Podatkovne baze NoSQL

Aljaž Zrnec, 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, lovro.subelj, slavko.zitnik, ales.kumer, marko.bajec}@fri.uni-lj.si

Izvleček

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

Podatkovne baze NoSQL niso najboljša rešitev za reševanje vsakdanjih problemov, povezanih z upravljanjem s podatki. Nastale so na podlagi zahtev po visokih učinkovitosti in skalabilnosti v okolju, kakršno je svetovni splet. Testirali smo več vrst podatkovnih baz NoSQL, ki se razlikujejo med seboj glede na namen uporabe. Za izbrane scenarije smo primerjali zmogljivosti podatkovne baze ključ-vrednost, podatkovne baze vrste razširljivi 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.

Ključne besede: NoSQL, podatkovna baza, nerelacijska podatkovna baza, podatkovne baze ključ-vrednost, podatkovna baza vrste razširljivi zapis, dokumentne podatkovne baze.

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 solutions.

Keywords: NoSQL, database, nonrelational database, key-value database, extensible record database, document-oriented database.

1 UVOD

Podatkovne baze NoSQL niso bile razvite z namenom popolne zamenjave relacijskih baz. Pri nekaterih rešitvah bi nas lahko toga shema relacijskih podatkovnih baz ovirala in so nerelacijske baze s svojo fleksibilnostjo prava izbira (Žlender, 2011). Prav tako relacijske baze niso najbolj primerne za reševanje problemov, pri katerih se srečujemo s količinami podatkov reda petabajtov. Podatkovne strukture nerelacijskih podatkovnih baz se lahko nahajajo na tisočih računalnikih in vsebujejo več tera- ali petabajtov podatkov. Pri tem podatkovne baze samodejno skrbijo za želeno stopnjo celovitosti in poskrbijo, da problemi s posameznimi strežniki ne ogrozijo celotne gruče.

V prispevku predstavimo ključne lastnosti posameznih vrst testiranih podatkovnih baz NoSQL in podamo glavne prednosti pred relacijskimi podatkovnimi bazami. V nadaljevanju primerjamo zmogljivosti štirih vidnejših predstavnikov podatkovnih baz NoSQL z uporabo štirih različnih scenarijev in rezultate primerjamo z zmogljivostmi klasične relacijske baze. V sklepnem delu prispevka navedemo področja, na katerih je uporaba rešitev NoSQL bolj smiselna od uporabe relacijskih podatkovnih baz.

2 KLJUČNE LASTNOSTI TESTIRANIH PODATKOVNIH BAZ NOSQL

Čeprav podatkovne baze NoSQL prinašajo veliko prednosti, to ne pomeni, da bi morali opustiti relacijske podatkovne baze (Leavitt, 2010; Knez, 2012;

Lavbič, 2012). Relacijske podatkovne baze in podatkovne baze NoSQL se med seboj bistveno razlikujejo in so izdelane za različne potrebe. Relacije in transakcije ACID so še vedno potrebne v določenih primerih, kot so borze in bančni sistemi. 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 poznamo aplikacije za socialna omrežja, pri katerih sta bolj kot celovitost pomembni skalabilnost in visoka razpoložljivost. Če se zadnja objava uporabnika pojavi na strani šele čez nekaj minut, to ni ključnega pomena na uporabo storitve.

V nadaljevanju so predstavljene ključne lastnosti podatkovnih baz NoSQL, ki so izpeljanke osnovnega tipa podatkovnih baz NoSQL – podatkovne baze ključ-vrednost. Namen testiranja je bil primerjati učinkovitosti posameznih podatkovnih baz NoSQL iz te skupine in jih primerjati z učinkovitostjo najbolj razširjene relacijske podatkovne baze MySQL. Prav zaradi tega v primerjavo nismo vključili tudi kakšnega primerka podatkovne baze NoSQL iz skupine baz NoSQL, ki temeljijo na grafih.

Podatkovne baze ključ-vrednost

Podatkovne baze ključ-vrednost veljajo za temelj, na katerem so zasnovane vse druge vrste podatkovnih baz NoSQL. Večina podatkovnih baz NoSQL namreč za shranjevanje uporablja mehanizem ključ-vrednost, čeprav morda to ni razvidno na prvi pogled.

Podatkovni model pri teh bazah je preprost in temelji na množici parov ključ-vrednost. Vsak ključ je unikaten in se preslika v pripadajočo vrednost. Pogosto je dolžina ključev, ki jih je treba shraniti, omejena na določeno število bajtov, medtem ko je glede vrednosti manj omejitev. Vrednosti so tako lahko poljubnega podatkovnega tipa, saj jih podatkovna baza 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 vrste ključ-vrednost so podobne predpomnilnikom, saj omogočajo hiter dostop do podatkov, ki so običajno majhne in preproste zbirke atributov in vrednosti.

Membase, Tokyo Cabinet in Redis so trije predstavniki implementacije podatkovnih baz vrste 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 bogat nabor operacij za dostopanje do podatkov teh različnih vrst podatkovnih struktur.

Podatkovne baze vrste razširljivi zapis

Podatkovnim bazam vrste razširljivi zapis (Extensible record stores) rečemo tudi shrambe s širokimi stolpci (Wide column datastores), vendar večina avtorjev uporablja prvo poimenovanje (Cattell, 2010; Knez, 2012). Uvrščamo jih v skupino stolpično usmerjenih shramb (Column oriented datastores). Sem spadajo tudi stolpično usmerjene podatkovne baze (Column oriented databases), ki ne spadajo v skupino NoSQL. Gre namreč za relacijske podatkovne baze, pri katerih se operacije izvajajo nad stolpci in ne nad vrsticami.

Podatkovne baze vrste razširljivi zapis temeljijo na modelu BigTable (Chang, 2006), tj. Googlovem porazdeljenem sistemu 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 je temelj večini shramb vrste razširljivi zapis, sem spadajo še Cassandra, HBase in Hypertable. Med naštetimi smo testirali podatkovni bazi Cassandra in HBase.

Dokumentno usmerjene podatkovne baze

Dokumentne podatkovne baze so v bistvu podatkovne baze vrste ključ-vrednost, ki za shranjevanje podatkov uporabljajo bolj kompleksne podatkovne strukture, tj. dokumente. Beseda dokument v primeru dokumentne podatkovne baze pomeni nabor parov vrste ključ-vrednost, ki so strukturirani v obliko dokumenta. Povod za tako strukturo so podatki, ki danes ne nastopajo več v tako preprostih oblikah, kot so vrstice ali stolpci. Pogosto se namreč nahajajo v obliki datotek XML ali JSON (Andersen, 2010; Osborne, 2011; Terrastore, 2012), saj so te tehnologije visoko prenosljive, kompaktne in standardizirane. Namesto da dokumente XML in JSON preslikamo v relacijsko obliko, je veliko bolj smiselno uporabiti dokumentne podatkovne baze. Te so brez sheme, saj za dokumente XML/JSON ni vnaprej določenega formata, tako da je vsak dokument neodvisen od drugih.

Danes je na voljo nekaj različnih odprtokodnih dokumentnih podatkovnih baz, kot so npr. SimpleDB, Terrastore, najbolj obetavni med njimi 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 oblak EC2. 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 ima te lastnosti:

- 7,5 GB pomnilnika,
- 4 računske enote EC2,
- 850 GB lokalne hrambe podatkov,
- visoka hitrost dostopa do trajne podatkovne shrambe,
- 64-bitna platforma.

Pojem računske enote EC2 je Amazon uvedel zato, da bi uporabnikom zagotovil približno enake lastnosti na zelo različni fizični opremi. Tako naj bi bila ena računska enota EC2 primerljiva z 1,0–1,2 GHz procesorjem 2007 Opteron ali 2007 Xeon (Amazon EC2, 2012). V času izvajanja testiranj je operacijski sistem v Large primerku virtualnega strežnika prepoznal dvojedrni 2,66 Hz procesor Intel Xeon, kar ustreza približno štirim računskim enotam EC2, kot jih ponuja Amazon.

Testiranje zmogljivosti je potekalo na štirih virtualnih strežnikih Large. Arhitektura virtualnih strežnikov Amazon EC2 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 diskovni podsistem EBS (Amazon Elastic Block Store) (Amazon EBS, 2012). Do naprav EBS dostopa strežnik prek 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 enot EBS v polje RAID, s čimer je mogoče 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 virtualni strežnik EC2 Large na lokaciji, na kateri so tekle tudi podatkovne baze. En sam klient je imel dovolj zmogljivosti, da je lahko generiral zahteve in ni predstavljal omejitve.

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 te podatkovne baze NoSQL:

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

Za primerjavo rezultatov z zmogljivostjo relacijske podatkovne baze smo uporabili podatkovno bazo MySQL 5.1.41-3 Ubuntu 12.10, 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 omogočajo preprost zagon podprtih podatkovnih baz na strežnikih Amazon. Trenutno podprta programska oprema je Cassandra, Hadoop, ZooKeeper, HBase, elasticsearch, Voldamort in Hama. Za zagon Mongo, Redis in MySQL gruč so bila izdelana lastna skripta.

Orodje za izvajanje testiranja

Za izvajanje testov smo uporabili odprtokodno orodje YCSB (Yahoo! Cloud System Benchmark) (Github, 2006). Pri Yahooju so orodje razvili prav z namenom iskanja najboljših rešitev za shranjevanje podatkov v porazdeljenih podatkovnih bazah. Že napisani odjemalci obstajajo za podatkovne baze MongoDB, Cassandra, HBase, Voldemort, Infinispan, kot tudi za podatkovne baze s podporo JDBC. Podpora za Redis prav tako obstaja, vendar smo morali dodati podporo za porazdelitev obremenitve na več strežnikov Redis. 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.

Nastavitve podatkovnih baz

Podatkovne baze smo namestili na štiri podatkovne strežnike. Naše poznavanje nastavitev za optimizacijo posameznih podatkovnih baz ni bilo enako za vse baze. 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 poskušali izbrati tako, da so med seboj čim bolj primerljive. Še posebno Cassandra in HBase omogočata zelo natančno nastavitev želenih lastnosti. Določimo lahko, na koliko strežnikov se replicirajo določeni podatki, in s tem nastavljamo razmerje med hitrostjo zapisovanja in branja ter varnostjo podatkov.

Testni scenariji

Izbrane podatkovne baze se zelo razlikujejo po svoji arhitekturi in zaradi tega ni vsaka primerna za vse vrste uporabe (Žlender, 2011). Za ugotavljanje primernosti delovanja posameznih podatkovnih baz v določenih okoliščinah smo opredelili štiri scenarije uporabe, ki jih prikazuje tabela 1.

Tabela 1: **Uporabljeni testni scenariji**

Scenarij A	**Scenarij B**	**Scenarij C**	**Scenarij Č**
Scenarij A simulira pisanje velike količine podatkov v dnevnik.	Scenarij B simulira uporabo, pri kateri večino dostopov predstavlja branje. Zahtevani podatki so novejši. Scenarij predstavlja spletno stran, na kateri je zanimiva novejša vsebina.	Scenarij C predstavlja sporočilni sistem, pri katerem se podatki berejo in pišejo v približno enakem razmerju.	Scenarij Č 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.

Struktura testnih podatkov

Zapisi v podatkovnih bazah so bili sestavljeni iz ključa in petnajstih besedilnih polj. Vsako besedilno polje je vsebovalo natančno sto znakov. Približna velikost enega zapisa je tako znašala 1,5 kB. Pri tem so lahko nastopila odstopanja, saj vsaka podatkovna baza shranjuje podatke v svojem formatu. Cassandra je npr. v okviru vsakega polja shranila tudi časovno oznako zadnjega popravka. V podatkovne baze smo pred izvajanjem testiranj shranili 10,000.000 zapisov.

4 ANALIZA

Začetno polnjenje s podatki

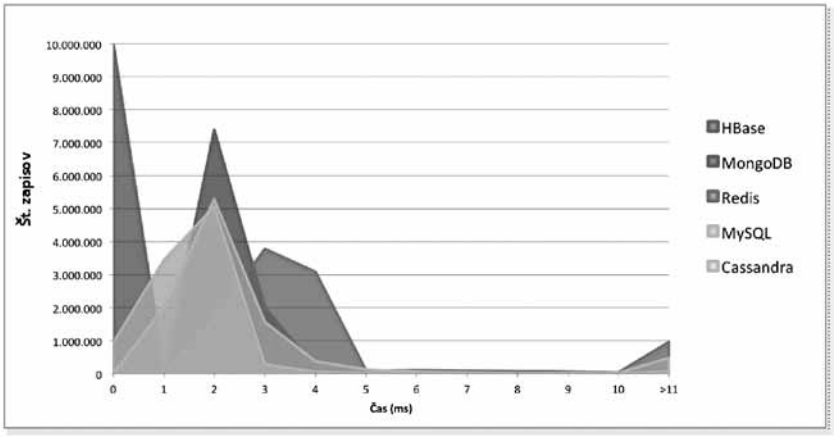
Začetni vnos podatkov kaže, kako hitro lahko izbrane podatkovne baze shranjujejo podatke. Tabela 2 prikazuje podatke o hitrosti vnosa podatkov v opazovane podatkovne baze. Najhitreje je 10 milijonov zapisov shranila gruča MySQL s precej višjim povprečnim številom operacij na sekundo kot vse druge podatkovne baze. Prav tako je pri tej bazi dosežen

Tabela 2: **Statistika vnosov v podatkovne baze**

	Čas izvajanja	**Štev. operacij/s**	**Povprečni čas pisanja zapisa**	**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

najnižji povprečni čas zapisa. HBase je druga na seznamu, kljub temu da je gruča HBase nekoliko drugačna kot vse druge in ima na voljo samo tri 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 1 ms/zapis. To prikazuje tudi slika 1.

Podatkovna baza Redis 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, pri kateri je bil najdaljši čas izvajanja ene zahteve kar 58 sekund.



Slika 1: **Porazdelitev časov zapisovanja pri začetnem vnosu podatkov**

Scenarij A

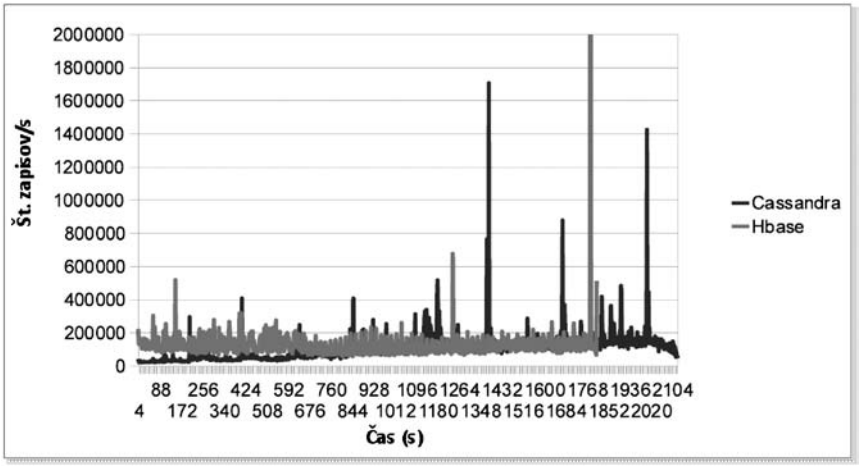
Prve meritve scenarija A smo izvajali tako, da je orodje YCSB maksimalno obremenilo strežnike. To se je kmalu izkazalo za problematično. Strežniki so po nekaj minutah postali preobremenjeni in število operacij na sekundo je padlo na 0. Zaradi tega smo se odločili, da število operacij na sekundo, ki jih sproži orodje YCSB, omejimo na 1500 in 2,700.000 operacij skupno. Če bodo podatkovne baze zmogle izvesti to število operacij, bo test končan po pol ure. Gonilnik JDBC, ki smo ga uporabili, ni dokončal nobene operacije iskanja, saj so ta vrnila preveč podatkov in je klientu zmanjkalo pomnilnika. Scenarija A z bazo MySQL nismo dokončali. Pri podatkovni bazi Redis prav tako nismo mogli dokončati testa. Strežniški procesi Redis so se konstantno ugašali in po nekaj minutah smo prekinili test. Rezultate podatkovnih baz HBase, Cassandra in MongoDB prikazuje tabela 3.

Tabela 3: **Statistika scenarija A**

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

Pri bazi HBase kljub dodatnim iskanjem ni bilo opaziti nobenih zakasnitev pri hitrosti zapisov. Po drugi strani se je pri podatkovni bazi MongoDB zelo podaljšal čas zapisov, čeprav smo izvajali precej manj pisanj kot pri začetnem vnosu podatkov. Cassandra je bila najhitrejša pri iskanjih, pri pisanju pa je bila približno devetnajstkrat počasnejša od baze HBase, vendar še vedno petkrat hitrejša od najpočasnejše baze MongoDB.

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



Slika 2: **Povprečna hitrost iskanja**

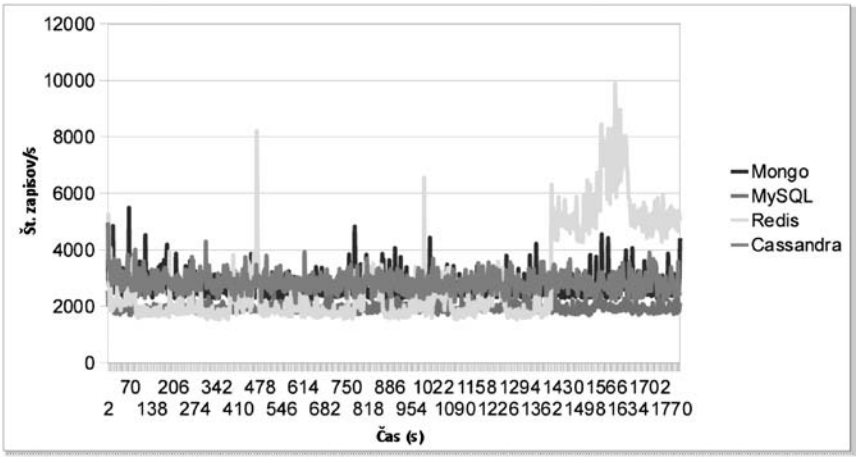
Scenarij B

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 4, je HBase v povprečju potreboval kar 19 ms 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 datoteki HFile, katero moramo nato preiskati, da dobimo želeni podatek. Kljub temu da branja trajajo nekaj dlje, HBase še vedno zapisuje najhitreje.

Tabela 4: **Statistika scenarija B**

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

Slika 3 prikazuje razmerje med podatkovnimi bazami pri branju. MySQL in Redis sta na začetku testiranja še nekako enakovredna, vendar je Redis proti koncu najhitrejši 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**

Scenarij C

Pri scenariju C smo skupno število operacij zmanjšali na 1,500.000, število operacij na sekundo je ostalo nespremenjeno, to je 1500. Ponovno smo imeli veliko težav s podatkovno bazo Redis. Nikakor nam ni uspelo uspešno končati izvajanja 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 osemkrat več časa kot pri scenariju B.

Tabela 5: **Statistika scenarija C**

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

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

- MySQL: 68 kilobajtov
- MongoDB: 99 megabajtov
- Cassandra: 26 gigabajtov
- HBase: 62,5 gigabajtov

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 gigabajtov.

Scenarij Č

Pri scenariju Č 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 s časi, primerljivimi z drugimi, neuspešno opravil približno 10 odstotkov operacij. Z MySQL, Redis in MongoDB smo naknadno izvedli še dodaten test pri večjih obremenitvah (rezultati teh testov niso prikazani v tabeli). Največje število operacij scenarija Č je zmogel MySQL (kar 9166), sledi Redis s 4650 in MongoDB s 3577 operacijami. Rezultate testiranja prikazuje tabela 6.

Tabela 6: **Statistika scenarija Č**

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

5 SKLEP

Rezultati kažejo, da pri strukturiranih podatkih, preprostih poizvedbah in indeksu samo na primarnem ključu, nerelacijske baze ne uspejo slediti hitrosti podatkovne baze MySQL. Ob tem je treba poudariti, da sta bila Redis in MySQL v privilegiranem položaju, saj je za porazdelitev podatkov na pravo vozlišče skrbel odjemalec in ne podatkovna baza. MongoDB, Cassandra in HBase imajo vse vgrajene algoritme, ki podatke samodejno porazdelijo po vozliščih. Pri sto ali več vozliščih bi bil ročni nadzor porazdeljevanja podatkov tako rekoč nemogoč. Vendar je 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 podatkovnih baz NoSQL izpostavimo njihovo izjemno skalabilnost in razpoložljivost ter nizko ceno. Najprimernejše situacije za uporabo podatkovnih baz NoSQL so: preprost podatkovni model, prilagodljivost je pomembnejša od strogega nadzora nad opredeljeno podatkovno strukturo, zahtevana visoka zmogljivost, stroga podatkovna celovitost ni nujna in računalništvo v oblaku.

Glede na rezultate lahko ugotovimo, da je podatkovna baza ključ-vrednost Redis primerna za uporabo v primerih, ko imamo večinoma opravka z branjem velikih količin podatkov (scenarij A). Podobno velja tudi za Cassandro, ki spada v množico podatkovnih baz vrste razširljivi zapis in dokumentno usmerjeno podatkovno bazo MongoDB. Hbase (baza vrste razširljivi zapis) lahko priporočimo v primeru, ko imamo opravka z velikim številom pisanj podatkov v podatkovno bazo (scenarija A in C).

Relacijska baza MySQL se je odrezala presenetljivo dobro pri branju podatkov (scenarija B in Č), slabše pa pri pisanju, kar je lepo razvidno iz scenarija C.

Iz tega lahko povzamemo, da torej ne gre za vprašanje, ali so podatkovne baze NoSQL boljše od relacijskih, ampak predvsem za vprašanje, kdaj je smiselno uporabiti baze vrste NoSQL. Odgovor je, kadar se soočamo z zahtevami za poizvedbe po obsežnih podatkovnih naborih z veliko hitrostjo izvajanja poizvedb. Pri tem nastopijo baze NoSQL. Gre za podatkovne nabore, med katere 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 relacijska podatkovna baza ni dovolj zmogljiva in hitra rešitev.

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

6 VIRI IN LITERATURA

[1] Chang, F., Dean, J., Ghemawat, S., Hsieh, W. C., Wallach, D. A., Burrows, M., Chandra, T., Fikes, A., Gruber, R. E. (2006). Bigtable: A Distributed Storage System for Structured Data, OSDI 2006.

[2] Github (2012). Yahoo! Cloud Serving Benchmark (YCSB), Dostopno 10. 1. 2012 na https://github.com/brianfrankcooper/YCSB/wiki.

[3] Knez, N. (2012). Analiza podatkovnih baz NoSQL in izdelava odločitvenega modela za izbiro med relacijskimi in NoSQL podatkovnimi bazami. Diplomsko delo. Univerza v Ljubljani, Fakulteta za računalništvo in informatiko.

[4] Leavitt, N. (2010). Will NoSQL Databases Live Up to Their Promise?, IEEE Xplore digital library, Vol. 43, No. 2, pp. 12–14.

[5] Stonebraker, M. (2010). SQL databases vs. NoSQL databases, ACM Digital Library, Vol. 53, No. 4.

[6] Žlender, R. (2011). Primerjava zmogljivosti nerelacijskih podatkovnih baz. Diplomsko delo. Univerza v Ljubljani, Fakulteta za računalništvo in informatiko.
[7] Lavbič, D., Vasilecas, O., Rupnik, R. (2012). Ontology-based Multi-Agent System to support business users and management. Technological and Economic development of Economy, št. 16 (2), str. 327–347.
[8] Andersen, C., Lehnardt, J., Slater, N. (2010). CouchDB: The Definitive Guide, 1st ed., Sebastopol: O'Reilly Media, Inc., pogl. 2.
[9] Osborne, R. (2011). »Playing around with MongoDB and MapReduce functions« Dostopno 20. 12. 2011 na: http://rickosborne.org/blog/2010/02/playing-around-with-mongodb-and-mapreducefunctions/.
[10] Cattell, R. (2010). Relational Databases, Object Databases, Key-Value Stores, Document Stores, and Extensible Record Stores: A Comparison, Dostopno 16. 1. 2012 na: http://www.odbms.org/download/Cattell.Dec10.pdf.
[11] Terrastore (2011). Dostopno 15. 12. 2011 na: http://code.google.com/p/terrastore/.
[12] Amazon EBS (2012). Dostopno 30. 1. 2012 na: http://aws.amazon.com/ebs/.
[13] Amazon EC2 FAQs (2012). Dostopno 30. 1. 2012 na: http://aws.amazon.com/ec2/faqs/#What_is_an_EC2_Compute_Unit_and_why_did_you_introduce_it.

▪

Aljaž Zrnec je magistriral leta 2002 na Fakulteti za računalništvo in informatiko Univerze v Ljubljani. Leta 2006 je doktoriral s področja konstruiranja metodologij. Zaposlen je v Laboratoriju za podatkovne tehnologije kot asistent za področje podatkovnih baz. Na raziskovalnem področju se ukvarja s konstruiranjem metodologij, podatkovnimi bazami NoSQL in računalništvom v oblaku. Je avtor ali soavtor številnih prispevkov v strokovnih in znanstvenih publikacijah.

▪

Lovro Šubelj je mladi raziskovalec in asistent na Fakulteti za računalništvo in informatiko Univerze v Ljubljani. Raziskovalno se ukvarja z analizo realnih omrežij, natančneje z odkrivanjem skupin vozlišč v velikih kompleksnih omrežjih.

▪

Slavko Žitnik je doktorski študent na Fakulteti za računalništvo in informatiko Univerze v Ljubljani in je hkrati zaposlen kot mladi raziskovalec iz gospodarstva v podjetju Optilab, d. o. o. Raziskovalno se ukvarja predvsem s procesiranjem besedil, bolj natančno z razpoznavanjem entitet in povezav med njimi, z uporabo metod iz strojnega učenja in semantičnih tehnologij.

▪

Aleš Kumer je asistent na Fakulteti za računalništvo in informatiko Univerze v Ljubljani.

▪

Marko Bajec je izredni profesor na Fakulteti za računalništvo in informatiko Univerze v Ljubljani, kjer poučuje dodiplomske in podiplomske predmete s področja razvoja informacijskih sistemov in podatkovnih baz. Raziskovalno se ukvarja z metodami in pristopi k snovanju in razvoju informacijskih sistemov, obvladovanjem informatike ter v zadnjih letih predvsem s podatkovnimi tehnologijami za predstavitev, analizo in vizualizacijo podatkov. Leta 2009 je ustanovil Laboratorij za podatkovne tehnologije ter prevzel njegovo vodenje. Je član številnih domačih in tujih združenj, komisij in odborov. V okviru fakultete je vodil več aplikativnih in raziskovalnih projektov. Svoje raziskovalne rezultate in dosežke iz prakse redno objavlja v domačih in mednarodnih znanstvenih in strokovnih krogih.