PODATKOVNE BAZE NOSQL

Aljaž Zrnec, Dejan Lavbič, Lovro Šubelj, Slavko Žitnik, Aleš Kumer, Marko Bajec
Univerza v Ljubljani , Fakulteta za računalništvo in informatiko, Laboratorij za podatkovne tehnologije, Tržaška 25, 1000 Ljubljana
aljaz.zrnec, dejan.lavbič, lovro.subelj, slavko.zitnik, ales.kumer, marko.bajec@fri.uni-lj.si

Povzetek

Kratica NOSQL pomeni »Not Only SQL« in se jo pogosto povezuje z novo skupino podatkovnih baz, ki so se pojavile kot odgovor na težave, s katerimi se srečujemo pri uporabi relacijski baz podatkov. Za NOSQL podatkovne baze je težko podati natančno opredelitev, lahko pa trdimo, da zanje veljajo naslednje pomembne lastnosti: nimajo opredeljene sheme, prožnost, drobljenje, asinhrona replikacija, BASE pristop namesto ACID in arhitektura brez skupne rabe.

Podatkovne baze NOSQL ne predstavljajo najboljše rešitve za reševanje vsakdanjih problemov povezanimi z upravljanjem s podatki. Nastale so na podlagi zahtev po visokih učinkovitosti in skalabilnosti v okolju kakršno je svetovni splet. Testirali smo več vrst NOSQL podatkovnih baz, ki se med seboj razlikujejo glede na namen uporabe. Za izbrane scenarije smo primerjali zmogljivosti ključ-vrednost podatkovne baze, podatkovne baze tipa razširljiv zapis in dokumentno usmerjene podatkovne baze. Rezultate smo primerjali tudi z zmogljivostmi klasične relacijske podatkovne baze MySQL. Na podlagi ugotovitev smo podali priporočila kdaj je smiselno uporabiti katero izmed naštetih rešitev.

Abstract

NOSQL DATABASES

NOSQL acronym stands for "Not Only SQL" and is associated with a new group of databases that have arisen in response to problems encountered in the use of relational databases. It is difficult to give a precise definition for NOSQL databases, but it can be argued that they are subject to the following important features: no defined schema, flexibility, shredding, asynchronous replication, BASE approach instead of ACID approach and architecture without sharing.

NOSQL databases do not represent the best solutions to solve everyday problems related to data management. They evolved from requirements for high performance and scalability in an environment such as the World Wide Web. We tested several types of NOSQL databases, which differ depending on their applications. For a given scenarios we tested the performance of the following NOSQL solutions: key-value database, extensible record database and document-oriented database. The results were also compared with the performance of classical relational database MySQL. Based on the findings we made the recommendation when it makes sense to use any of the above solution.

Ključne besede

NOSQL, podatkovna baza, nerelacijska podatkovna baza, ključ-vrednost podatkovne baze, podatkovna baza tipa razširljiv zapis, dokumentne podatkovne baze

Key words

NOSQL, database, nonrelational database, key-value database, extensible record database, document-oriented database

1. UVOD

NOSQL podatkovne baze niso bile razvite z namenom popolne zamenjave relacijskih. Pri nekaterih rešitvah bi nas lahko toga shema relacijskih podatkovnih baz ovirala in so

nerelacijske baze s svojo fleksibilnostjo prava izbira [7]. Prav tako relacijske baze niso najbolj primerne za reševanje problemov z ogromno količino podatkov. Podatkovne strukture nerelacijskih podatkovnih baz se lahko nahajajo na tisočih računalnikih in vsebujejo več tera bytov podatkov. Pri tem pa podatkovne baze avtomatično skrbijo za želeno stopnjo konsistentnosti in poskrbijo, da problemi s posameznimi strežniki ne ogrozijo celotne gruče.

V okviru prispevka predstavimo ključne lastnosti posameznih vrst testiranih NOSQL podatkovnih baz in podamo glavne prednosti pred relacijskimi podatkovnimi bazami. V nadaljevanju med seboj primerjamo zmogljivosti štirih vidnejših predstavnikov NOSQL podatkovnih baz z uporabo štirih različnih scenarijev in rezultate primerjamo z zmogljivostmi klasične relacijske baze. V zaključku prispevka navedemo področja, kjer je uporaba NOSQL rešitev bolj smiselna od uporabe relacijskih podatkovnih baz.

2. KLJUČNE LASTNOSTI TESTIRANIH NOSQL PODATKOVNIH BAZ

Čeprav NOSQL podatkovne baze prinašajo veliko prednosti, to ne pomeni, da bi morali relacijske podatkovne baze opustiti [6, 5]. Relacijske in NOSQL podatkovne baze se med seboj bistveno razlikujejo in so izdelane za različne potrebe. Relacije in ACID transakcije so v določenih primerih, kot so borze in bančni sistemi, še vedno potrebne. Podatki morajo biti namreč v teh primerih vedno razpoložljivi in pravilni, saj pri upravljanju s finančnimi podatki ne sme prihajati do napak. Na drugi strani pa poznamo aplikacije za socialna omrežja, kjer sta bolj kot konsistentnost pomembni skalabilnost in visoka razpoložljivost. Če se zadnja objava uporabnika pojavi na strani šele čez nekaj minut, to ni kritičnega pomena. V nadaljevanju so predstavljene nekatere značilnosti, v okviru testiranja obravnavanih, NOSQL podatkovnih baz.

2.1 Ključ-vrednost podatkovne baze

Ključ-vrednost podatkovne baze veljajo za temelj, na katerem so zasnovani vsi ostali tipi podatkovnih baz NOSQL. Večina NOSQL podatkovnih baz namreč za shranjevanje uporablja mehanizem ključ-vrednost, pa čeprav morda to na prvi pogled ni razvidno.

Podatkovni model pri teh bazah je enostaven in temelji na množici parov ključ-vrednost. Vsak ključ je unikaten in se preslika v pripadajočo vrednost. Pogosto je dolžina ključev, ki jih je potrebno shraniti, omejena na določeno število bajtov, medtem ko je glede vrednosti manj omejitev. Vrednosti so tako lahko poljubnega podatkovnega tipa, saj jih podatkovna baza namreč vedno shranjuje kot objekt BLOB.

Shranjevanje podatkov v obliki ključ-vrednost je pravzaprav mehanizem, ki ga uporabljajo predpomnilniki. Predpomnilnik je hitri pomnilnik, ki vsebuje največkrat uporabljene podatke aplikacije, z namenom razbremeniti počasnejši disk. Podatkovne baze tipa ključ-vrednost so podobne predpomnilnikom, saj omogočajo hiter dostop do podatkov, ki so običajno majhne in enostavne zbirke atributov in vrednosti.

Membase, Tokyo Cabinet in Redis so trije predstavniki implementacije podatkovnih baz tipa ključ-vrednost. Med omenjenimi je Redis, ki smo ga testirali, poseben primer, saj lahko ključ nastopa v obliki različnih podatkovnih struktur, kot so nizi, seznami in množice. Prav tako omogoča zelo bogat nabor operacij za dostopanje do podatkov teh različnih tipov podatkovnih struktur.

2.2 Podatkovne baze tipa razširljiv zapis

Podatkovnim bazam tipa razširljiv zapis (Extensible record stores) rečemo tudi shrambe s širokimi stolpci (Wide column datastores), vendar večina avtorjev uporablja prvo poimenovanje. Uvrščamo jih v skupino stolpično usmerjenih shramb (Column oriented datastores). Sem spadajo tudi stolpično usmerjene podatkovne baze (Column oriented databases), ki pa ne spadajo v skupino NOSQL. Gre namreč za relacijske podatkovne baze, kjer se operacije izvajajo nad stolpci in ne nad vrsticami.

Podatkovne baze tipa razširljiv zapis so osnovane na podlagi modela BigTable [3], tj. Googlovega porazdeljenega sistema za shranjevanje podatkov. Osnovni koncept njihovega podatkovnega modela je delitev tako vrstic kot stolpcev preko več vozlišč (vertikalno in horizontalno particioniranje).

Poleg Google BigTable, ki predstavlja temelj večini shramb tipa razširljiv zapis, sem spadajo še Cassandra, HBase in Hypertable. Med naštetimi smo testirali podatkovni bazi Cassandra in in HBase.

2.3 Dokumentno usmerjene podatkovne baze

Dokumentne podatkovne baze so v osnovi podatkovne baze tipa ključ-vrednost, ki za shranjevanje podatkov uporabljajo bolj kompleksne podatkovne strukture, to je dokumente. Beseda "dokument" v primeru dokumentne podatkovne baze predstavlja nabor parov tipa ključ-vrednost, ki so strukturiran v obliko dokumenta. Povod za tako strukturo so podatki, ki danes ne nastopajo več v tako enostavnih oblikah kot so preproste vrstice ali stolpci. Pogosto so namreč nahajajo v obliki XML ali JSON datotek, saj so te tehnologije visoko prenosljive, kompaktne in standardizirane. Namesto da se XML in JSON dokumenti preslikajo v relacijsko obliko, je veliko bolj smiselno uporabiti dokumentne podatkovne baze. Te so brez sheme, saj za XML/JSON dokumente ni vnaprej določenega formata, tako da je vsak dokument neodvisen od drugih.

Danes je na voljo nekaj različnih odprtokodnih dokumentnih podatkovnih baz, npr. SimpleDB, Terrastore, najbolj obetavni med njimi pa sta MongoDB in CouchDB. V okviru testiranja smo uporabili MongoDB.

3. TESTNO OKOLJE

Ker nismo imeli na razpolago dovolj fizičnih strežnikov, smo se odločili, da testno okolje namestimo v Amazonov EC2 oblak. Amazon ponuja različne konfiguracije virtualnih strežnikov, in sicer od povsem osnovnih s 600 MB pomnilnika in eno računsko enoto, do takih z 68 GB pomnilnika in 26 računskimi enotami. Za potrebe testiranja smo izbrali konfiguracijo Large, ki se ponaša z naslednjimi karakteristikami:

- 7.5 GB pomnilnika,
- 4 EC2 računske enote,
- 850 GB lokalne hrambe podatkov,
- visoka hitrost dostopa do trajne podatkovne shrambe,
- 64-bitna platforma.

Pojem EC2 računske enote je Amazon uvedel zato, da bi uporabnikom zagotovil približno enake karakteristike na zelo različni fizični opremi. Tako naj bi bila ena EC2 računska enota primerljiva z 1.0 - 1.2 GHz 2007 Opteron ali 2007 Xeon procesorjem [2]. V času izvajanja testiranj je operacijski sistem v Large primerku virtualnega strežnika prepoznal dvo-jedrni 2.66Hz Intel Xeon procesor, kar ustreza približno štirim EC2 računskim enotam, kot jih obljublja Amazon.

Testiranje zmogljivosti je potekalo na štirih Large virtualnih strežnikih. Arhitektura Amazon EC2 virtualnih strežnikov omogoča, da so nekateri viri dodeljeni posameznemu strežniku, npr. procesorska moč in pomnilnik, nekateri pa se delijo med več strežnikov, npr. mrežna povezava in diskovni podsistem. Na svojih testnih strežnikih smo uporabili EBS (Amazon Elastic Block Store) diskovni podsistem [1]. Do EBS naprav dostopa strežnik preko mrežnih povezav. Konfiguracija strežnika Large ima hitrost dostopa omejeno na 1Gbit/s. Predvidevamo lahko, da imajo baze, ki pogosto pišejo ali berejo s trdega diska, zaradi te lastnosti slabše izhodišče pri testiranju. Amazon sicer ponuja možnost povezovanja več EBS enot v RAID polje, s čimer je možno povečati zmogljivost vhodno/izhodnega sistema, vendar se v okviru izvajanja meritev nismo spuščali v to vrsto optimizacije.

Amazon za spremljanje uporabe virov na virtualnih strežnikih ponuja svojo storitev CloudWatch. Viri, ki smo jih spremljali na strežnikih so: uporaba CPU-ja, število pisanj in branj na disk ter količina podatkov, ki se bere ali piše. Kot klienta za izvajanje testov smo uporabili en EC2 Large virtualni strežnik na isti lokaciji, kot so tekle podatkovne baze. En sam klient je imel dovolj zmogljivosti, da je lahko generiral zahteve in ni predstavljal omejitve.

3.1 Programska oprema

Na virtualne strežnike smo namestili operacijski sistem Ubuntu 10.04 LTS za katerega obstajajo tudi zagonske slike za Amazon EC2. V okviru primerjave zmogljivosti smo testirali naslednje NOSQL podatkovne baze:

- Redis 2.4.3, Jedis JAVA klient 2.0.0 (ključ-vrednost podatkovna baza),
- MongoDB 2.0.1, MongoDB Java klient 2.6.5 (dokumentna podatkovna baza),
- Cassandra 0.8.7 podpora YCSB (podatkovna baza tipa razširljiv zapis) in
- HBase 0.90.4 (podatkovna baza tipa razširljiv zapis).

Za primerjavo rezultatov z zmogljivostjo relacijske podatkovne baze smo uporabili podatkovno bazo:

- MySQL 5.1.41-3ubuntu12.10, MySQL JDBC klient Connector/J 5.1.18

Za zagon gruče s HBase in Cassandro smo uporabili odprtokodni projekt Whirr. Apache Whirr je skupek knjižnic, ki nudijo enostaven zagon podprtih podatkovnih baz na Amazon strežnikih. Trenutno podprta programska oprema je Cassandra, Hadoop, ZooKeeper, HBase, elasticsearch, Voldamort in Hama. Za zagon Mongo, Redis in MySQL gruč so bile izdelane lastne skripte.

3.2 Orodje za izvajanje testiranja

Za izvajanje testov smo uporabili odprtokodno orodje imenovano YCSB (Yahoo! Cloud System Benchmark) [4]. Pri Yahoo-ju so orodje razvili prav z namenom iskanja najboljših rešitev za shranjevanje podatkov v porazdeljenih podatkovnih bazah. Že napisani odjemalci obstajajo za podatkovne baze MongoDB, Cassandro, HBase, Voldemort, Infinispan, kot tudi za podatkovne baze z JDBC podporo. Podpora za Redis prav tako obstaja, vendar smo morali dodati podporo za porazdelitev obremenitve na več Redis strežnikov. Prav tako lahko sami definiramo lastne delovne obremenitve, s čimer lahko bolje simuliramo svojo aplikacijo. Delovne obremenitve definirajo podatke, katere smo vnašali v podatkovne baze in transakcije, katere smo izvajali v okviru meritev.

3.3 Nastavitve podatkovnih baz

Podatkovne baze smo namestil na 4 podatkovne strežnike. Naše poznavanje nastavitev za optimizacijo posameznih podatkovnih baz ni bilo za vse baze enako. Ker bi lahko z večjo optimizacijo posameznih baz nepravično vplivali na rezultate meritev, smo se odločili, da podatkovne baze nastavimo le toliko, da delujejo v porazdeljenem načinu na štirih strežnikih. Arhitekturo gruč smo poizkušali izbrati tako, da so med seboj čim bolj primerljive. Še posebej Cassandra in HBase omogočata zelo natančno nastavitev želenih lastnosti. Določimo lahko na koliko strežnikov se določeni podatki replicirajo in s tem nastavljamo razmerje med hitrostjo zapisovanja, hitrostjo branja in varnostjo podatkov.

3.4 Testni scenariji

Izbrane podatkovne baze se zelo razlikujejo po svoji arhitekturi in zaradi tega ni vsaka primerna za vse tipe uporabe [8]. Za ugotavljanje primernosti delovanja posameznih podatkovnih baz v določenih okoliščinah smo opredelili štiri scenarije uporabe, ki jih prikazuje Tabela 1.

Scenarij A	**Scenarij B**	**Scenarij C**	**Scenarij D**
Scenarij A simulira pisanje velike količine podatkov v dnevnik.	Scenarij B simulira uporabo, kjer večino dostopov predstavlja branje. Zahtevani podatki so novejši. Scenarij predstavlja spletno stran, na kateri je zanimiva novejša vsebina.	Scenarij C predstavlja sporočilni sistem, kjer se podatki berejo in pišejo v približno enakem razmerju.	Scenarij D uporablja za porazdelitev branj Zipfov zakon. Prevladujejo branja, pisanj je le en odstotek.
• delež branj 0%, • delež posodobitev 0%, • delež pisanj 95%, • delež iskanj 5%, • iskanje uporablja novejše podatke.	• delež branj 98%, • delež posodobitev 1%, • delež pisanj 1%, • delež iskanj 0%, • branje uporablja novejše podatke.	• delež branj 50%, • delež posodobitev 0%, • delež pisanj 50%, • delež iskanj 0%, • branje uporablja novejše podatke.	• delež branj 99%, • delež posodobitev 0%, • delež pisanj 1%, • delež iskanj 0%, • berejo se podatki porazdeljeni po Zipfovem zakonu.

Tabela 1. Uporabljeni testni scenariji

3.5 Struktura testnih podatkov

Zapisi v podatkovnih bazah so bili sestavljeni iz ključa in 15-ih tekstovnih polj. Vsako tekstovno polje je vsebovalo natančno 100 znakov. Približna velikost enega zapisa je tako znašala 1.5 kB. Tukaj so lahko nastopila odstopanja, saj vsaka podatkovna baza shranjuje podatke v svojem formatu. Cassandra je na primer v okviru vsakega polja shranila tudi

časovno oznako zadnjega popravka. V podatkovne baze smo pred izvajanjem testiranj shranili 10.000.000 zapisov.

4. ANALIZA

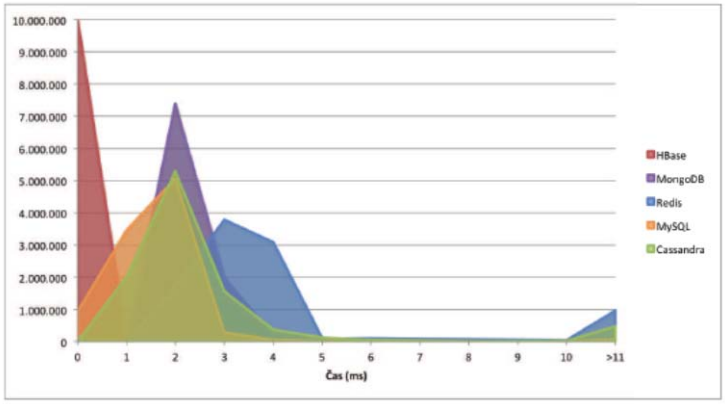
4.1 Začetno polnjenje s podatki

Začetni vnos podatkov kaže, kako hitro lahko izbrane podatkovne baze shranjujejo podatke. Tabela 2 prikazuje podatke o hitrosti vnosa podatkov v opazovane podatkovne baze. Najhitreje je 10 milijonov zapisov shranila MySQL gruča s precej višjim povprečnim številom operacij na sekundo kot vse ostale podatkovne baze. Prav tako je pri tej bazi dosežen najnižji povprečen čas zapisa. HBase je druga na seznamu kljub temu, da je HBase gruča nekoliko drugačna kot vse ostale in ima na voljo samo 3 vozlišča (eno vozlišče se uporablja izključno za izvajanje nadrejenih procesov) za shranjevanje podatkov. Zanimivo je tudi, da je HBase zapisal 99 odstotkov zahtev v manj kot 0 ms. To prikazuje tudi slika Slika 1.

	Čas izvajanja	Št. operacij /s	Povprečen čas	99% zahtev (ms)	95% zahtev (ms)
Redis	66,6 min	2734	7,025 ms	61	16
MongoDB	61,5 min	2771	7,1165 ms	24	4
Cassandra	23,8 min	7053	5,477 ms	71	9
HBase	19,4 min	8908	4,3945 ms	0	0
MySQL	14,4 min	11963	3,275 ms	7	3

Tabela 2. Statistika vnosov v podatkovne baze

Redis podatkovna baza je bila edina, ki ni shranila vseh zapisov. Približno 800.000 zapisov ni bilo uspešno shranjenih. Časovna omejitev povezave je pri Redisu nastavljena na precej majhno vrednost in je bila pri tako velikem številu zahtev pogosto prekoračena. Podobno zakasnitev smo opazili tudi pri podatkovni bazi HBase, kjer je bil najdaljši čas izvajanja ene zahteve kar 58 sekund.



Slika 1. Začetni vnos podatkov

4.2 Scenarij A

Prve meritve scenarija A smo izvajali tako, da je orodje YCSB maksimalno obremenilo strežnike. To se je kmalu izkazalo za problematično. Strežniki so po nekaj minutah postali preobremenjeni in število operacij na sekundo je padlo na 0. Zaradi tega smo se odločili, da število operacij na sekundo, ki jih sproži orodje YCSB, omejimo na 1500 in 2.700.000 operacij skupno. Če bodo to število operacij podatkovne baze zmogle izvesti, bo test končan

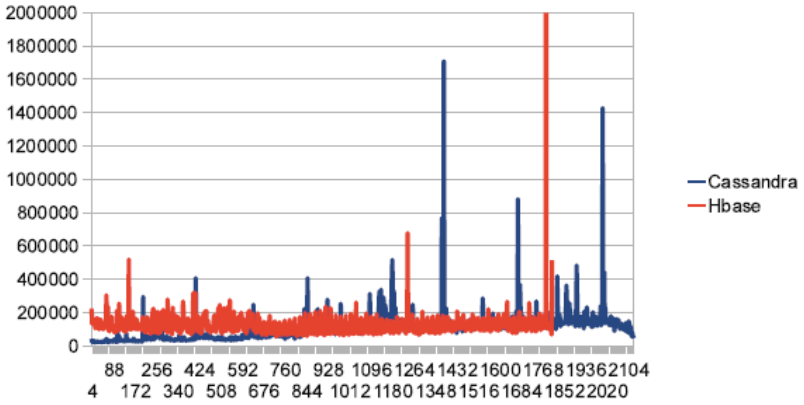
po pol ure. MySQL JDBC gonilnik, ki smo ga uporabili, ni dokončal nobene operacije iskanja, saj so ta vrnila preveč podatkov in klientu je zmanjkalo pomnilnika. Scenarija A z MySQL bazo nismo dokončali. Pri Redis podatkovni bazi prav tako nismo mogli dokončati testa. Redis strežniški procesi so se konstantno ugašali in po nekaj minutah smo test prekinili. Rezultate HBase, Cassandra in MongoDB podatkovnih baz prikazuje Tabela 3.

	Čas izvajanja	Št. operacij /s	Povprečen čas iskanja (ms)	Povprečen čas pisanj (ms)
Cassandra	36,5 min	1232	91,578	3,041
HBase	30,2 min	1493	109,832	0,162
MongoDB	97 min	461	136,545	15,5
Redis	/	/	/	/
MySQL	/	/	/	/

Tabela 3. Statistika scenarija A

Pri HBase bazi, kljub dodatnim iskanjem, ni bilo opaziti nobenih zakasnitev pri hitrosti zapisov. Po drugi strani se je pri podatkovni bazi MongoDB čas zapisov zelo povečal, čeprav smo izvajali precej manj pisanj kot pri začetnem vnosu podatkov. Cassandra je bila najhitrejša pri iskanjih in nekaj počasnejša pri pisanjih od HBase baze.

Slika 2 prikazuje povprečno hitrost iskanja v določenem trenutku izvajanja testa. Vidi se, da Cassandra na začetku vrača rezultate hitreje kot HBase potem pa se približno izenačita. Čeprav ima Cassandra več trenutkov, ko se čas iskanja poveča, je skupni povprečni čas še vedno manjši.



Slika 2. Povprečna hitrost iskanja

4.3 Scenarij B

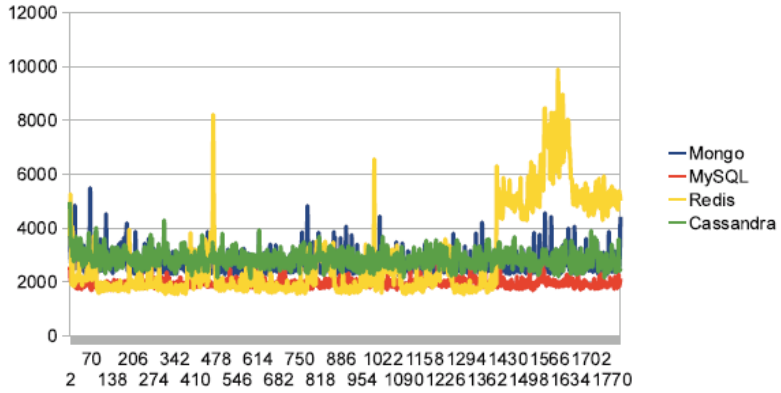
Enako kot pri scenariju A, smo tudi pri scenariju B omejili skupno število operacij na 2.700.000 in 1500 operacij na sekundo. Vse podatkovne baze razen HBase so s scenarijem opravile v optimalnem času. Pri HBase nas je zelo presenetil slab čas branj, ki jih je bilo v scenariju B kar 98 odstotkov. Kot je razvidno iz tabele Tabela 4, je HBase v povprečju potreboval kar 19ms za operacijo branja. Interno HBase operacije branja razume kot iskanje. HBase nima indeksov, ki bi omogočali dostop do posamezne vrstice ali stolpca. Najmanjša enota je blok v HFile datoteki, katero moramo nato preiskati, da dobimo želen podatek. Kljub temu, da branja trajajo nekaj dlje, pa HBase še vedno zapisuje najhitreje.

	Čas izvajanja	Št. operacij /s	Povprečen čas	Povprečen čas	Povprečen čas

			posodobitve (ms)	pisanj (ms)	branj (ms)
Cassandra	30 min	1500	2,3	2,488	2,84
HBase	85 min	529	0,024	0,047	19,22
MongoDB	30 min	1500	4,153	4,291	2,62
MySQL	30 min	1500	2,016	2,165	1,94
Redis	30 min	1500	2,768	6,281	2,87

Tabela 4. Statistika scenarija B

Slika 3 prikazuje razmerje med podatkovnimi bazami pri branju. MySQL in Redis sta na začetku testiranja še nekako enakovredna, proti koncu pa je Redis počasnejši od vseh. Redis tudi pri tem testu ni uspešno dokončal vseh operacij. Neuspešno je bilo izvedenih 3000 pisanj in posodobitev ter približno 460.000 branj.



Slika 3. Povprečna hitrost branja

4.4 Scenarij C

Pri scenariju C smo skupno število operacij zmanjšali na 1.500.000, število operacij na sekundo pa obdržali na 1500. Ponovno smo imeli veliko problemov s podatkovno bazo Redis. Nikakor nam ni uspelo uspešno zaključiti izvajanje meritev. Redis pri velikem številu pisanj ne zmore dovolj hitro shranjevati podatkov na disk, zaradi česar se procesi ugasnejo. HBase trpi za enako hibo kot v scenariju B, branja podatkov niso dovolj hitra in tudi v tem scenariju jih je bilo veliko. Pri Cassandri se lepo vidi kako večje število pisanj vpliva na čas branj. Branja vzamejo skoraj deset krat več časa kot pri scenariju B.

	Čas izvajanja	Št. operacij /s	Povprečen čas pisanj (ms)	Povprečen čas branj (ms)
Cassandra	30 min	839	2,183	21,157
HBase	45 min	551	0,106	36,011
MongoDB	16,6 min	1500	2,981	2,596
MySQL	16,6 min	1500	4,768	3,788
Redis	/	/	/	/

Tabela 5. Statistika scenarija C

Časi branj posameznih podatkovnih baz, ki jih prikazuje Tabela 5, se precej razlikujejo. Eno izmed možnih razlag ponuja tudi vpogled v to, koliko podatkov se prebere iz diska. Podatke smo pridobili iz Amazon CloudWatch storitve za posamezna vozlišča v gruči in jih sešteli po podatkovnih bazah in časih, ko se je izvajal scenarij C.

- MySQL: 68 kiloBytov,

- MongoDB: 99 megaBytov,
- Cassandra: 26 gigaBytov,
- HBase: 62.5 gigaBytov.

Razlike so skoraj neverjetne. MySQL se v malo več kot 16 minutah in skoraj 800.000 branjih skoraj ne dotakne diska vse zahteve se postrežejo iz pomnilnika. Medtem ko HBase prebere 62 gigaBytov.

4.5 Scenarij D

Pri scenariju D so se vse podatkovne baze razen HBase zelo dobro odrezale, zato smo povečali število operacij na sekundo v testu na 2500. Pri testiranju HBase se test ni izvedel v celoti niti pri 1500 operacijah na sekundo. Redis je kljub dobrim rezultatom in časi primerljivimi z ostalimi, neuspešno opravil približno 10 odstotkov operacij. Z MySQL, Redis in MongoDB smo test izvedli še pri večjih obremenitvah. Največje število operacij scenarija D je zmogel MySQL kar 9166, sledi Redis s 4650 in MongoDB s 3577 operacijami. Rezultate testiranja prikazuje Tabela 6.

	Čas izvajanja	Št. operacij /s	Povprečen čas pisanj (ms)	Povprečen čas branj (ms)
Cassandra	18 min	2500	2,229	3,265
HBase	/	/	/	/
MongoDB	18 min	2500	4,424	2,651
MySQL	18 min	2500	2,252	1,901
Redis	18 min	2500	5,082	2,017

Tabela 6. Statistika scenarija D

5. ZAKLJUČEK

Rezultati kažejo, da pri strukturiranih podatkih, enostavnih poizvedbah in indeksu samo na primarnem ključu, ne-relacijske baze ne uspejo slediti hitrosti MySQL podatkovne baze. Tukaj je potrebno poudariti, da sta bila Redis in MySQL v privilegiranem položaju, saj je za porazdelitev podatkov na pravo vozlišče skrbel klient in ne podatkovna baza. MongoDB, Cassandra in HBase imajo vse vgrajene algoritme, ki podatke avtomatsko razdelijo po vozliščih. Pri 100 ali več vozliščih bi bil ročni nadzor porazdeljevanja podatkov praktično nemogoč. Je pa pri tako velikem številu vozlišč izpad vozlišča zelo verjeten dogodek. Zato bi bilo tudi zanimivo primerjati, kako se hitrosti poizvedb in zapisovanja spremenijo v primeru izpada vozlišča in kako hitro v takem primeru podatkovne baze podatke prenesejo na druga vozlišča.

V splošnem lahko kot prednosti NOSQL podatkovnih baz izpostavimo njihovo izjemno skalabilnost in razpoložljivost ter nizko ceno. Najprimernejše situacije za uporabo NOSQL podatkovnih baz so sledeče: preprost podatkovni model, prilagodljivost je pomembnejša od strogega nadzora nad opredeljeno podatkovno strukturo, zahtevana visoka zmogljivost, stroga podatkovna konsistentnost ni nujna in računalništvo v oblaku.

Ne gre torej za vprašanje, ali so podatkovne baze NOSQL boljše od relacijskih, ampak predvsem za vprašanje, kdaj je smiselno uporabiti baze tipa NOSQL. Odgovor je, kadar se soočamo z zahtevami za poizvedbe po obsežnih podatkovnih naborih z veliko hitrostjo izvajanja poizvedb. Tu nastopijo baze NOSQL. Gre za podatkovne nabore, kamor spadajo

npr. spletni dnevniki, transakcije nakupov, proizvodni podatki iz naprav na tekočem traku, znanstvene zbirke podatkov za izvajanje različnih analiz itd., ki se kopičijo v velikem številu vsako sekundo in za njihovo shranjevanje in obdelavo SUPB ni dovolj zmogljiva in hitra rešitev.

Na drugi strani pa, če potrebujemo konsistentne podatke, obstojna pisanja, ACID transakcije in imamo zapletene podatkovne strukture in razmerja, je tradicionalna relacijska podatkovna baza še vedno najboljša rešitev. Področja, ki so primerna za relacijske podatkovne baze lahko vključujejo katerekoli zapletene poslovne aplikacije, zlasti finančno usmerjene aplikacije, kjer so celovitost transakcij, konsistentnost in trajnost nujne zahteve. Primeri so tudi OLTP obdelave, kjer še vedno zmaguje kombinacija kakovosti podatkov in zmogljivosti dobro načrtovanega SUPB-ja.

VIRI IN LITERATURA

[1] Amazon EBS (2012). Dostopno na: http://aws.amazon.com/ebs/

[2] Amazon EC2 FAQs (2012). Dostopno na: http://aws.amazon.com/ec2/faqs/#What_is_an_EC2_Compute_Unit_and_why_did_you_introduce_it

[3] CHANG, F., J. DEAN, S. GHEMAWAT, W. C. HSIEH, D. A. WALLACH, M. BURROWS, T. CHANDRA, A. FIKES, R. E. GRUBER (2006). Bigtable: A Distributed Storage System for Structured Data, OSDI 2006

[4] Github (2012). Yahoo! Cloud Serving Benchmark (YCSB), Dostopno 10.1.2012 na https://github.com/brianfrankcooper/YCSB/wiki

[5] KNEZ, N. (2012). Analiza podatkovnih baz NOSQL in izdelava odločitvenega modela za izbiro med relacijskimi in NOSQL podatkovnimi bazami. Diplomska naloga v nastajanju.

[6] LEAVITT, N. (2010). Will NOSQL Databases Live Up to Their Promise?, IEEE Xplore digital library, Vol. 43, No. 2, pp. 12-14.

[7] STONEBRAKER, M. (2010). SQL databases vs. NOSQL databases, ACM Digital Library, Vol. 53, No. 4.

[8] ŽLENDER, R. (2011). Primerjava zmogljivosti nerelacijskih podatkovnih baz, Fakulteta za računalništvo in informatiko, Diplomsko delo