimenta ovisno o bazi koja će se koristiti, o ciljanoj propusnosti i sl. [27].

Set osnovnih scenarija opterećenja može dati kvalitetnu sliku o performansama sustava, no bez obzira na to on je proširiv tj. pruža mogućnost dodavanja proizvoljnih scenarija opterećenja kako bi se ispitali aspekti sustava ili scenariji aplikacije koji nisu obuhvaćeni u setu osnovnih scenarija [27]. Također, *YCSB* je proširiv i što se tiče podrške ispitivanja različitih baza podataka pa je tako uz baze podataka *Redis* i *Hbase*, za koje postoji primjerak koda na klijentu, poprilično jednostavno provesti i ispitivanja na drugim bazama podataka [28].

Kako bi se provelo testiranje opterećenja putem *YCSB* alata potrebno je prvo instalirati i podesiti odgovarajući sustav za upravljanje bazom podataka. Podešavanje u suštini predstavlja stvaranje poznatih „šablonskih tablica“ (ovisi o modelu podataka te o scenariju opterećenja), u koje će sam *YCSB* alat generirati podatke. Potom slijedi odabir ispravnog sloja sučelja baze podataka ostvaren *Java* razredom koji će izvršavati čitanje, pisanje, ažuriranje, brisanje i pretraživanje. Sljedeći korak je izbor scenarija opterećenja iz osnovnog seta ili stvaranje proizvoljnog (primjer seta osnovnih opterećenja nalazi se u tablici 3.4.). Nakon odabira željenog opterećenja slijedi odabir prikladnih parametara izvršenja (broj klijentskih niti, ciljana propusnost, itd.). Zadnji koraci koji se provode su stvaranje, odnosno učitavanje željene količine podataka u bazu te pokretanje same simulacije [29].

2.5. ARM arhitektura

ARM je linija mikroprocesora koje razvija tvrtka ARM Holdings od osamdesetih godina dvadesetog stoljeća pod imenima *Acorn RISC Machine* ili kasnije *Advanced RISC Machine*. Prvi ARM prototip predstavljen je godine 1985. Tijekom vremena ARM arhitektura se razvijala kako bi zadovoljila rastuće zahtjeve za novom vrstom funkcionalnosti, za svojstvima integrirane sigurnosti, za visokim performansama i potrebama novih tržišta te tržišta koja su tek u nastajanju. ARM arhitektura podržava implementacije u širokom rasponu točaka performansi, što ju čini vodećom arhitekturom u mnogim segmentima tržišta. Arhitekturalna jednostavnost ARM procesora vodi do jako malih i do vrlo učinkovitih implementacija naprednog dizajna koje omogućuju vrlo malu potrošnju energije. Osnovne značajke ARM arhitekture su veličina implementacije, izvedba i niska potrošnja [30].

ARM arhitektura je zastupljena kod procesora namijenjenih za rad s tzv. *lighweight* zadacima (nema potrebe za obradom velike količine podataka te brze obrade istih). Glavna primjena ARM procesora je u raznim mikro-upravljačima te pametnim telefonima. ARM arhitektura koristi skraćeni skup instrukcija (engl. *Reduced Instruction Set Computer*), te kao takav ima tipične karakteristike kao što su: velike i uniformirane datoteke registara, *load/store* arhitektura (operacije za obradu podataka rade sa sadržajem registara a ne direktno sa sadržajem memorije), jednostavne načine adresiranja (*load/store* adrese određene samo sadržajem registara i poljem instrukcija) te jedinstveno polje instrukcija s fiksnom duljinom za pojednostavljenje dekodiranja instrukcija [30]. ARM arhitektura omogućuje kontrolu nad aritmetičko-logičkom jedinicom i nad posmačnim registrom u svakoj instrukciji nad obradom podataka kako bi se povećala njihova upotreba. Još neke od karakteristike ARM arhitekture su automatsko povećanje (engl. *auto-increment*) i automatsko smanjenje (engl. *auto-decrement*), odnosno načini na koji se provodi adresiranje za optimizaciju programske petlje, potom povećanje propusnosti podataka putem *load/store* višestrukih instrukcija, te uvjetno izvršavanje svih instrukcija za povećanje propusnosti izvršenja [30].

ARM procesor podržava sljedeće tipove podataka: bajtove, polu-riječi (engl. *halfword*) i riječi (engl. *word*). ARM ima trideset i jedan, opće-namjenski, 32-bitni registar. Istodobno je vidljivo šesnaest registara, dok se ostali registri koriste za ubrzanje obrade iznimki. Svi specifikatori registara u ARM instrukciji mogu adresirati bilo koji od tih 16 registara. Dva od šesnaest vidljivih registara imaju posebne uloge. Jedan od njih je registar poveznica (engl. *link registar*) koji sadrži adresu sljedeće instrukcije poslije *branch and link* instrukcija koje se koriste za pozivanje sub-rutina. Ostatak se vremena registar poveznica koristi kao registar opće namjene.

Drugi je programsko brojilo (engl. *program counter*) koji se koristi u instrukcijama kao pokazivač na instrukciju koja je dvije instrukcije poslije instrukcije koja se izvršava. Ostalih 14 registara nemaju posebnu hardversku namjenu te je njihova funkcija definirana izričito preko softvera [30].

ARM podržava sedam tipova iznimki a to su resetiranje (engl. *reset*), nedefinirana instrukcija, programski prekid (engl. *software interrupt*), dohvat krive instrukcije (engl. *instruction fetch memory abort*), krivi dohvat podatka (engl. *data access memory abort*), prekid (IRQ) i brzi prekid (FIQ). Svaka iznimka ima svoj prioritet po kojima se odvija redoslijed obrade pa tako najveći prioritet ima resetiranje, dok najmanji prioritet ima nedefinirana instrukcija [30].

Iznimke se mogu generirati interno ili eksterno kako bi uzrokovale da procesor obradi određeni slučaj kao što su primjerice eksterno generiran prekid (engl. *interrupt*) ili pokušaj izvršavanja nedefinirane instrukcije. Stanje procesa neposredno prije obrade iznimke mora biti sačuvano, tako da se originalni program može nastaviti kada rutina za obradu iznimke završi s radom. Odjednom se može dogoditi i više od jedne iznimke. Kada se dogodi iznimka, izvršavanje se poziva s fiksne memorijske adrese, koja odgovara tipu iznimke. Te fiksne memorijske adrese zovu se vektori iznimki (engl. *exception vectors*) [30].

Kako bi se poboljšale performanse u ARM memorijskim sustavima koristi se priručna memorija ili predmemorija (engl. *cache*).

3. ISTRAŽIVANJE

3.1. Cilj i metodologija istraživanja

Cilj ovog rada je mjerenje i usporedba kvalitete usluge u ne-relacijskim distribuiranim bazama podataka ograničenih resursa, odnosno na računalnim sustavima ograničenih resursa. Kvaliteta usluge bit će mjerena u raznim uvjetima, simulacijom stvarnog sustava te specifičnih opterećenja i zahtjeva na isti.

Koncept kvalitete usluge, najjednostavnije se može predočiti kao skup karakteristika sustava koje utječu na njegovu kvalitetu. Formalna definicija kao i opća suglasnost o točnom značenju pojma kvalitete usluge ne postoji. Kvaliteta usluge može se definirati kao skup kvantitativnih i kvalitativnih karakteristika nužnih za ostvarenje potrebnih funkcionalnosti aplikacije. Kvaliteta usluge danog sustava izražava se kao skup parova parametar-vrijednost, a svaki se parametar sagleda kao klasificirana varijabla čije vrijednosti variraju [31]. Koncept kvalitete usluge u ne-relacijskim bazama podataka, oslanja se na nekoliko osnovnih parametara koji utječu na performanse i koji se koriste u mjerenjima kvalitete usluge. Osnovni parametri koji utječu na kvalitetu usluge u ne-relacijskim bazama podataka i koji će se mjeriti u ovom istraživanju su propusnost baza (eng. *throughput*) te vrijeme odgovara na zahtjev (eng. *latency*).

Propusnost je definirana kao broj specifičnih operacija (npr. čitanja i/ili pisanja) koje baza podataka može obaviti u jedinici vremena. Uobičajeno je da se propusnost mjeri u operacijama po sekundi, stoga će se i u ovom istraživanju mjeriti broj operacija koje se obrade u jednoj sekundi. Vrijeme odgovora na zahtjev označava vrijeme koje protekne od slanja zahtjeva do primitka uspješnog odgovora od strane baze podataka. Vrijeme odgovora na zahtjev najčešće se mjeri kao prosjek ili percentil vremena odgovora za određen broj operacija [32]. Konkretno, bit će mjereno prosječno vrijeme, minimalna vremena, maksimalna vremena, 95. percentil, te 99. percentil odgovora na zahtjev određenog broja mjerenja. Ni propusnost ni vrijeme odgovora na zahtjev nisu u potpunosti precizne metrike. Za mnoge baze podataka propusnost može varirati ovisno o tipu operacije. Slično tomu, vrijeme odgovora na zahtjev može varirati ovisno o načinu mjerenja ali i ovisno o različitim utjecajima (vanjski i unutarnji čimbenici). Vrijeme odgovora na zahtjev može se mjeriti kao vrijeme koje protekne od slanja zahtjeva s klijentske strane do primitka odgovora na istoj a može se mjeriti i kao vrijeme koje protekne od trenutka kada poslužitelj baze podataka primi zahtjev do trenutka kada se zahtjev postavi u red čekanja zahtjeva. Ovaj prvi primjer mjerenja vremena odgovora iz perspektive klijenta aplikacije najrealnija je metrika. Međutim, ta metrika uključuje i vrijeme odgovora mreže između klijenta i poslužitelja te baze podataka [32].

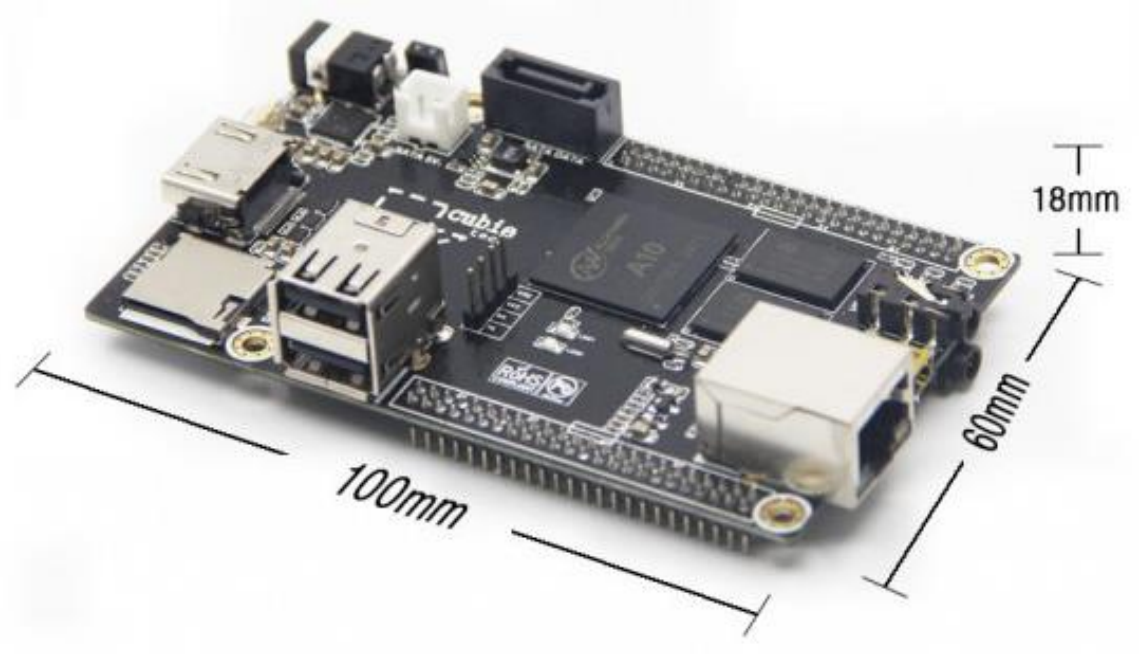
U ovom je istraživanju korišten drugi primjer mjerenja vremena odgovora, iz perspektive baze podataka, jer predstavlja točnije mjerenje samih tehničkih mogućnosti baze podataka. Ta metrika uključuje vrijeme obrade same operacije kao i vrijeme koje zahtjev provodi u redu čekanja izvršenja [32]. Propusnost i vrijeme odgovora mjeri se ovisno o promjeni broja niti što je posljednji faktor kvalitete usluge mjerene u ovom istraživanju. Svaki scenarij, odnosno proces opterećenja raspoređen je na manje dijelove, tj. razdijeljen je na niti. Povećanjem broja niti, u teoriji, trebalo bi se smanjivati vrijeme izvršenja određenog procesa, no u praksi to nije uvijek slučaj.

Cubieboard, prikazan na slici 3.1. te slici 3.2. predstavlja računalni sustav ograničenih resursa na kojem će se obavljati dio simulacije i mjerenja. *Cubieboard* pripada skupini tzv. računala na jednoj pločici (eng. *SBC – single board computer*), odnosno kompletnih računala malih dimenzija, cijena i potrošnje, te prilagodljive primjene. Među njima su uz *Cubieboard* još i *Arduino, BeagleBoard, Raspberry Pi*, itd. Iako ih karakterizira niska cijena i potrošnja, nerealno je očekivati velike performanse, ali zato to nadomještaju praktičnošću. Važna karakteristika *Cubieboard* sustava je procesor, odnosno ARM arhitektura na kojoj se zasniva.

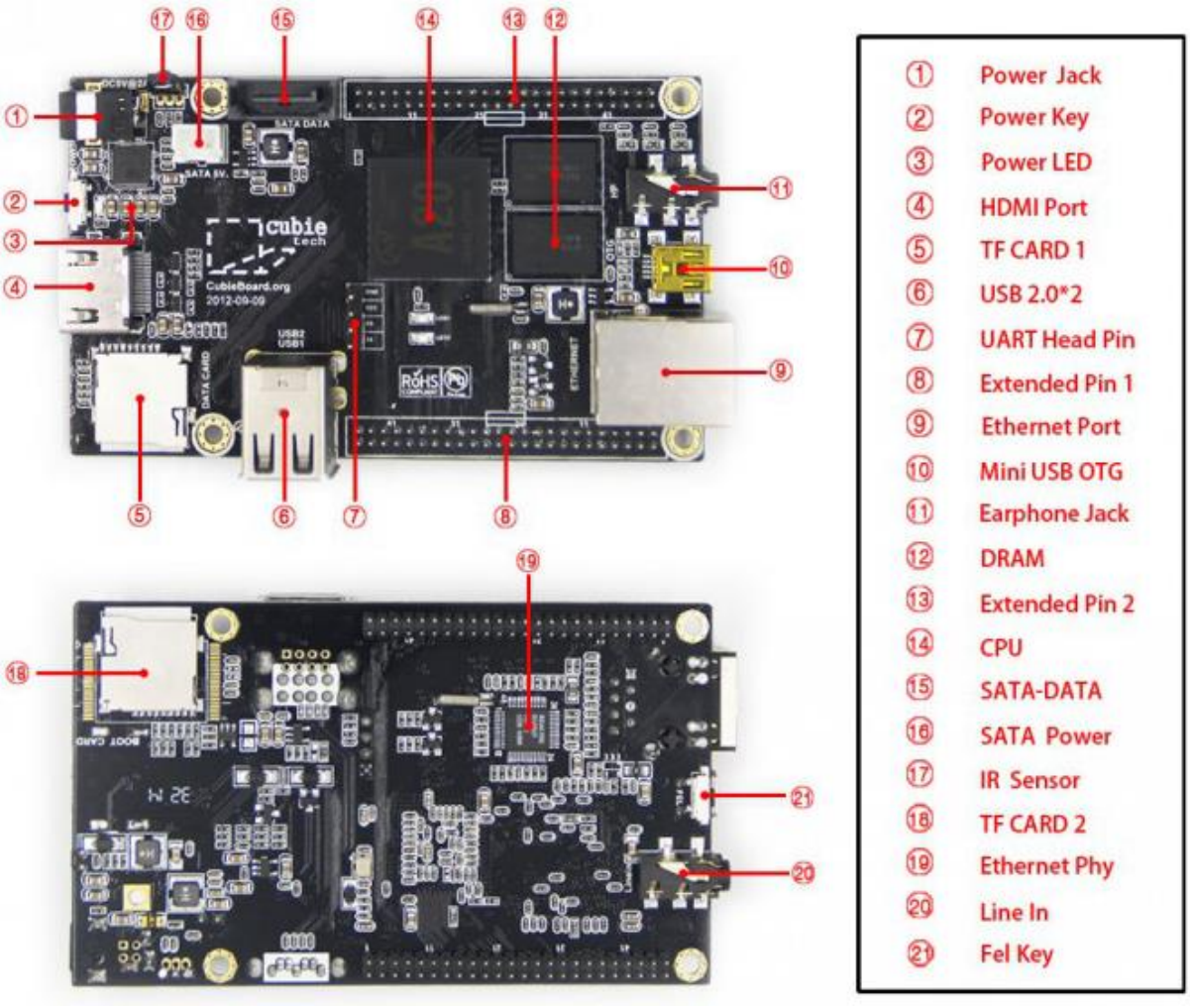
Cubieboard1 je prva generacija hardvera otvorenog koda (eng. *open source hardware*) dostupnog svima izdana od strane Cubietech Limiteda [32]. Kao operativni sustav koristi razne verzije Linux distribucija (Debian, Ubuntu, itd.), te Android sustava. Uz to što se može koristiti kao razvojni sustav, može se koristiti i kao upravljač (eng. *controller*) ili jezgra nekakvog sustava.

Cubieboard2, prikazan na slici 3.2. predstavlja drugu generaciju, odnosno nadogradnju *Cubieboard1* s novim dvojezgrenim procesorom [33]. Koristi identični dizajn tiskane pločice PCB (eng. *printed circuit board*) kao *Cubieboard1* te slične operacijske sustave, stoga jedinu razliku među njima predstavlja sam procesor kao što je vidljivo u tablici 3.1.

U tablici 3.2. prikazane su karakteristike prijenosnog osobnog računala *HP650* (nadalje u tekstu *HP650*) relativno slabih performansi na kojem će se također odrađivati ranije navedena mjerenja propusnosti te vremena odgovora na odziv.



Sl. 3.1. Prikaz *Cubieboard1* sustava [10]



Sl. 3.2. Prikaz *Cubieboard2* sustava i prikaz komponenti [33]

Tab. 3.1. Specifikacije *Cubieboard1* i *Cubieboard2* sustava [32,33]

	Cubieboard1	***Cubieboard2***
Procesor	AllWinner A10 ARM Cortex-A8 1GHz, ARM Mali 400 (Compiles with OPENGL ES 2.0/1.1)	AllWinner A20 ARM Cortex-A7 Dual-Core 1GHz, ARM Mali 400MP2 (Compiles with OPENGL ES 2.0/1.1)
Radna memorija	1GB DDR3@480MHz (960MTPS)	
Memorijski prostor	NAND Flash, 4GB	
Dodatni memorijski prostor	SATA - podržava 2.5“ HDD/SSD veličine do 2TB	
TF kartica	Micro SD card slot veličine do 32GB	
Video izlaz	HDMI Port A, HDMI V1.4a, podržava 1080P@60Hz rezolucijski izlaz	
Ethernet	10M/100M RJ45	
Napajanje	DCIN 5V@2.5A, podržava USB napajanje	

Tablica 3.2 Specifikacije prijenosnog osobnog računala *HP650*

	Notebook HP650
Procesor	Intel Pentium B980 Dual-Core 2.4GHz
Radna memorija	4GB DDR3@1333MHz
Memorijski prostor	500GB SATA HDD 5400rpm
Ethernet	Gigabit Ethernet

Kako bi se ostvario cilj ovog rada bilo je potrebno odabrati i adekvatne ne-relacijske distribuirane baze podataka. Prvotno su izabrane tri najpopularnije [5], odnosno tri najčešće korištene ne-relacijske baze podataka s različitim modelima podataka: *MongoDB*, kao baza podataka s dokument modelom podataka, *Cassandra*, kao baza podataka sa stupčastim modelom podataka, te *Redis*, kao baza podataka s ključ-vrijednost modelom podataka.

Međutim, tu dolazi do poteškoća s ranije spomenutom ARM arhitekturom procesora *Cubieboard2* sustava. Nakon nekoliko neuspjelih pokušaja instaliranja, pokretanja i podešavanja *Cassandra* baze podataka, gdje je u najboljem slučaju ispravno funkcionirao samo dio traženih funkcija na red dolazi sljedeća po popularnosti, *Hbase* baza podataka sa stupčastim modelom podataka. Također, do problema je došlo i zbog nedostatne kompatibilnosti *Cassandra* baze podataka i *YCSB* klijenta na korištenom *HP650*. Baze podataka s graf modelom podataka nisu

uključene u istraživanje zbog nedostatka podrške i zbog relativne nekompatibilnosti s *YCSB* klijentom. Glavna svojstva i karakteristike odabranih baza nalaze se u tablici 3.3., a detaljne karakteristike nalaze se u teorijskom dijelu rada, točnije u poglavlju 2.3.

Tab. 3.3. Usporedni prikaz karakteristika baza podataka *Redis, MongoDB* i *Hbase* [34, 35]

	Redis	***MongoDB***	***Hbase***
Model podataka	Baza podataka s ključ-vrijednost modelom podataka (engl. *key-value database*)	Baza podataka s dokument modelom podataka (engl. *document database*)	Baza podataka sa stupčastim modelom podataka (engl. *wide column database*)
Napisan u prog. Jeziku	C	C++	Java
Licenca	BSD	AGPL	Apache
Protokol	RESP (REdis Serialization Protocol)	MongoDB Wire Protocol	HTTP/REST & Thrift
Način repliciranja	*Master-slave*	*Master-slave*	*HDFS repliciranje*
Glavno svojstvo	Izuzetno brza	Robusna, velike mogućnosti kontorliranja upita	Velike tablice-milijarde redaka i milijuni stupaca
Istovremenost (engl. *concurrency*)	Pošto se gotovo sve *Redis* operacije izvršavaju preko jedne niti, sve operacije se izvode sekvencijalno, te su transakcije zaštićene do kraja svog izvršenja bez obzira na uspjeh	Ažuriranje na mjestu, zaključavanje dokumenata, kontrola pristupa dokumentima	Višestruko verzioniranje polja (eng. *Multiversion*), sustav kontrole zaključavanja redaka (eng. *control-row locks*)

Za ostvarenje mjerenja bit će korišten *Yahoo! Cloud Service Benchmark* (*YCSB*) alat koji je također ranije detaljnije opisan u teorijskom dijelu rada, točnije u poglavlju 2.4. Bit će korišteni zadani scenariji opterećenja samog *YCSB* klijenta, prikazani u tablici 3.4. Važno je napomenuti da je scenarij opterećenja „E“ preskočen zbog nemogućnosti kompletiranja mjerenja na *Cubieboard2* sustavu, no o tome više u nastavku rada.

Zbog raznih prepreka i nemogućnosti, na *Cubieboardu2* su uspješno određena samo mjerenja za *Redis* bazu podataka, što će biti pojašnjeno u nastavku rada, odnosno u analizi rezultata. Sukladno tomu bit će provedena analiza rezultata u dvije kategorije. U prvoj će kategoriji biti predstavljena usporedba *Cubieboard2* i *HP650* rezultata mjerenja na *Redis* bazi podataka dok će u drugoj kategoriji biti predstavljena usporedba performansi sve tri skupine baza podataka dobivenih mjerenjem na *HP650*.

Tab. 3.4. Osnovni set opterećenja *YCSB* alata [27]

Scenarij opterećenja	Operacije	Algoritam izbora podataka
A – „teško“ ažuriranje (eng. *Update heavy*)	Čitanje: 50% Ažuriranje: 50%	Zipfian[1]
B – „teško“ čitanje (eng. *Read heavy*)	Čitanje: 95% Ažuriranje: 5%	Zipfian
C – samo čitanje (eng. *Read only*)	Čitanje: 100%	Zipfian
D – čitanje „zadnjih“ (eng. *Read latest*)	Čitanje: 95% Pisanje: 5%	Latest[2]
E – kratki rasponi (eng. *Short ranges*)	Pretraživanje: 95% Pisanje: 5%	Zipfian/ Unifrom[3]
F – čitanje-modificiranje-pisanje (eng. *Read-modify-write*)	Čitanje: 50% Čitanje/Modificiranje/Pisanje: 5%	Zipfian

Prije samog mjerenja mogu se pretpostaviti sljedeći rezultati. Zbog cilja, namjene, karakteristika i najmanjih zahtjeva, od *Redis* baze podataka očekuje se najveća propusnost te najmanje latencije. *Hbase* bazu podatka karakterizira korištenje u slučajevima enormnog broja podatka, mjerenih u milijardama, što dovodi do zaključka da se na nekom *lighweight* sustavu s neusporedivo manjim brojem podatka ne može očekivati efikasnost, brzina niti propusnost. *MongoDB* baza podataka predstavlja jako pouzdano i popularno rješenje koje se također koristi u slučajevima enormnog broja podataka, ali zbog svoje sličnosti relacijskim bazama podataka predstavlja određen misterij što se tiče ishoda mjerenja. Od *Cubieboard2* računalnog sustava ne očekuje se da će biti dosljedan ulozi poslužitelja s ne-relacijskim bazama podataka zato što ARM arhitektura i same performanse nisu namijenjene za tako zahtjevne i/ili sigurnosno-kritične zadatke.

[1]Izbor podatka prema Zipfian distribuciji. Npr. podaci sadržani u „glavi“ distribucije bit će popularniji naspram onih u „repu“ [27]
[2] Slično Zipfian distribuciji, osim što se tek dodani podaci nalaze u „glavi“ distribucije [27]
[3] Uniformni slučajni izbor, svaki podatak ima jednaku šansu da bude izabran [27]

4. ANALIZA REZULTATA MJERENJA

4.1. Uporedba *Cubieboard2* i *HP650* rezultata mjerenja – *Redis* baza podataka

Kao što je spomenuto ranije, kod rada s *Cubieboardom2*, točnije kod same konfiguracije te kod pokušaja izvršenja mjerenja došlo je do mnogih prepreka i poteškoća. To je dovelo do odabira navedenih kategorija usporedbe i, što je najvažnije, do postavljanja parametara koji su se koristili za izvršenje samih mjerenja. Prvu prepreku predstavljalo je podešavanje samog *Cubieboarda2* za osnovni rad, odnosno pronalazak kvalitetne inačice operacijskog sustava. Zbog nedostatka podrške i repozitorija, relativno je teško instalirati i podesiti potrebno okruženje i alate. Problem su također predstavljale i same performanse *Cubieboarda2* koje, kao što je bilo i za očekivati, nisu na prihvatljivoj razini za kvalitetan poslužiteljski sustav. Sve je to rezultiralo nedostatkom ispunjenja zahtjeva koje *Cassandra, Hbase* i *MongoDB* imaju kao baze podataka specifičnih namjena.

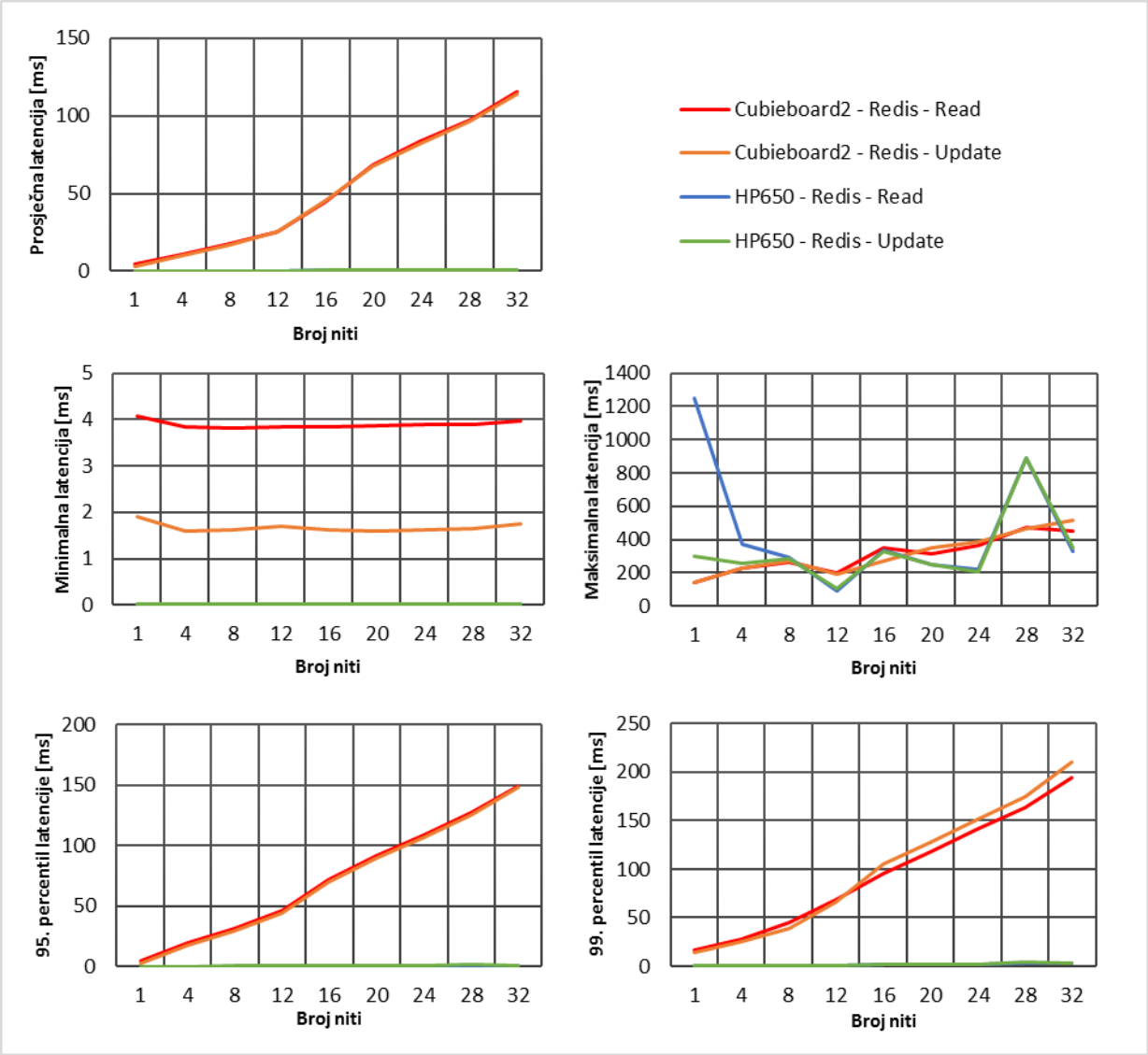
YCSB klijent, korišten za izvršenje samih mjerenja, odabran je s ciljem da se odabrane baze podataka, koje rade na u potpunosti drugačije načine, svedu na približno istu razinu kako bi se relativno realno mogle usporediti. Dakle, prvi parametar čine podaci koji se učitavaju u bazu i predstavljeni su kao zapisi veličine od 1 KB, odnosno 10 polja, svaki veličine od 100 bajta, plus ključ. Drugi parametar je količina zapisa. U slučaju korištenja *Redis* baze podataka, na *Cubieboardu2* je maksimalna količina zapisa oko osamsto tisuća, no tu se javlja mnogo problema te je zbog sigurnosti odabran broj od pola milijuna zapisa kao parametar koji će se koristiti u svim mjerenjima. Sljedeći parametar predstavlja količinu operacija koje će se odraditi u jednom mjerenju na ranije učitanim podacima. Zbog vremena trajanja samih mjerenja te učestalosti pojava prekida odabran je broj od dva milijuna operacija kao parametar koji će se koristiti u svim mjerenjima.

Važno je napomenuti da *Cubieboard2* sustav nije mogao „podnijeti“ niti jedno mjerenje koje je trajalo duže od dvanaest do petnaest sati. U tim je slučajevima bilo potrebno resetirati i pokrenuti cijeli sustav iz početka, što dovodi do problema nesigurnosti, odnosno mane nedostatka oporavka od pogreške.

U nastavku slijede rezultati mjerenja pomoću *YCSB* klijenta korištenjem zadanih scenarija koji pokušavaju replicirati realne događaje u radu samih poslužitelja i sukladno tome u bazama podataka. Kao što je spomenuto ranije, scenarij opterećenja „E“ nije bilo moguće odraditi na *Cubieboardu2* jer broj operacija po sekundi nije prelazio broj pet, što je dovodilo do

beskonačnih testova, a sa smanjenjem broja operacija ne bismo dobili realne podatke prikladne za usporedbu.

4.1.1. Scenarij opterećenja „A“



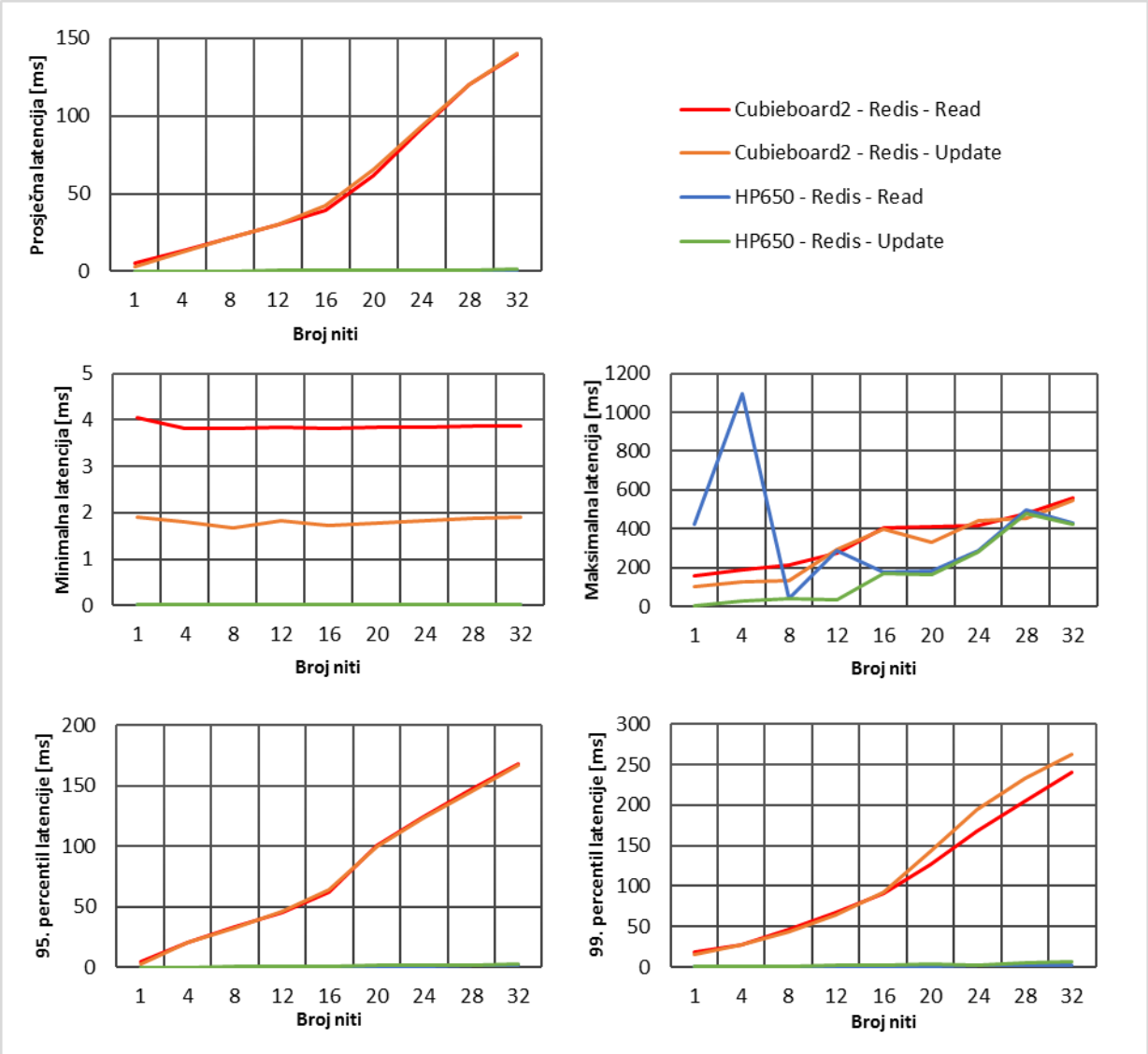
Sl. 4.1. Usporedba rezultata mjerenja „A“ scenarija opterećenja

Scenarij opterećenja „A“ može se poistovjetiti s periodom u kojem poslužitelj prolazi kroz sve izmjene koje su se dogodile u tome periodu te pohranjuje iste. Čini ga 50% operacija čitanja (eng. *Read*) te 50% operacija ažuriranja (eng. *Update*).

Iz grafova sa slike 4.1. vidljivo je da se *Cubieboard2* sustav ne nosi dobro s povećanjem broja niti. Također je vidljivo da je maksimalna latencija jednaka u mjerenjima na oba sustava što je veoma zanimljivo, dok je minimalna latencija neusporedivo manja kod mjerenja na *HP650* što je bilo očekivano. Bitno je napomenuti da se sve točne vrijednosti rezultata mjerenja nalaze u

prilogu P.4.1., jer iz ponekih grafova nije jasno vidljiva točna vrijednost (npr. na grafu prosječne latencije rezultati dobiveni mjerenjem na *HP650* izgledaju kao da teže k nuli, što nije točno).

4.1.2. Scenarij opterećenja „B“

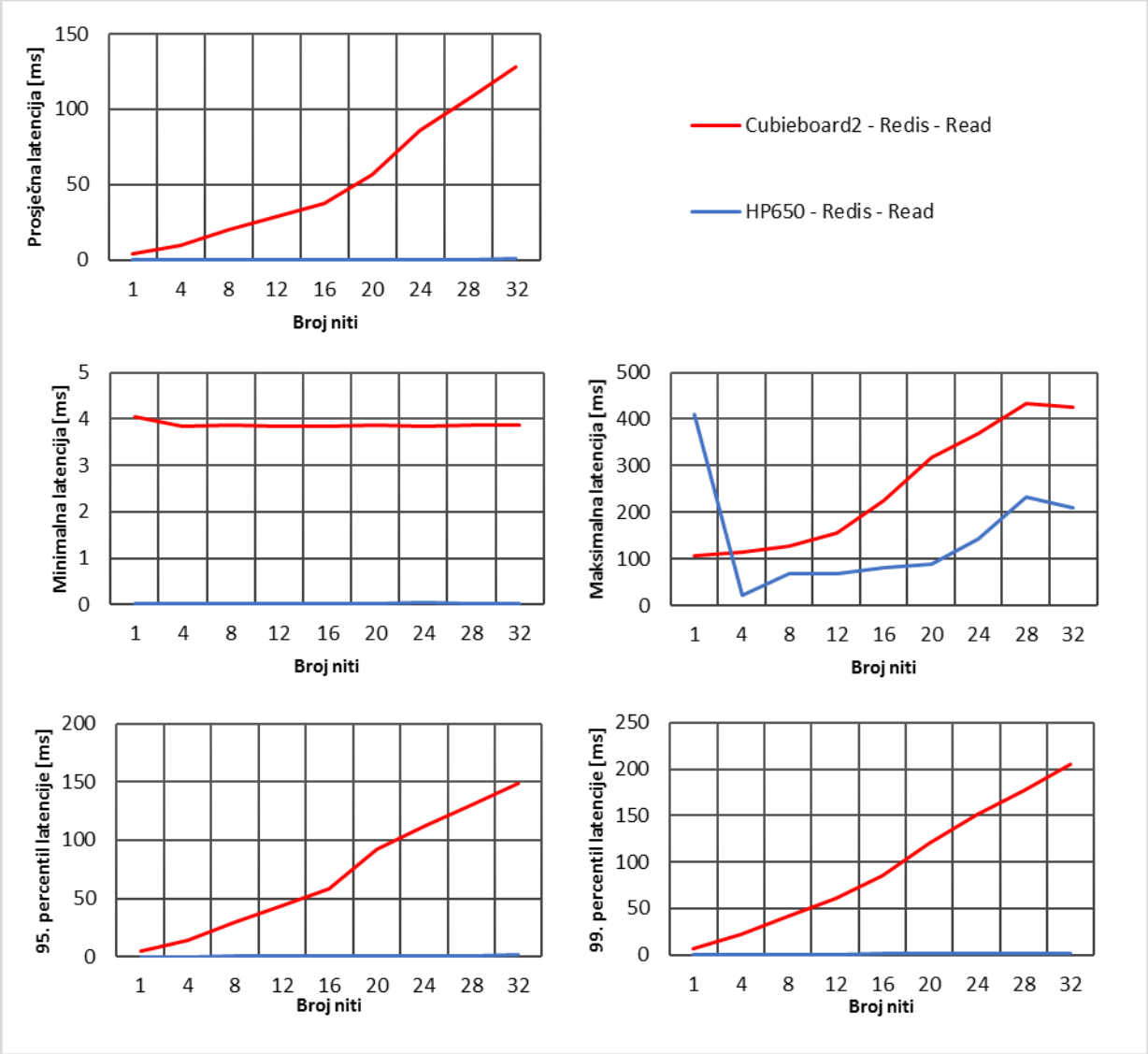


Sl. 4.2. Usporedba rezultata mjerenja „B“ scenarija opterećenja

Scenarij opterećenja „B“ može se poistovjetiti s npr. *tagiranjem* slika na društvenim mrežama. Čini ga 95% operacija čitanja te 5% operacija ažuriranja.

Iz grafova sa slike 4.2. vidljiv je uzorak sličan scenariju opterećenja „A“ sa slike 4.1., gdje *Cubieboard2* sustav ima probleme s povećanjem broja niti. Minimalna latencija je očekivano manja kod *HP650* dok je maksimalna latencija približno jednaka na oba sustava što je poprilično neočekivano. Međutim, u oba scenarija opterećenja maksimalna latencija na minimalnom broju niti kod mjerenja na *HP650* postiže iznenađujuće visoke vrijednosti.

4.1.3. Scenarij opterećenja „C“

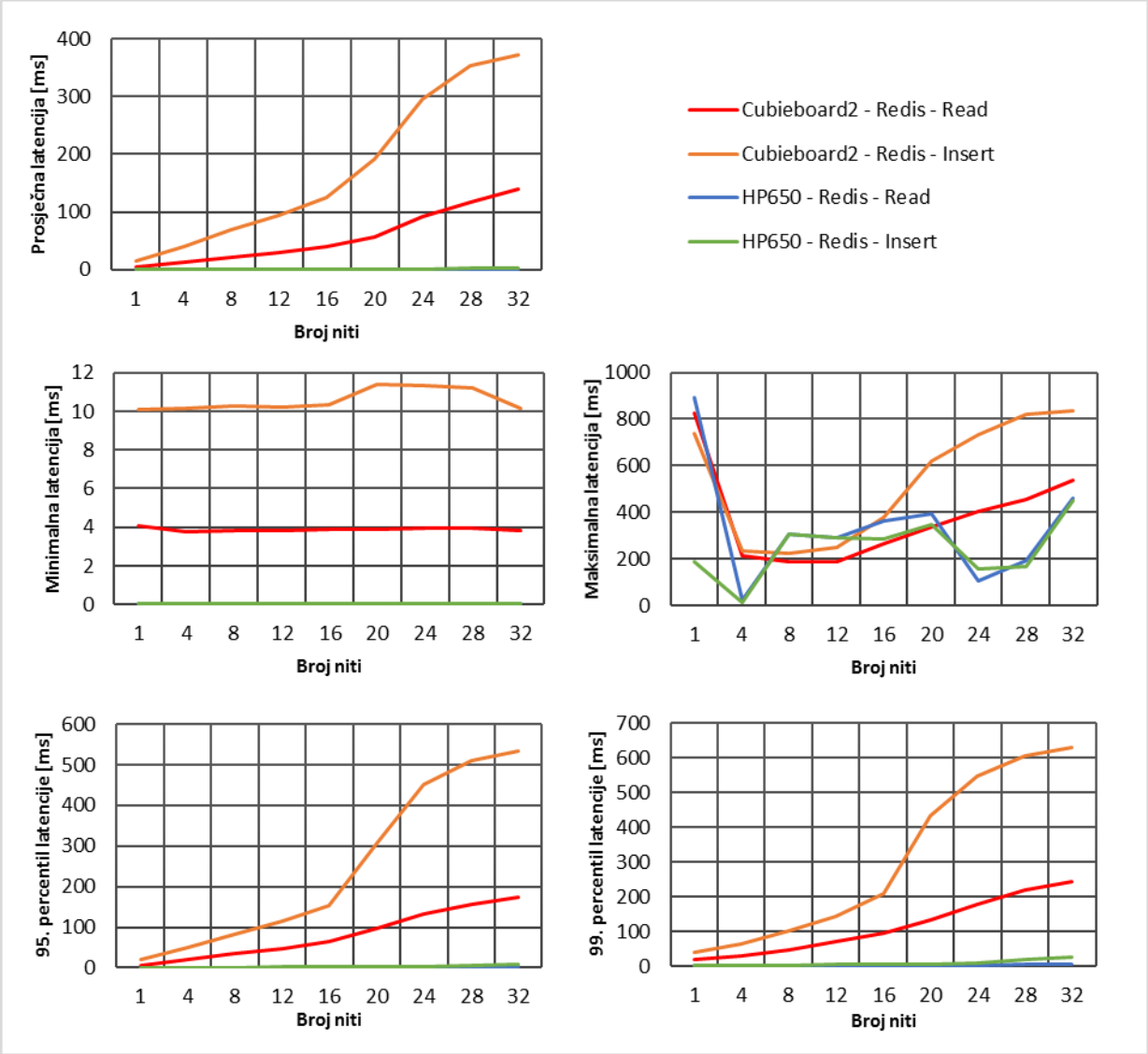


Sl. 4.3. Usporedba rezultata mjerenja „C“ scenarija opterećenja

Scenarij opterećenja „C“ može se poistovjetiti s tzv. c*aching*-om, odnosno učitavanjem podataka u radnu memoriju poslužitelja radi brže obrade istih. Čini ga 100% operacija čitanja.

Iz rezultata mjerenja prikazanih na slici 4.3. vidljiv je, kao i u dva prethodna scenarija, isti uzorak. Povećanjem broja niti kod *Cubieboarda2* dolazi do velikih problema, odnosno primjetan je lagani eksponencijalni skok. Minimalna latencija mjerenja na *HP650* je očekivano manja, te se opet pojavljuje neočekivano visoka maksimalna latencija korištenjem jedne niti u mjerenjima na *HP650*.

4.1.4. Scenarij opterećenja „D“



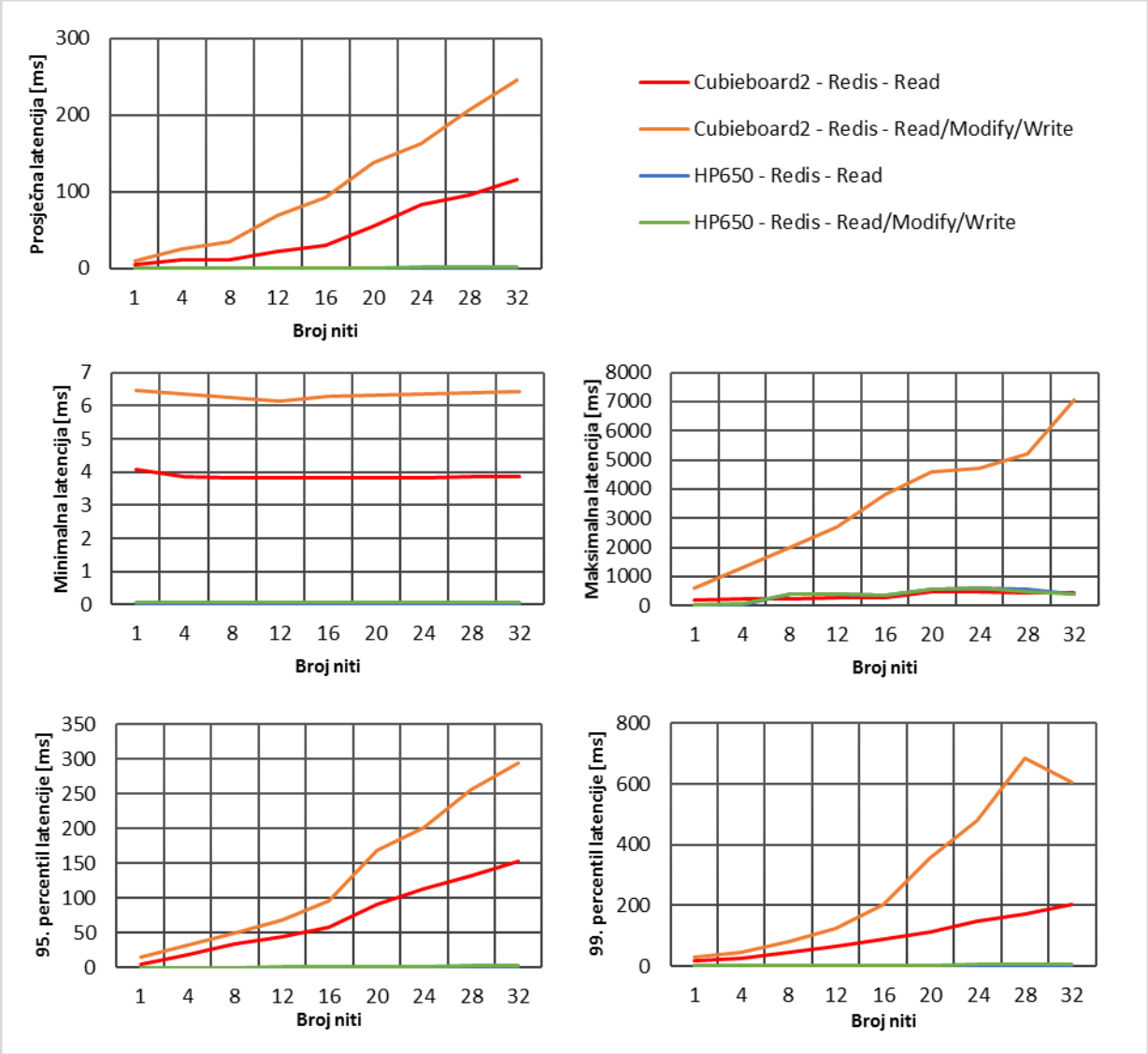
Sl. 4.4. Usporedba rezultata mjerenja „D“ scenarija opterećenja

Scenarij opterećenja „D“ može se poistovjetiti s ažuriranjem statusa korisnika ili osvježavanjem *web* stranice kako bi se došlo do najnovijeg sadržaja. Čine ga 95% operacija čitanja te 5% operacija pisanja (eng. *Insert*).

Iz rezultata sa slike 4.4. vidljivo je da operacije pisanja kod *Cubieboard2* sustava uzrokuju duplo veću latenciju nego operacije čitanja, a na nekim dijelovima i više od toga dok su u prva dva scenarija operacije ažuriranja bile relativno slične operacijama čitanja, osim u slučajevima minimalnih latencija.

Nastavlja se uzorak poprilično jednakih maksimalnih latencija na oba sustava, dok su se u ovom mjerenju neočekivanoj visokoj vrijednosti korištenjem jedne niti pridružile operacije čitanja i pisanja na *Cubieboard2* sustavu.

4.1.5. Scenarij opterećenja „F“

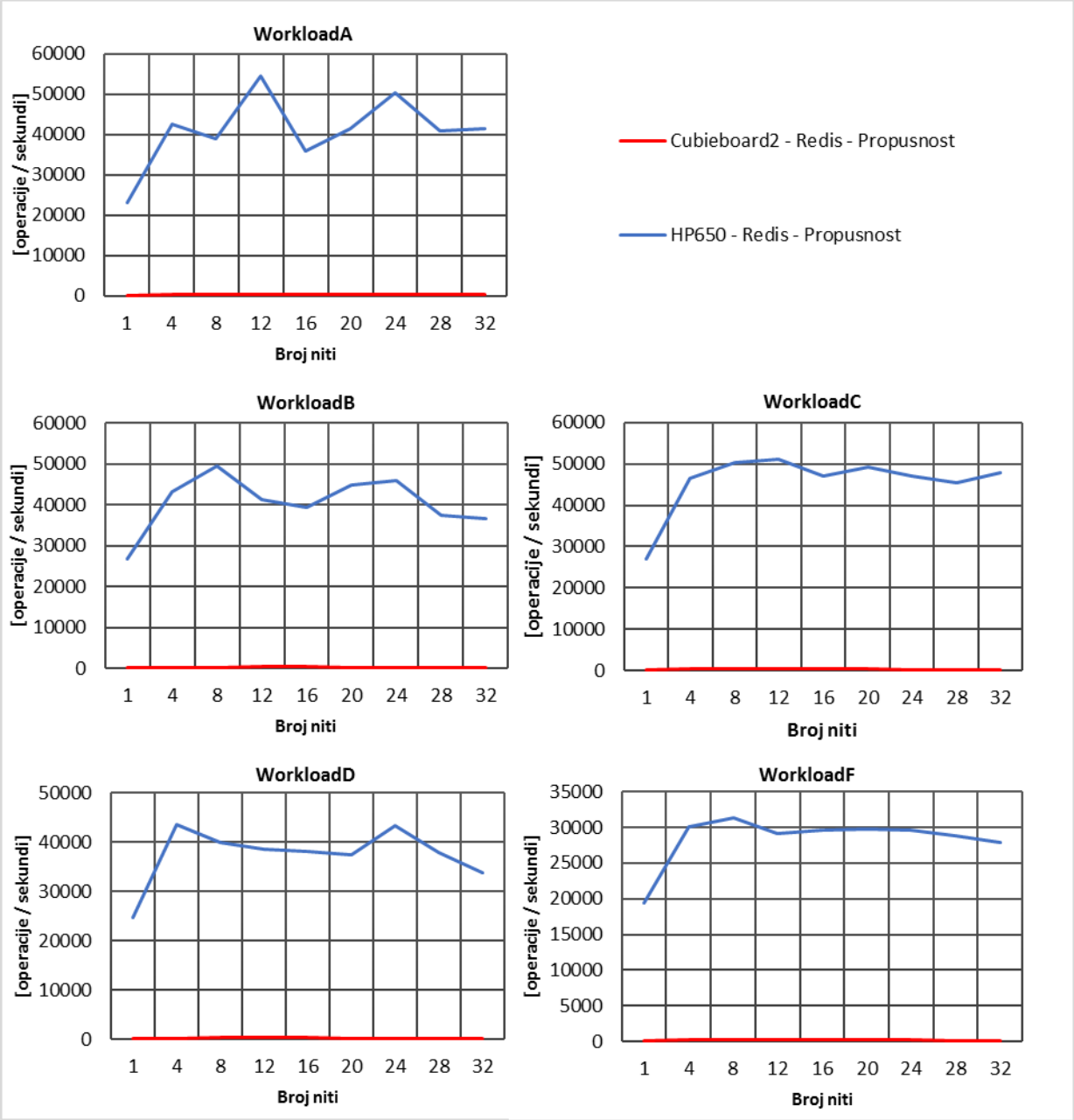


Sl. 4.5. Usporedba rezultata mjerenja „F“ scenarija opterećenja

Scenarij opterećenja „F“ može se poistovjetiti s normalnim radom baze podataka gdje se podaci o korisniku redovito čitaju a neki i modificiraju nakon čitanja te se ponovno spremaju u bazu. Čine ga 50% operacija čitanja te 50% operacija čitanja/modificiranja/pisanja (eng. *Read/Modify/Write*).

Iz rezultata mjerenja prikazanih na slici 4.5. vidljiv je uzorak koji se ponavlja kroz sve scenarije opterećenja. Operacije čitanja/modificiranja/pisanja daju slične rezultate kao u prethodnom scenariju operacija pisanja. Međutim, ovo je prvi slučaj da maksimalne latencije mjerenja na *HP650* korištenjem jedne niti ne daju neočekivano visoke rezultate, ali zato operacije čitanja/modificiranja/pisanja porastom niti eksponencijalno rastu do izuzetno velikih vrijednosti.

4.1.6. Propusnost svih scenarija opterećenja



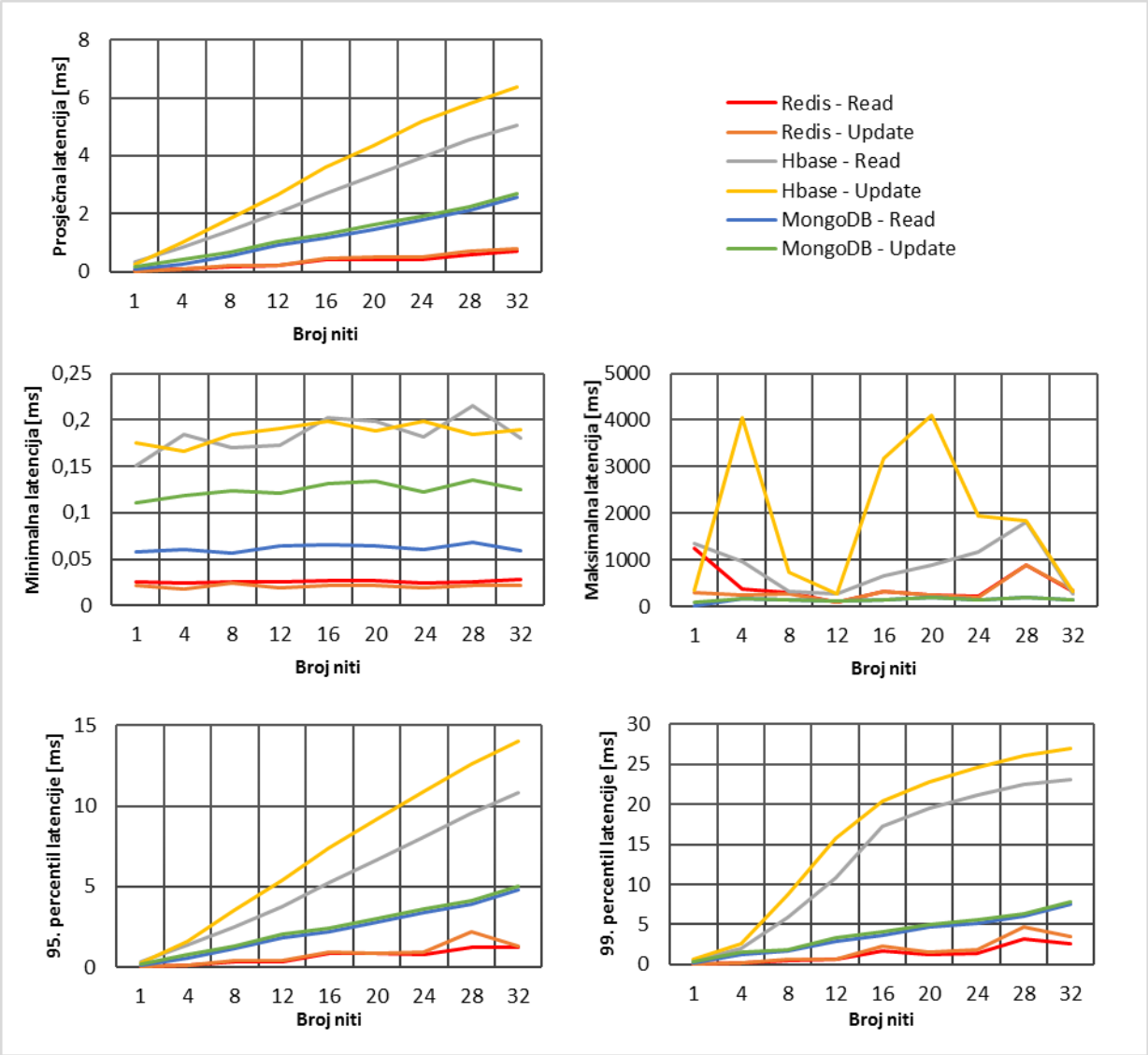
Sl. 4.6. Usporedba propusnosti mjerenja svih scenarija opterećenja

Kao što je vidljivo iz rezultata prikazanih na slici 4.6. propusnost ostvarena mjerenjem na *HP650* dosegla je iznimno visoke vrijednosti, točnije, srednja vrijednost propusnosti iznosi 38633,21 operacije po sekundi, više nego što je to bilo očekivano. *Cubieboard2* u istim scenarijima, postiže srednju vrijednost propusnosti u iznosu od 266,42 operacije po sekundi, što je 145 puta manje. Također, iz grafova je vidljivo da se najmanje propusnosti ostvaruju korištenjem samo jedne niti, što je očekivano, te da će prevelikim povećanjem broja niti propusnosti početi ponirati. Jedina iznenađenja su lagani padovi propusnosti na sredini testiranja, odnosno korištenjem oko šesnaest niti.

4.2. Usporedba rezultata mjerenja *Redis*, *Hbase* i *MongoDB* baza podataka

Svi parametri korišteni u mjerenjima su identični u obje kategorije usporedbe, kao i scenariji opterećenja koji su također ranije navedeni i objašnjeni. Sva mjerenja iz ove kategorije usporedbe obavljena su na *HP650*.

4.2.1. Scenarij opterećenja „A“

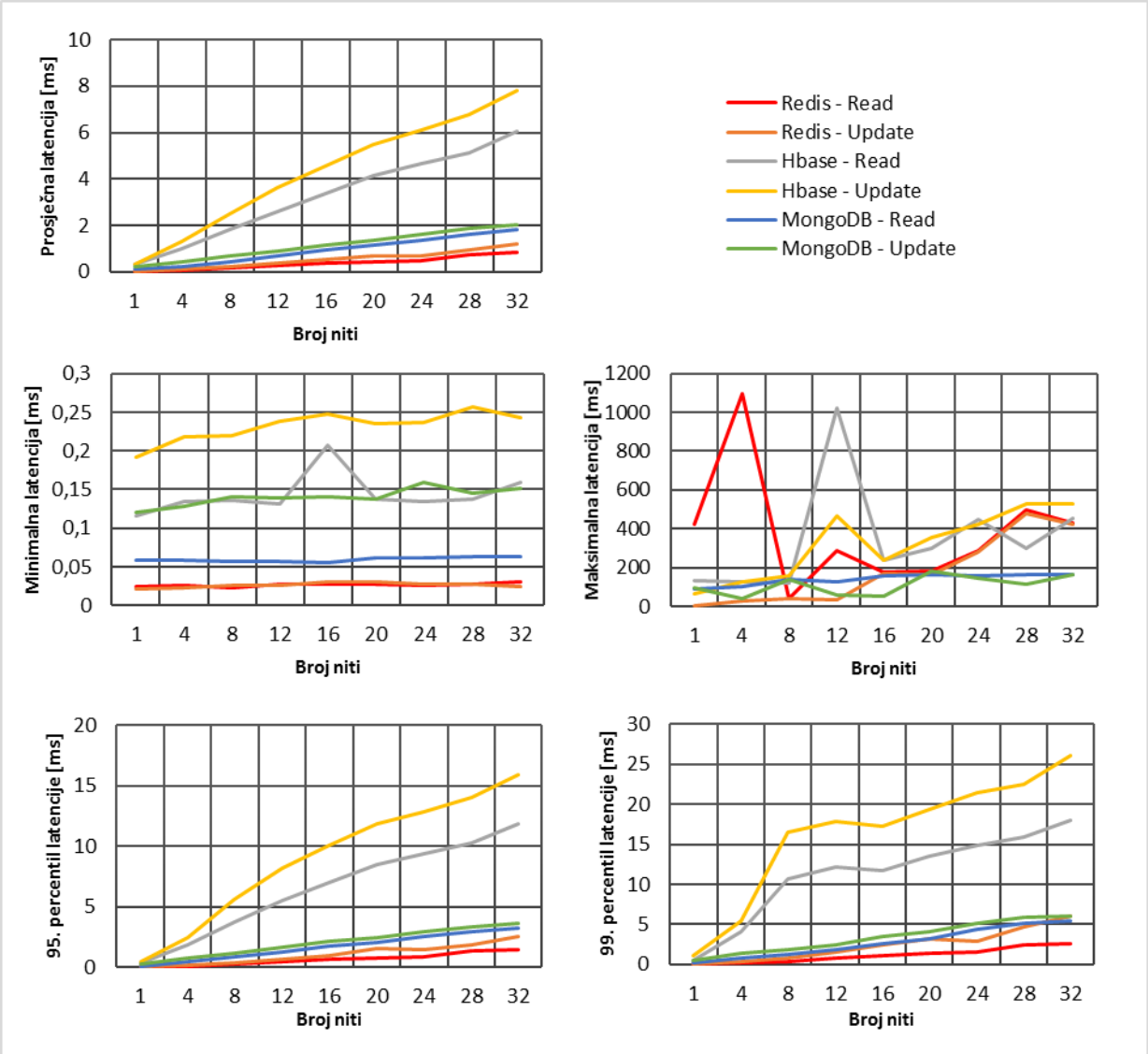


Sl. 4.7. Usporedba rezultata mjerenja „A“ scenarija opterećenja

Kao što je očekivano, latencije sve tri baze podataka izuzetno su male što je vidljivo iz rezultata mjerenja prikazanih na slici 4.7.. *Redis* baza podataka pokazuje najbolje rezultate mjerenja s izuzetno niskim latencijama, dok *Hbase* pokazuje najgore, što je bilo i očekivano s obzirom na specifičnosti namjena navedenih baza. Također, iz rezultata je vidljivo da operacije ažuriranja rezultiraju većim latencijama kod *Hbase* i *MongoDB* baza podataka u usporedbi s

operacijom čitanja, te je vidljiv porast latencije sukladno s povećanjem broja niti koji je posebno izražen kod *Hbase* baze podataka.

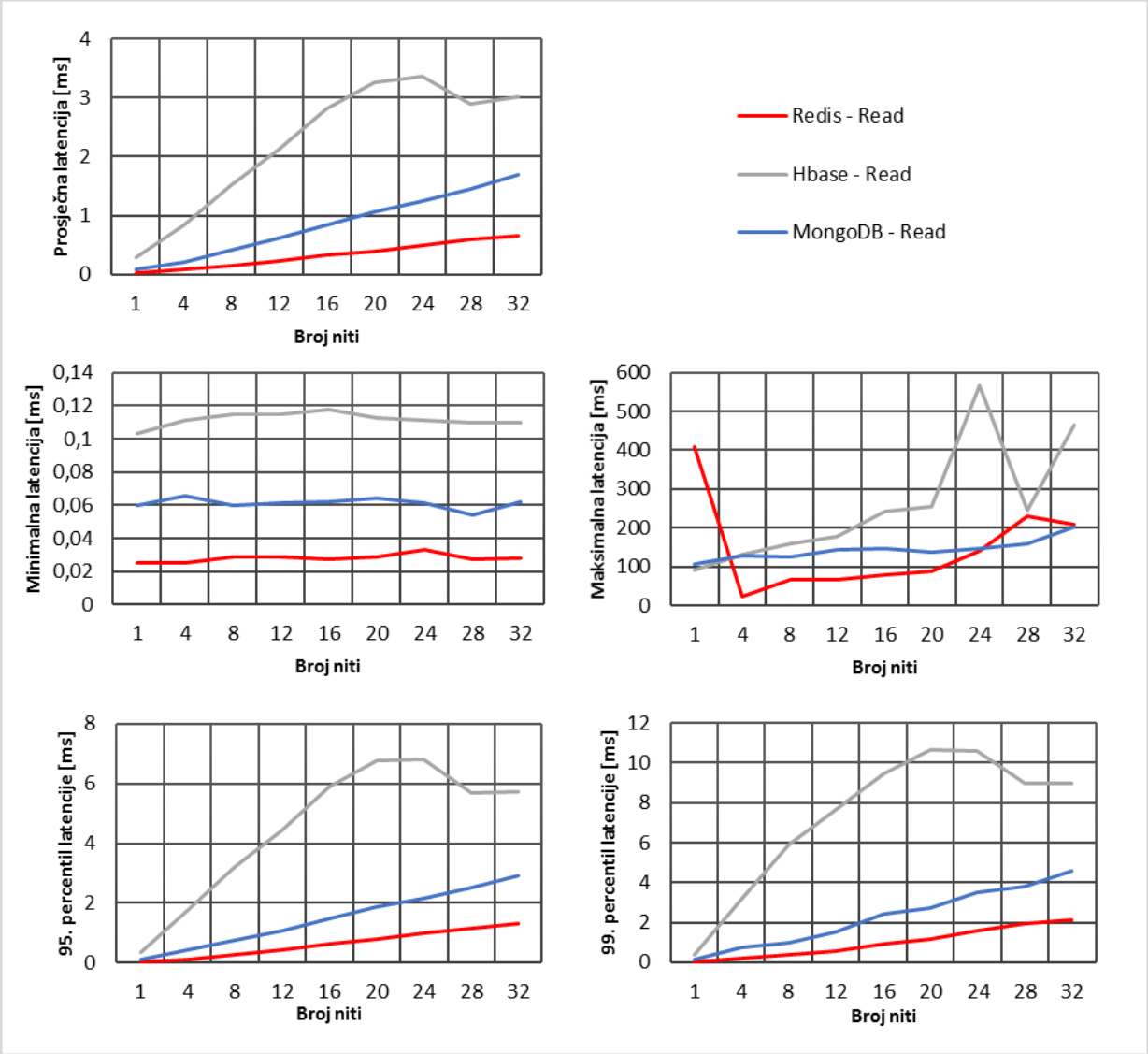
4.2.2. Scenarij opterećenja „B“



Sl. 4.8. Usporedba rezultata mjerenja „B“ scenarija opterećenja

Iz rezultata mjerenja prikazanih na slici 4.8. vidljivi su generalno isti rezultate kao i u prethodnom scenariju s iznimkom minimalne latencije čitanja *Hbase* baze podataka koja se približila minimalnoj latenciji ažuriranja *MongoDB* baze podataka. Također, kao i u prošloj kategoriji usporedbi, vidljiv je skok maksimalne latencije *Redis* baze podataka sukladno s korištenjem minimalne količine niti. Promatrajući oba scenarija opterećenja vidljivo je da je maksimalna latencija najkonzistentnija u *MongoDB* bazi podataka, dok maksimalna latencija u *Hbase* bazi podataka ima najviše oscilacija.

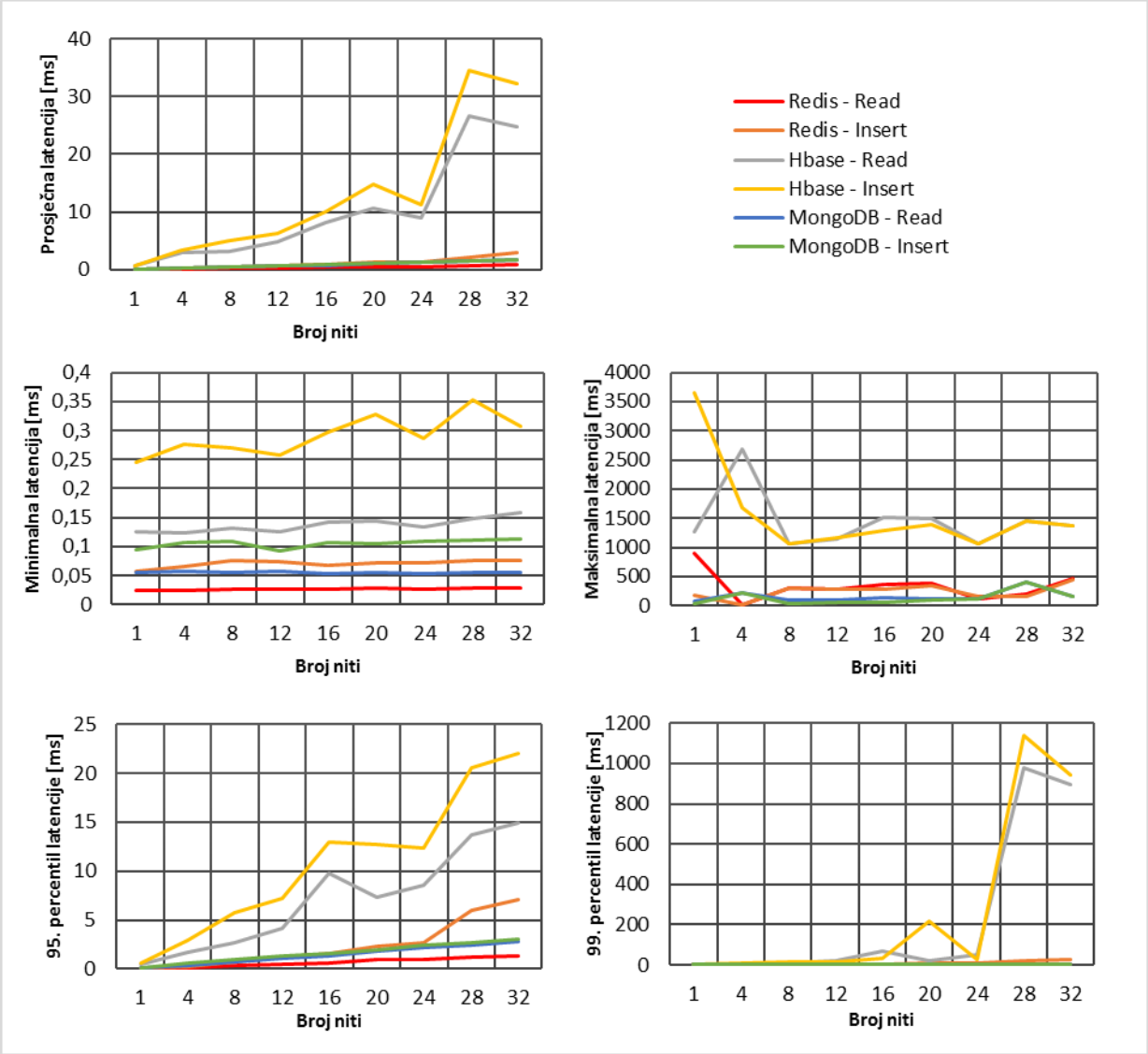
4.2.3. Scenarij opterećenja „C“



Sl. 4.9. Usporedba rezultata mjerenja „C“ scenarija opterećenja

Iz rezultata mjerenja prikazanih na slici 4.9. te prethodnih scenarija, vidljivo je da korištenjem jedne niti sve tri baze ostvaruju podjednaku latenciju, dok povećanjem broja istih *Hbase* doživljava najveće skokove latencije. Jedino je iznenađenje kod rezultata mjerenja „C“ scenarija opterećenja prestanak izuzetnog rasta latencije dolaskom do maksimalnih brojeva korištenih niti kod *Hbase* baze podataka. Također, opet se pojavljuje visoka vrijednost maksimalne latencije kod korištenja jedne niti u *Redis* bazi podataka.

4.2.4. Scenarij opterećenja „D“

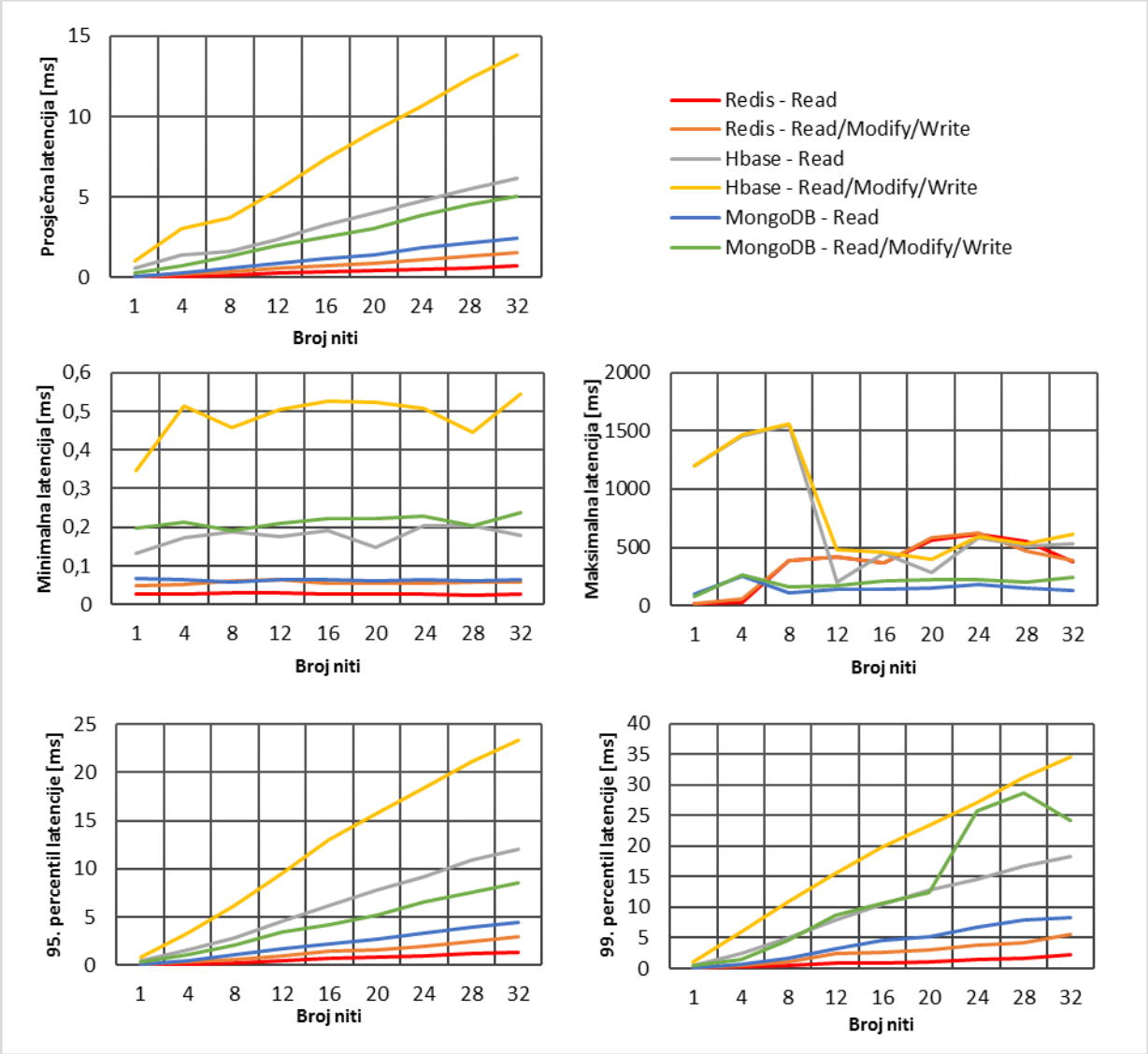


Sl. 4.10. Usporedba rezultata mjerenja „D“ scenarija opterećenja

Po prvi puta *Redis* baza podataka gubi dominaciju od strane *MongoDB* baze podataka i to u slučaju operacije pisanja što je vidljivo iz rezultata mjerenja prikazanih na slici 4.10. Točnije, latencije operacije pisanja u *Redis* bazi podataka su u gotovo svim grafovima veće od operacija pisanja u *MongoDB* bazi.

Latencije *Hbase* baze podataka pokazuju jako velike skokove korištenjem maksimalnog broja niti a posebice operacija pisanja. Iznenađenje kod rezultata mjerenja „D“ scenarija opterećenja je visoka vrijednost maksimalne latencije kod operacija *Hbase* baze podataka pri korištenju minimalnog broja niti, dok se ta vrijednost kasnije stabilizira.

4.2.5. Scenarij opterećenja „F“

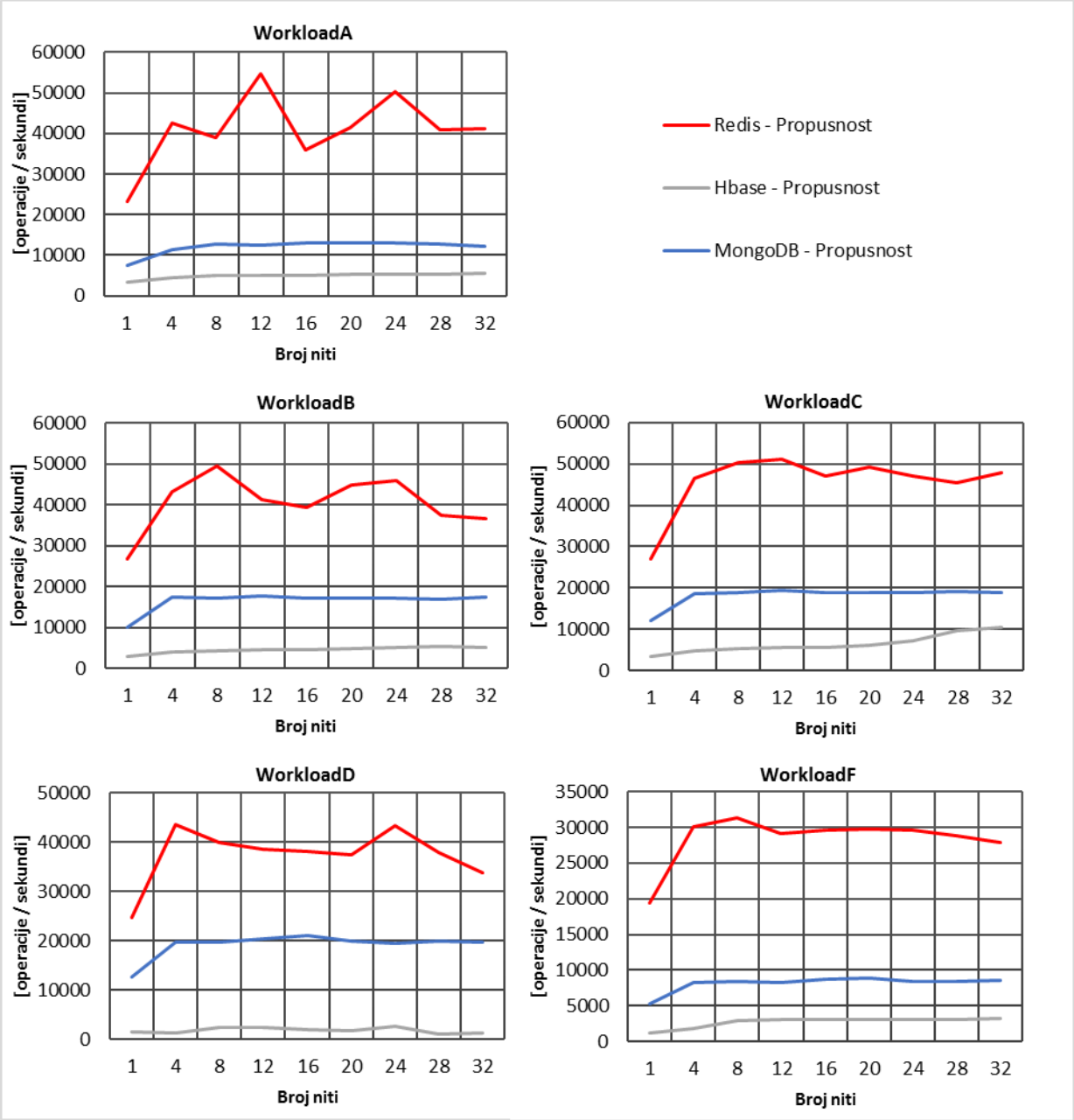


Sl. 4.11. Usporedba rezultata mjerenja „F“ scenarija opterećenja

Iz rezultata mjerenja „F“ scenarija opterećenja prikazanih na slici 4.11. lako je uočljiv skok latencije operacije čitanja/modificiranja/pisanja u svim mjerenim bazama podataka, no posebice u *MongoDB* bazi podataka. Taj iznos se u nekim slučajevima čak približava operaciji čitanja *Hbase* baze podataka.

Vidljiva je i ponovna dominacija *Redis* baze podataka te konzistentnost maksimalne latencije kod *MongoDB* baze podataka. Također, javlja se veliki skok vrijednosti maksimalne latencije kod korištenja minimalnih količina niti u *Hbase* bazi podataka.

4.2.6. Propusnost svih scenarija opterećenja



Sl. 4.12. Usporedba propusnosti mjerenja svih scenarija opterećenja

Iz rezultata mjerenja propusnosti prikazanih na slici 4.12. vidljiva je nedvojbena i očekivana nadmoć *Redis* baze podataka. Također, vidljivo je da na propusnost *Redis* baze podataka najviše utječu povećanja broja niti, dok su *Hbase* i *MongoDB* baze podataka relativno konzistentne. Kao što smo vidjeli i iz mjerenja propusnosti u prethodnoj kategoriji, iznos prosječne propusnosti u *Redis* bazi podataka je 38633,21 operacija po sekundi. *MongoDB* postiže prosječnu propusnost u iznosu od 14797,43 operacija po sekundi, dok *Hbase* postiže 4028,48 operacija po sekundi.

5. ZAKLJUČAK

U nekoliko posljednjih godina, razvijao se velik broj sustava za upravljanje bazama podataka od kojih su mnogi postali veoma korisni jer su, za razliku od relacijskih baza podataka, dizajnirani na način da zadovoljavaju zahtjeve koje postavlja veoma nagli i brzi porast količine različitih podataka. Ne-relacijske baze podataka sve se više koriste u poslovnom svijetu ali i među znanstvenom zajednicom. Temeljna se struktura ne-relacijskih baza podataka u mnogim aspektima razlikuje od strukture relacijskih baza podataka a na pitanje koja će tehnologija biti vodeća u budućnosti još ne postoji jasno definiran odgovor.

Svrha ovog rada bila je ispitati kvalitetu usluge u ne-relacijskim bazama podataka na računalnim sustavima ograničenih resursa. *Cubieboard2* te *HP650* ranije navedenih specifikacija korišteni su kao računalni sustavi ograničeni resursa, a *Redis*, *Hbase* i *MongoDB* kao ne-relacijske baze podataka.

Iz rezultata mjerenja potvrđena je pretpostavka da će se kao najbolje rješenje pokazati *Redis* baza podataka. To je jedina baza podataka za koju je bilo moguće uspješno provesti mjerenje na *Cubieboard2* računalnom sustavu iz nekoliko razloga. Svrha je *Redisa* da se implementira u sustavima gdje ne postoje zahtjevi za visokim stupnjem sigurnosti te za kompliciranim modelom podataka ili gdje su minimalna latencija i visoka propusnost od prevelike važnosti.

Iz rezultata dobivenih mjerenjem na *HP650* moguće je zaključiti da *Hbase* nije validan odabir kao ne-relacijska baza podataka na *Cubieboard2* računalnom sustavu zbog čega mjerenje u konačnici nije niti provedeno. Ako se kao prosječna propusnost *Redisa* uzme 38633,21 operacija po sekundi i ako se uzme *Hbaseova* prosječna propusnost koja iznosi 4082,48 operacija u sekundi moguće je zaključiti da *Redis* ima 9,46 puta veću propusnost. Dakle, ako je *Cubieboard2* na *Redisu* postigao prosječnu propusnost od 266,42 operacije po sekundi, tada bi s *Hbaseom* postigao prosječnu propusnost od 28,15 operacija po sekundi što bi značilo da bi prilikom testiranja na uzorcima od dva milijuna operacija prosječno trajanje testa iznosilo 19,73 sati. Kao što je u radu već navedeno, pri opterećenju dužem od dvanaest do petnaest sati *Cubieboard2* je prestajao s radom što znači da teoretski ne bi bilo moguće izvršiti mjerenja s odabranim parametrima. Ovi rezultati nisu iznenađujući ako se u obzir uzme da *Hbase* baza podataka nije namijenjena za rad u ovako ograničenom okruženju te s ovako malo podataka. Njena prednost bila bi vidljiva kada bi radili s milijardama podataka i u tome slučaju bi vjerojatno bile postignute iste brzine.

Rezultati dobiveni mjerenjem na *MongoDB* bazi podataka potvrdili su njenu kvalitetu, pouzdanost te zavidnu brzinu. *MongoDB* baza podataka odabrana je jer je daleko

najpopularnija ne-relacijska baza podataka zbog sličnosti s relacijskim bazama podataka i pouzdanosti (*ACID* svojstva). Mjerenja *MongoDB* baze podataka na *Cubieboard* računalnom sustavu nisu provedena zbog nemogućnosti instalacije uzrokovane nedostatkom podrške za ARM arhitekturu. Pretpostavka je da je *MongoDB* prekompleksna baza podataka za ograničeni sustav poput *Cubieboarda2* koji je namijenjen samo za izvršavanje *lightweight* zadataka.

Cubieboard2 kao izbor za poslužitelja dokazao je da nije prihvatljivo, praktično i pouzdano rješenje.

Ne-relacijske baze podataka predstavljaju veoma kompleksne entitete i svaka posebno mogla bi biti tema nekog daljnjeg dubinskog istraživanja. Jedan od aspekata mogućeg testiranja bila bi vertikalna i horizontalna skalabilnost odnosno klasteriranje. Također, u današnje vrijeme postoji enorman izbor računalnih sustava ograničenih resursa koje tek treba testirati kako bi se istražile njihove mogućnosti i potencijalne namjene.

LITERATURA

[1] NoSQL, Napredni modeli i baze podataka, Fakultet elektrotehnike i računarstva, Zagreb, studeni 2013., [Prezentacija s predavanja], (https://www.fer.unizg.hr/predmet/nmbp), pristup ostvaren 20.4.2016.

[2] A. K. Zaki, NoSQL Databases: New Millennium Database for Big Data, Big Users, Cloud Computing and its Security Challenges, IJRET: International Journal of Research in Engineering and Technology, 3, 3, str. 403-409., svibanj, 2014., (http://esatjournals.net/ijret/2014v03/i15/IJRET20140315080.pdf), pristup ostvaren 20.4.2016.

[3] D. T. Lee, Database-oriented decision support systems, (http://www.computer.org/csdl/proceedings/afips/1983/5090/00/50900453.pdf), pristup ostvaren 25.04.2016

[4] Five big data challenges and how to overcome them with visual analytics, (http://libapps.libraries.uc.edu/blogs/dlc/wp-content/uploads/sites/3/2013/03/2_25952_FiveBigDataChallenges.pdf), pristup ostvaren 11.05.2016.

[5] DB-Engines, (http://db-engines.com/en/ranking), pristup ostvaren 11.05.2016.

[6] O. Orel, Otkrivanje znanja iz relacijskih podataka uporabom graf baza podataka, (https://www.fer.unizg.hr/_download/repository/KDI,_Ognjen_Orel.pdf), pristup ostvaren 11.5.2016.

[7] C. Strauch, NoSQL Databases, (http://www.christof-strauch.de/nosqldbs.pdf), pristup ostvaren 25.04.2016.

[8] S. Weber, NoSQL Databases, (http://wiki.hsr.ch/Datenbanken/files/Weber_NoSQL_Paper.pdf) pristup ostvaren 25.04.2016.

[9] B. Medjahed, M. Ouzzani, A. K. Elmagarmid, Generalization of ACID Properties, (https://www.cs.purdue.edu/homes/ake/pub/GeneralizationOfACIDProperties.pdf), pristup ostvaren 25.04.2016.

[10] Top 5 Considerations When Evaluating NoSQL Databases, (http://www.ascent.co.za/documents/mongodb/10gen_Top_5_NoSQL_Considerations.pdf), pristup ostvaren 11.05.2016.

[11] Redis, (http://redis.io/), pristup ostvaren 02.06.2016.

[12] Redis Sponsors, (http://redis.io/topics/sponsors), pristup ostvaren 02.06.2016.

[13] Redis, An introduction to Redis data types and abstractions, (http://redis.io/topics/data-types-intro), pristup ostvaren 02.06.2016.

[14] Redis, Data Type, (http://redis.io/topics/data-types), pristup ostvaren 02.06.2016.

[15] Redis,Transaction, (http://redis.io/topics/transactions), pristup ostvaren 02.06.2016.

[16] Redis, Redis Persistence, (http://redis.io/topics/persistence), pristup ostvaren 02.06.2016.

[17] Redis, Redis Protocol specification, (http://redis.io/topics/protocol), pristup ostvaren 02.06.2016.

[18] K. Seguin, The Little MongoDB Book, (http://openmymind.net/mongodb.pdf), pristup ostvaren 13.06.2016.

[19] Apache Hbase, (https://Hbase.apache.org/), pristup ostvaren 11.05.2016.

[20] F. Chang...[et al.], Bigtable: A Distributed Storage System for Structured Data, (http://static.googleusercontent.com/media/research.google.com/en//archive/bigtable-osdi06.pdf), pristup ostvaren 13.06.2016.

[21] Apache Hbase, Hbase History, (http://Hbase.apache.org/0.94/book/Hbase.history.html), pristup ostvaren 11.05.2016.

[22] A. Khurana, Introduction to Hbase Schema Design, (http://0b4af6cdc2f0c5998459-c0245c5c937c5dedcca3f1764ecc9b2f.r43.cf2.rackcdn.com/9353-login1210_khurana.pdf), pristup ostvaren 11.05.2016.

[23] J. Anderson, How to use Hbase Thrift Interface, (http://blog.cloudera.com/blog/2013/09/how-to-use-the-Hbase-thrift-interface-part-1/), pristup ostvaren 11.05.2016.

[24] Planet Cassandra, (http://www.planetcassandra.org/what-is-apache-cassandra/), pristup ostvaren 11.05.2016.

[25] O. Janković, NoSQL graf baza podataka: od domena do modela preko upita,Infoteh-Jahorina, 14, str. 567-571., ožujak, 2015., (http://infoteh.etf.unssa.rs.ba/zbornik/2015/radovi/RSS-3/RSS-3-2.pdf), pristup ostvaren 11.05.2016.

[26] Neo4J, Neo4J Highlights, (http://neo4j.com/docs/2.2.0-M02/introduction-highlights.html), pristup ostvaren 11.05.2016.

[27] B. F. Cooper...[et al.], Benchmarking Cloud Serving Systems with YCSB, (https://www.cs.duke.edu/courses/fall13/cps296.4/838-CloudPapers/ycsb.pdf), pristup ostvaren 10.06.2016.

[28]Research Yahoo! Yahoo Cloud Serving Benchmark, (https://research.yahoo.com/news/yahoo-cloud-serving-benchmark/), pristup ostvaren 10.06.2016.

[29] B. F. Cooper, YCSB, (https://github.com/brianfrankcooper/YCSB/wiki/Running-a-Workload), pristup ostvaren 10.06.2016.

[30] ARM Architecture Reference Manual, (https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/third-party/ddi0100e_arm_arm.pdf), pristup ostvaren 13.06.2016.

[31] VoltDB, Documentation, (https://docs.voltdb.com/PlanningGuide/ChapBenchmark.php), pristup ostvaren 02.06.2016.

[32] A. Vogel, B. Kerherve, G. v. Bochmann, and J. Gecsei, Distributed multimedia applications and quality of service: a survey, (http://www.site.uottawa.ca/~bochmann/Curriculum/Pub/1995%20-%20Distributed%20multimedia%20applications%20and%20quality%20of%20service%20-%20A%20survey.pdf), pristup ostvaren 17.08.2016.

[33] Cubietech Limited, Cubieboard 1, (http://www.cubietech.com/index.php?route=product/product&path=62&product_id=111), pristup ostvaren 13.06.2016.

[34] Cubietech Limited, Cubieboard 2, (http://www.cubietech.com/index.php?route=product/product&path=62&product_id=112), pristup ostvaren 13.06.2016.

[35] K. Kovacs, Cassandra vs MongoDB vs CouchDB vs Redis vs Riak vs Hbase vs CoucHbase vs OrientDB vs Aerospike vs Neo4j vs Hypertable vs ElasticSearch vs Accumulo vs VoltDB vs Scalaris Comparison, (http://kkovacs.eu/cassandra-vs-mongodb-vs-couchdb-vs-redis), pristup ostvaren 02.06.2016.

[36] NoSQL database, (http://nosql-database.org/), pristup ostvaren 02.06.2016.

SAŽETAK

NoSQL baze podataka nastale su kao alternativa najčešće korištenim relacijskim bazama podataka. Nastale su iz potrebe za rješavanjem problema vezanih uz skalabilnost, uz upravljanje podacima u realnom vremenu, uz upravljanje polustrukturiranim i nestrukturiranim podacima i dr.

Cilj ovog diplomskog rada je ispitivanje kvaliteta usluge u ne-relacijskim distribuiranim bazama podataka na računalnim sustavima ograničenih resursa. U teorijskom dijelu rada, definirani su osnovni pojmovi vezani uz ne-relacijske baze podataka te je opisana klasifikacija ne-relacijskih baza podataka. Uz to, navedeni su i opisani konkretni primjeri baza podataka svake vrste. Nadalje, predstavljene su i glavne karakteristike *Yahoo! Cloud Service Benchmark*-a (*YCSB*) i ARM arhitekture. U istraživačkom dijelu rada provedena je usporedba kvalitete usluge *Redis* baze podataka na *Cubieboardu2* i prijenosnom osobnom računalu *HP650*. Također, provedena je i usporedba kvalitete usluge u različitim bazama podataka (*Redis*, *Hbase*, *MongoDB*). Kvaliteta usluge u različitim bazama mjerena je na laptopu.

Ključne riječi : *ne-relacijske baze podataka, NoSQL, kvaliteta usluge, ARM arhitektura, YCSB, Cubieboard, Redis, Hbase, MongoDB*

ABSTRACT

NoSQL databases are an emerging alternative to the most commonly used relational databases. Non-relational databases were designed to overcome problems such as scalability, handling real time data and handling unstructured or semi structured data etc.

Main goal of this paper was to examine the quality of service in non-relational (NoSQL) distributed databases on the computer systems of limited resources. In the theoretical part of the thesis, basic terms related to the non-relational databases were described along with classification of non-relational databases. In addition, specific examples of each type of database were listed and described. Furthermore, the main features of the *Yahoo! Cloud Service Benchmark's* (*YCSB*) and ARM architecture were presented. In the research part of the work Quality of Service comparison was conducted between *Redis* database on *Cubieboard2* and Notebook *HP650*. Also, quality of service comparison was conducted on different databases (*Redis*, *Hbase*, *MongoDB*). Quality of service, on different databases, was measured on laptop.

Keywords: non-relational databases, NoSQL, quality of service, ARM arhitecture, *YCSB*, *Cubieboard, Redis, Hbase*, *MongoDB*

ŽIVOTOPIS

Ivan Kordaso rođen je u Požegi 13. veljače 1990. godine. Pohađao je osnovnu školu u mjestu Brestovac i srednju tehničku školu u Požegi. Elektrotehnički fakultet, smjer sveučilišni preddiplomski studij računarstva, upisuje 2008. godine kao redovni student te isti završava 2012. godine. Godine 2012. upisuje diplomski studij procesnog računarstva. Krajem 2015. godine počinje raditi kao mlađi Java programer u tvrtki PPD - Informacijski Sustavi.

Ivan Kordaso

_____

PRILOZI

P.4.1. Vrijednosti rezultata mjerenja

WORKLOADA											
Broj niti	Propusnost [ops/s]	Prosječna latencija [µs]		Minimalna latencija [µs]		Maksimalna latencija [µs]		95. percentil latencije [µs]		99. percentil latencije [µs]	
		Read	Update	Read	Update	Read	Update	Read	Update	Read	Update
Cubieboard - Redis											
1	206,28	4932,35	2735,02	4088	1895	139135	141055	5107	2755	16183	14647
4	328,37	10941,99	9770,10	3852	1598	230271	226047	19247	17887	27215	25551
8	399,85	17956,00	16856,07	3826	1610	265727	269311	31199	29167	45023	39263
12	415,64	25570,56	25209,00	3836	1698	203263	189311	45919	44191	68607	66111
16	319,06	44484,12	45092,83	3852	1625	353279	273919	71615	70399	95551	104959
20	269,29	68688,59	67877,61	3880	1604	316415	350463	92031	90303	117311	127807
24	264,02	83741,48	82663,50	3904	1631	368127	385279	108863	107135	141183	151039
28	260,34	97356,73	96047,27	3886	1658	472831	463871	127615	125759	163455	174847
32	252,49	115407,50	114363,91	3970	1751	449791	515583	149375	147711	194687	210303
Laptop - Redis											
1	23158,06	48,41	29,64	25	22	1252557	302079	37	37	49	49
4	42517,01	88,11	92,37	24	18	372735	253951	142	148	205	252
8	38845,51	183,55	210,47	26	24	295423	283135	329	405	526	674
12	54595,58	199,73	226,35	25	19	92863	107455	387	439	619	722
16	35807,00	411,24	464,34	27	21	339455	327935	895	937	1637	2305
20	41384,73	438,15	492,93	27	22	251903	251391	869	908	1305	1526
24	50238,63	410,55	517,04	24	19	219647	204799	779	925	1432	1930
28	40964,30	596,79	732,24	26	22	890879	890367	1252	2203	3251	4703
32	41309,51	709,63	796,29	28	21	330239	350975	1214	1290	2581	3525
Laptop - HBase											
1	3237,09	355,82	252,36	151	176	1347583	350207	383	278	496	680
4	4358,51	832,28	985,67	184	166	959999	4063231	1365	1655	2053	2665
8	4906,41	1408,66	1829,61	170	184	329215	738303	2529	3515	5947	8687
12	5068,89	2036,80	2665,08	173	191	269823	282367	3789	5379	10863	15775
16	5038,33	2713,28	3603,92	203	198	654847	3190783	5243	7387	17247	20431
20	5177,48	3321,53	4352,79	199	188	904191	4095999	6691	9191	19551	22735
24	5236,15	3941,26	5175,85	182	199	1180671	1952767	8039	10895	21087	24623
28	5385,44	4534,36	5810,42	216	185	1828863	1831935	9567	12607	22431	26127
32	5563,19	5043,80	6391,79	181	190	274943	325631	10839	14023	23103	26927
Laptop - MongoDB											
1	7580,47	85,61	172,71	58	111	26239	85119	108	199	160	306
4	11465,39	274,78	412,74	60	118	170111	179455	599	799	1230	1582
8	12743,56	558,77	685,81	57	124	157823	158463	1159	1344	1642	1869
12	12379,53	898,41	1027,40	64	121	118911	129023	1859	2055	2957	3323
16	12965,88	1160,82	1293,05	66	132	149887	148095	2231	2431	3667	4081
20	12941,38	1472,13	1604,19	64	134	193663	193407	2841	3049	4639	5007
24	13017,53	1769,60	1899,37	60	122	160511	161279	3381	3591	5223	5587
28	12774,00	2115,78	2251,23	68	135	201087	200703	3919	4135	6119	6407
32	12063,67	2571,82	2711,24	59	125	155647	158719	4811	5035	7615	7871

WORKLOADB											
Broj niti	Propusnost [ops/s]	Prosječna latencija [µs]		Minimalna latencija [µs]		Maksimalna latencija [µs]		95. percentil latencije [µs]		99. percentil latencije [µs]	
		Read	Update	Read	Update	Read	Update	Read	Update	Read	Update
Cubieboard - Redis											
1	171,45	5036,79	2912,77	4046	1909	157311	104127	5091	2821	18079	15583
4	270,46	13014,25	12542,90	3806	1815	189695	130495	20255	20623	28015	26991
8	328,26	21587,50	21719,86	3818	1675	215935	134783	33599	32527	46559	43487
12	352,55	30167,71	29674,59	3850	1818	273151	291839	45759	45983	67519	64895
16	365,00	39031,50	42615,23	3814	1716	403199	397311	61791	63839	91199	92287
20	296,30	61596,36	65353,34	3842	1781	410111	331519	100927	99775	126975	143103
24	240,13	91564,27	93064,35	3852	1820	416767	441599	124799	123519	168575	194303
28	220,77	120033,18	120225,48	3862	1875	478719	457471	147071	145663	205695	233087
32	218,07	139864,63	140139,56	3866	1905	558079	546815	168191	166527	239871	262655
Laptop - Redis											
1	26664,18	35,36	34,19	25	22	423935	7779	44	44	53	74
4	43148,09	88,03	102,89	26	23	1093631	31807	129	156	197	402
8	49412,00	155,88	194,85	23	26	40671	39775	300	370	438	805
12	41330,85	279,01	352,71	27	26	290815	37983	468	726	743	1588
16	39322,86	390,67	507,07	28	30	175103	173823	685	935	1160	2501
20	44810,90	419,50	678,33	28	30	180607	167295	775	1539	1349	3271
24	46038,39	497,27	682,34	26	28	289535	282111	923	1444	1614	2925
28	37538,95	716,47	933,19	28	27	496639	478719	1378	1882	2383	4655
32	36527,01	829,22	1199,42	30	25	427007	425471	1444	2557	2653	6079
Laptop - HBase											
1	2999,63	328,50	338,41	116	192	136447	68031	373	435	463	1136
4	3958,70	986,81	1300,87	135	218	125439	128191	1843	2491	4115	5395
8	4286,36	1820,40	2508,45	136	220	120831	161279	3787	5667	10671	16447
12	4480,72	2614,42	3614,76	131	238	1021439	468479	5519	8155	12199	17903
16	4640,87	3367,97	4572,10	207	248	240511	239743	7015	10111	11791	17311
20	4732,09	4140,07	5482,79	137	235	303615	358399	8455	11895	13535	19327
24	5059,87	4647,80	6105,77	135	237	447487	422399	9423	12871	14871	21439
28	5326,26	5150,90	6764,73	138	257	302847	526847	10231	14039	15855	22495
32	5196,56	6035,99	7835,94	160	243	455423	527359	11879	15911	18079	26079
Laptop - MongoDB											
1	10049,65	91,64	197,90	59	121	88575	94975	129	332	197	485
4	17538,78	215,07	408,51	58	128	102975	39423	444	730	772	1421
8	17206,37	448,90	671,11	57	141	140799	139135	868	1184	1251	1813
12	17616,18	664,78	889,45	57	139	128191	59167	1290	1636	1842	2479
16	17085,55	918,87	1153,24	56	141	157695	52255	1754	2115	2619	3521
20	17210,22	1143,26	1380,51	61	137	164735	181631	2097	2447	3253	4103
24	17316,17	1365,93	1613,68	62	160	155903	148607	2545	2919	4379	5199
28	17027,51	1622,79	1863,45	63	146	164991	113791	3001	3377	5191	5847
32	17560,34	1801,01	2046,70	63	151	167679	163071	3223	3603	5379	6079

WORKLOADC						
Broj niti	Propusnost [ops/s]	Prosječna latencija [µs]	Minimalna latencija [µs]	Maksimalna latencija [µs]	95. percentil latencije [µs]	99. percentil latencije [µs]
		Read	Read	Read	Read	Read
Cubieboard - Redis						
1	205,98	4278,96	4062	106623	4335	6011
4	366,92	9833,53	3836	115711	13711	21759
8	363,28	19818,09	3860	126591	29519	40927
12	376,32	28654,56	3842	155391	43647	60287
16	378,49	37371,41	3834	226303	58431	85055
20	323,15	56819,43	3862	316927	92223	120447
24	262,93	85652,14	3832	368383	111551	150527
28	247,58	106389,88	3874	432639	130111	176895
32	239,80	128205,24	3870	425471	148863	205823
Laptop - Redis						
1	27108,72	34,89	25	410367	46	51
4	46535,44	83,52	25	23503	105	177
8	50322,06	154,92	29	68351	258	402
12	51013,90	229,50	29	68351	419	553
16	47120,91	331,20	27	80703	626	950
20	49200,49	395,69	29	88895	782	1160
24	46932,93	498,60	33	142207	981	1589
28	45535,27	594,31	27	231935	1158	1918
32	47818,29	650,18	28	208767	1317	2131
Laptop - HBase						
1	3436,04	287,77	103	91839	337	405
4	4662,22	849,43	111	131967	1738	3163
8	5239,72	1516,78	115	158719	3189	5927
12	5628,65	2118,04	115	178687	4439	7655
16	5673,90	2803,43	118	243583	5887	9463
20	6092,36	3264,49	113	255231	6787	10647
24	7098,89	3360,13	111	566271	6811	10575
28	9623,16	2882,76	110	246143	5703	9007
32	10532,19	3008,35	110	464639	5723	8999
Laptop - MongoDB						
1	11963,01	81,33	60	105727	108	116
4	18652,89	211,06	66	130175	415	742
8	18878,08	419,67	60	124607	767	1002
12	19321,80	616,17	61	145151	1093	1552
16	18857,61	842,38	62	148479	1460	2423
20	18849,61	1055,10	64	136831	1864	2723
24	18982,54	1255,31	61	145919	2139	3491
28	19245,02	1446,67	54	159487	2517	3827
32	18908,24	1683,12	62	203903	2905	4607

WORKLOADD											
Broj niti	Propusnost [ops/s]	Prosječna latencija [µs]		Minimalna latencija [µs]		Maksimalna latencija [µs]		95. percentil latencije [µs]		99. percentil latencije [µs]	
		Read	Insert	Read	Insert	Read	Insert	Read	Insert	Read	Insert
Cubieboard - Redis											
1	146,73	5244,09	14446,35	4078	10112	827903	735743	5219	21583	18879	40319
4	248,39	12980,46	40377,04	3798	10152	215423	236927	20399	51039	28639	63199
8	295,95	21883,21	68525,43	3830	10248	188287	226303	34495	83263	47839	102527
12	318,40	30465,66	93984,39	3834	10224	190079	250623	47071	115391	69439	143231
16	329,49	39896,62	126218,34	3862	10320	266239	380159	63359	154367	93631	209535
20	285,14	57218,06	191222,47	3922	11408	339199	620543	97215	305663	131711	433919
24	217,72	91519,08	295100,66	3982	11336	402431	732159	133119	452607	179327	546815
28	202,72	117579,28	353250,72	3950	11208	454911	821247	156799	510975	219263	606207
32	202,16	138915,48	372190,97	3844	10176	537599	835583	174335	533503	243583	631295
Laptop - Redis											
1	24746,66	35,64	85,51	25	58	889855	186879	43	108	52	139
4	43661,45	83,18	199,17	24	66	23583	15431	110	247	156	491
8	39884,34	180,08	485,14	27	75	307455	306431	300	775	497	2453
12	38646,60	281,44	717,73	27	73	290559	291071	490	1210	755	3705
16	38125,01	379,96	971,67	27	68	363519	286207	652	1639	1071	4907
20	37481,26	472,47	1233,85	28	71	394751	346367	916	2299	1387	6319
24	43330,37	492,46	1364,24	26	71	108927	159359	942	2665	1689	7631
28	37855,84	652,59	2071,68	29	75	196479	167679	1242	5979	3641	20383
32	33739,33	815,89	3013,63	28	77	460287	452607	1387	7079	3447	26863
Laptop - HBase											
1	1568,54	633,26	626,36	125	246	1263615	3645439	429	632	680	3069
4	1372,61	2878,52	3386,53	124	276	2689023	1680383	1686	2917	5203	6547
8	2399,07	3224,96	5119,04	132	270	1056767	1054719	2711	5695	5903	12607
12	2447,24	4785,92	6232,92	126	257	1155071	1156095	4175	7179	23167	14279
16	1936,90	8122,88	9992,00	142	297	1514495	1281023	9815	12959	70527	31471
20	1843,30	10601,52	14799,08	144	329	1485823	1399807	7395	12695	19023	219903
24	2648,59	8890,57	11295,88	133	287	1057791	1055743	8551	12383	50943	29071
28	1037,11	26523,07	34550,73	148	352	1463295	1462271	13647	20575	979455	1138687
32	1275,36	24658,44	32180,94	159	307	1367039	1369087	14887	22079	897535	945151
Laptop - MongoDB											
1	12723,78	73,46	134,49	56	94	74879	42911	85	172	114	277
4	19616,69	194,36	315,70	58	106	215039	215167	386	576	660	1065
8	19714,92	395,30	532,39	55	109	96319	42559	743	947	1004	1389
12	20351,47	578,40	712,76	57	93	93055	65375	1082	1279	1559	2113
16	21044,88	748,24	886,99	53	107	130047	66047	1360	1566	2143	2847
20	20038,67	985,96	1120,01	55	105	113791	104063	1787	1998	2711	3105
24	19596,12	1211,18	1355,48	54	109	123391	115839	2173	2401	3421	4123
28	19912,19	1390,48	1560,95	56	111	410879	410111	2475	2705	4411	5231
32	19659,11	1612,90	1757,09	55	113	151679	161663	2825	3057	4691	5215

WORKLOADF											
Broj niti	Propusnost [ops/s]	Prosječna latencija [µs]		Minimalna latencija [µs]		Maksimalna latencija [µs]		95. percentil latencije [µs]		99. percentil latencije [µs]	
		Read	Read/Modify/ Write	Read	Read/Modify/ Write	Read	Read/Modify/ Write	Read	Read/Modify/ Write	Read	Read/Modify/ Write
Cubieboard - Redis											
1	113,03	5085,82	9598,71	4092	6468	214271	590335	5159	14479	17855	28879
4	199,44	11188,76	24745,53	3874	6356	220799	1292287	19295	32479	27359	44031
8	230,68	11188,76	34892,35	3818	6243	227326	1994238	33431	50479	44869	79182
12	238,99	21779,02	69039,16	3839	6131	273854	2696190	44895	68479	65124	124334
16	231,24	29159,67	92708,48	3815	6287	290382	3798142	57897	96479	87688	202412
20	250,42	54239,35	137833,13	3819	6323	481022	4597439	90245	168686	114369	357687
24	199,82	83319,03	162957,78	3827	6359	472062	4711967	112895	201342	147557	479785
28	169,67	96398,71	207082,42	3846	6396	463103	5226495	132735	255999	172799	686079
32	166,04	116478,39	245207,08	3864	6432	454143	7041023	152575	294655	203391	606719
Laptop - Redis											
1	19454,69	33,97	63,64	26	50	18191	20399	44	83	50	96
4	30020,11	82,69	179,22	26	52	24015	54367	145	307	219	471
8	31420,36	157,13	346,62	29	61	385535	389119	251	544	420	1170
12	29178,34	253,17	558,03	29	63	415231	415231	477	1031	916	2499
16	29609,01	336,42	729,59	28	55	364031	364543	688	1419	929	2607
20	29770,32	419,36	901,57	27	55	564223	584703	791	1625	1173	3021
24	29590,18	503,65	1096,22	27	56	612863	625663	958	1979	1528	3909
28	28899,23	598,73	1313,30	25	58	549887	472831	1190	2421	1752	4279
32	27879,62	710,65	1552,43	28	58	382719	383487	1388	2921	2277	5495
Laptop - HBase											
1	1220,14	605,98	1024,91	133	348	1201151	1197055	468	808	547	1106
4	1809,87	1401,20	3022,08	174	515	1459199	1465343	1650	3285	2513	6055
8	2986,66	1605,53	3729,79	188	458	1550335	1557503	2891	6167	4923	10879
12	3053,34	2369,32	5456,86	175	505	201983	482047	4527	9599	7903	15607
16	3010,47	3225,70	7364,33	190	527	452095	461567	6231	12983	10519	19855
20	3048,86	3994,47	9064,90	148	522	288767	400127	7767	15807	12727	23471
24	3105,23	4744,41	10664,65	203	509	580095	595455	9223	18383	14663	27023
28	3126,31	5493,22	12334,95	203	447	514047	531455	10871	21135	16815	31247
32	3182,50	6201,21	13807,61	178	546	529407	610303	12047	23279	18255	34463
Laptop - MongoDB											
1	5345,68	95,17	273,08	66	199	101503	77503	121	334	147	432
4	8335,17	249,43	701,49	64	213	253695	260223	511	1039	734	1543
8	8345,64	562,27	1344,83	57	190	112319	163071	1097	2113	1580	4523
12	8201,43	893,07	2024,22	63	210	142463	172031	1737	3401	3283	8639
16	8676,94	1158,24	2516,74	63	223	144639	214527	2175	4259	4507	10575
20	8835,13	1438,42	3074,36	62	222	147199	220415	2687	5191	5191	12495
24	8378,79	1828,90	3879,42	63	229	178047	223487	3373	6535	6827	25647
28	8407,60	2141,92	4496,35	62	204	148991	198655	3911	7499	7887	28735
32	8512,74	2428,27	5061,91	65	237	134655	241023	4507	8487	8263	24175