Performanse Memcached API pristupa MySQL serveru

Jelena Jokić
Banjalučka berza hartija od vrijednosti a.d. Banja Luka
Banja Luka, RS, BiH
jelena.jokic@blberza.com

Mihajlo Savić, Slavko Marić
Elektrotehnički fakultet
Univerzitet u Banjoj Luci
Banja Luka, RS, BiH
m@etfbl.net, ms@etfbl.net

Sadržaj—**U radu su prikazane arhitekture dva načina keširanja pristupa MySQL serveru pomoću Memcached servisa – eksternog i internog. Izvršeno je testiranje jednonitnih performansi samostalnog Memcached-a, samostalnog MySQL-a, Memcached servisa eksternog MySQL servisu, te Memcached servisa integrisanog u MySQL servis.**

Ključne riječi-Memcached, MySQL, performanse, keširanje

I. UVOD

Poređenje performansi NoSQL i SQL baza podataka je izuzetno nezahvalan posao, naročito imajući u vidu suštinske razlike u načinu funkcionisanja kao i u problemima za čije su rješavanje projektovani navedeni sistemi. Jedan često prisutan vid upotrebe NoSQL baza je korištenje istih u funkciji sloja za keširanje upita, koji se nalazi između aplikacije i SQL baze podataka. Obično se u funkciji NoSQL servisa koriste jednostavni ključ-vrijednost sistemi (*key-value store*) čiji je najpopularniji predstavnik Memcached. S druge strane, makar kada se govori o *open-source* rješenjima, najpopularniji SQL server baza podataka je MySQL. Počevši od verzije 5.6 u sam MySQL server baze podataka ugrađena je i mogućnost direktnog Memcached API pristupa relacionim podacima smještenim u InnoDB tabele. Navedeno otvara mogućnost mjerenja performansi NoSQL i SQL baza podataka u ovom, koliko god ograničenom, toliko i realnom i često korištenom slučaju upotrebe.

II. RELACIONE BAZE PODATAKA I MYSQL

Kod relacionih baza podataka efikasno fizičko smještanje zapisa i praćenje veza između njih realizuje se u formi dvodimenzionalnih struktura – tabela koje odgovaraju matematičkim relacijama. Redovi u tabeli predstavljaju konkretne objekte u realnom svijetu, a kolone predstavljaju određena svojstva objekta. Relacioni koncept omogućava korištenje matematičkog aparata za rad sa relacijama.[1]

Svi pristupi bazi se realizuju korištenjem sistema za upravljanje bazama podataka koji olakšavaju programerima rješavanje nekih zajedničkih problema pri kreiranju aplikativnih sistema sa bazom. Uspjeh sistema za upravljanje bazama podataka je u jednostavnom upitnom jeziku koji omogućava lako pisanje kompleksnih izraza bez poznavanja fizičke organizacije podataka i naprednoj tehnologiji procesiranja upita koja transformiše upite visokog nivoa u efikasne upite niskog nivoa.[2]

Ugrađena je podrška za tri nivoa apstrakcije podataka: fizički, logički i nivo pogleda. Fizički nivo je najniži nivo apstrakcije koji opisuje kako su podaci zapravo smješteni u bazi koristeći kompleksne strukture podataka. Logički nivo je sljedeći, viši nivo apstrakcije koji opisuje koji podaci su smješteni u bazi i kakve veze postoje između njih. Dakle, logički nivo opisuje čitavu bazu putem jednostavnih struktura tako da korisnik ne primjećuje kompleksnost koja je sačuvana u pozadini. Najviši nivo apstrakcije predstavlja nivo pogleda koji je samo dio ukupne logičke strukture baze koji je potreban određenom korisniku.[1]

Takođe, sistem za upravljanje bazama podataka posjeduje modul za upravljanje konkurentnim izvršavanjem transakcija (*concurrency control manager*) koji omogućava uspješno izvršavanje konkurentnih transakcija. Transakcije predstavljaju skup operacija koje obezbjeđuju izvršavanje neke logičke cjeline u radu sa bazom podataka. Transakcije imaju svojstva koja su neophodna da bi se očuvali integritet i konzistentnost baze i poznata su kao ACID svojstva (*Atomicity, Consistency, Isolation, Durability*). [2]

Kada se govori o tradicionalnim klijent-server okruženjima, MySQL je najpopularniji *open-source* relacioni server baza podataka [3]. Jedna od specifičnosti MySQL-a je i ta da ne postoji samo jedan podsistem za pohranu i upravljanje podacima (*storage engine*) već se korisnicima nudi na raspolaganje zaista velik broj različitih podsistema koji su ravnopravni sa stanovišta upotrebe. Neki od njih su: MyISAM, InnoDB, IBMDB2I, MERGE, MEMORY, FEDERATED, ARCHIVE, CSV, BLACKHOLE, te NDB. Osim navedenih rješenja, po potrebi je moguće koristiti i neki od brojnih komercijalnih sistema od kojih svaki nudi određene prednosti i specifičnosti. Kako su svi podsistemi za pohranu i upravljanje podacima ravnopravni, nema prepreke miješanju tabela različitih vrsta unutar iste šeme što omogućava projektovanje sistema koji pokrivaju veliku većinu mogućih korisničkih zahtijeva. Počevši od verzije 5.5, InnoDB je podrazumijevani podsistem za pohranu i upravljanje podacima.

A. *InnoDB*

InnoDB sistem ima punu podršku za rad sa transakcijama i obezbjeđuje ACID svojstva. Transakcioni model sadrži commit, rollback i crash-recovery mogućnosti da bi obezbjedio zaštitu podataka. Korisiti se zaključavanje redova što omogućava istovremeno izvršavanje više upita nad različitim redovima iste tabele. MyISAM i memory storage engine koriste zaključavanje čitave tabele, dok BDB koristi zaključavanje na nivou strane, odnosno zaključava grupu redova. InnoDB takođe koristi nezaključavajuće čitanje što znači da, generalno upiti vide samo rezultate završenih transakcija, sa izuzetkom da upiti unutar jedne transakcije vide eventualne promjene koje su napravili prethodni iskazi iz iste transakcije.

Fizička organizacija je takva da su InnoDB tabele sortirane na osnovu primarnog ključa, čime se optimizuju upiti nad kolonama primarnog ključa. Ako primarni ključ ne postoji InnoDB će ga interno generisati. Uz svaki primarni ključ kreira se primarni indeks (*clustered index*). Primarni indeks kao ključeve uzima vrijednosti primarnih ključeva iz tabele, a vrijednosti čine ostale kolone referenciranog reda.

Svi ostali indeksi nazivaju se sekundarni indeksi (*secondary index*). Sekundarni indeksi kao vrijednost sadrže primarni ključ. Pored indeksa na primarnom ključu postoje i full text indeksi koji se koriste za full text pretrage. Ovi indeksi se fizički predstavljaju novom InnoDB tabelom. Indeksi i podaci se čuvaju keširani u baferu InnoDB podsistema, u glavnoj memoriji.

III. NoSQL baze podataka i Memcached

A. *Kratak pregled NoSQL baza podataka*

Iako je naziv NoSQL nastao kao složenica dvije riječi – ne i SQL, u praksi se ova skraćenica uglavnom čita kao Ne Samo SQL (*Not Only SQL*), što najbolje opisuje svrhu njihovog korištenja. Naime, ove baze podataka se ne koriste kao zamjena za klasične relacione baze podataka već kao njihova svojevrsna dopuna za brz rad sa podacima reda veličine nekoliko desetina gigabajta. NoSQL baze su posebno popularne u posljednje vrijeme, najviše zahvaljujući velikim kompanijama koje koriste nerelacione baze za čuvanje svojih podataka i time navode i ostale na upotrebu sličnih rješenja. Treba napomenuti da su ove baze generalno *open-source* projekti. [4][5]

Najveća novina koju NoSQL baze uvode je promjena modela podataka. NoSQL baze ne koriste relacioni model već različita rješenja nude različite modele i na osnovu toga se grupišu u 4 kategorije: kolonski orijentisane baze podataka (*wide column store*), baze podataka orijentisane ka dokumentima (*document store*), baze podataka tipa ključ-vrijednost (*key-value store*) i baze podataka orijentisane ka grafovima (*graph store*).

Baze podataka tipa ključ-vrijednost predstavljaju najjednostavniju vrstu NoSQL baza podataka. To su jednostavne heš tabele, sa glavnom upotrebom u slučajevima kada se svi pristupi bazi vrše preko primarnog ključa. Dakle, podaci se čuvaju u vidu para ključ-vrijednost. Vrijednost je uglavnom neka vrsta podatka primitivnog/standardnog tipa u programskim jezicima (string, cjelobrojni tip) ili referentnog tipa (objekat, niz). U nekim ključ-vrijednost bazama, kao što je Riak, vrijednosti mogu biti liste, skupovi i heš tabele; sa operacijama tipa razlike, unije, presjeka itd. [6]

B. *Memcached*

Memcached je *key-value store* koji čuva podatke u memoriji. Predstavlja *open source* sistem za keširanje objekata u distribuiranu memoriju. Tipična upotreba je za ubrzanje rada dinamičkih web aplikacija keširanjem podataka u RAM, čime se smanjuje broj pristupa osnovnoj bazi podataka. Koriste ga brojni sajtovi kao što su: YouTube, Facebook, Reddit, Twitter, Wikipedia, Flickr i mnogi drugi. [7]

Iako je sam Memcached izuzetno popularan proizvod, za slučaj koji se razmatra još je važnija činjenica da je API za pristup Memcached servisu postao *de-facto* standard za pristup *key-value store* sistemima.

Memcached je implementiran kao velika heš tabela distribuirana preko više servera. Serveri čuvaju vrijednosti u RAM-u i u slučaju nedostatka memorije koristi se algoritam koji zamjenjuje najmanje korištenu vrijednost (eng. *Least Recently Used* - LRU). Iz tog razloga se ne može sa sigurnošću očekivati da će se podaci uvijek nalaziti u heš tabeli. Još jedno od ograničenja ovog pristupa leži u činjenici da sam Memcached bez eksternih servisa ne nudi nikakvu zaštitu pristupa podacima a ni šifrovanje komunikacije.

IV. Memcached i MySQL

Kao što je prethodno navedeno, Memcached predstavlja često korišćeno rješenje za keširanje upita u sloju između aplikacije i MySQL servera baze podataka. Principijelno postoje dvije osnovne arhitekture sistema koji koristi Memcached i MySQL na ranije opisani način: samostalni Memcached i integrisani Memcached pristup [8].

A. *Samostalni Memcached server*

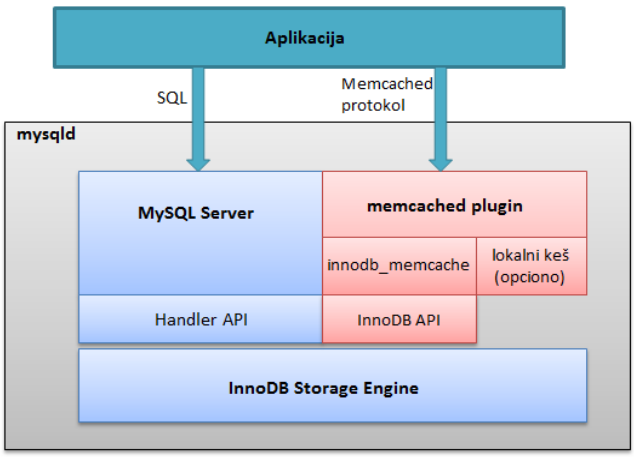
U ovom slučaju se, npr. postojeći, sistem zasnovan na MySQL-u proširuje dodatnim Memcached serverom (ili servisom) koji služi za keširanje rezultata SQL upita. Aplikacija pri upisu podataka u bazu, nakon uspješnog upisa odgovarajuće vrijednosti smiješta i u Memcached servis. Pri čitanju podataka se prvo pristupa Memcached servisu, a ukoliko podatak nije pronađen u kešu, isti se pribavlja iz MySQL servera i smiješta u keš za buduću upotrebu [8].

Očigledno je da postoji mnoštvo potencijalnih problema, od izbora podataka za keširanje, preko odabira perioda čuvanja podataka u kešu do problema izmjene podataka mimo aplikacije, odnosno keš sloja, što može uzrokovati keširanje podataka koji više nisu korektni. Navedeni problemi su generalni problemi vezani za keširanje i nisu specifični samo za opisani sistem, ali ih u analizi treba uzeti u obzir.

Sa stanovišta performansi, a imajući u vidu da su performanse samog Memcached-a mnogo više od MySQL-a, očigledno je da se može očekivati poboljšanje performansi čitanja osim u slučaju kada podaci nisu u kešu, dok je pri upisu podataka očekivan manji pad performansi.

B. *Integrisani Memcached pristup*

Ovaj pristup je moguć samo od MySQL-a verzije 5.6 u slučaju kada se koristi InnoDB podsistem za pohranu i upravljanje podacima. Na slici Sl. 1 je prikazana arhitektura sistema sa integrisanim Memcached servisom. [9]



Slika 1. MySQL 5.6 i integrisani Memcached plugin

Aplikacija direktno pristupa Memcached servisu integrisanom u MySQL server. Ukoliko aplikacija zahtijeva podatke koji trenutno nisu u kešu, MySQL će automatski pribaviti podatke i smjestiti ih u keš. Pri upisu podataka u keš od strane aplikacije, automatski se vrši ažuriranje podataka u tabelama, što ovaj način pristupa podacima čini ravnopravnim tradicionalnom SQL pristupu.

Postoje brojne prednosti korištenja Memcached interfejsa za InnoDB tabele: izbjegava se parsiranje SQL-a i optimizacija upita, izvršavanje *memcached* i *mysqld* procesa u istom procesnom prostoru izbjegava opterećenje mreže dodatnim zahtjevima, podaci mogu biti strukturirani ili ne u zavisnosti od potrebe, a i dalje je omogućen SQL pristup podacima za izvještavanja, analize i kompleksnije upite.

V. MJERENJE PERFORMANSI

A. *Aplikacija za mjerenje performansi i radno okruženje*

Za potrebe mjerenja performansi je realizovana aplikacija koja omogućava mjerenje vremena izvršavanja željenih operacija nad podacima na više različitih konfiguracija sistema.

Sama aplikacija je realizovana u programskom jeziku Java u verziji 1.7.0, uz upotrebu MySQL servera verzije 5.6.14-1, Memcached servera verzije 1.4.5-1, te biblioteka java_memcached_2.6.6 i mysql-connector-java-5.1.7 [10]. Sistem pod kojim su izvršena mjerenja je CentOS Linux 5.10 sa kernelom 2.6.18-348.6.1.el5 #1 SMP x86_64 dok je sistem sa hardverske strane sljedećih karateristika:

CPU: 2 x AMD Opteron 6128 (8x2GHz)

RAM: 8 x 2GB DDR3 1333 MHz

MB: AMD G34/SR5690/SP5100 - Supermicro H8DGU-F

HDD: WDC WD5002AALX-00J37A0

Realizovane su četiri varijante sistema:

S1. Memcached – samo Memcached server

S2. MySQL+Memcached – samostalni Memcached keš

S3. Memcached plugin – integrisani Memcached

S4. MySQL – samo MySQL bez dodatnog keširanja

Vrijedi napomenuti da u slučaju upotrebe samostalnog MySQL servera bez dodatnog keširanja i dalje postoji više slojeva keširanja, od keširanja pristupa disku do internog InnoDB keširanja. U cilju dobijanja realnih rezultata, podrazumijevana podešavanja sistema nisu mijenjana. U slučaju integrisanog Memcached plugin-a, sistem je podešen na takav način da je favorizovana pouzdanost sistema po cijenu gubitka performansi (*innodb-only* politika keširanja, COMMIT nakon svake transakcije, itd).

Kako bi izbjegli situaciju u kojoj se mjere samo performanse različitih slojeva keša, serveri su restartovani (brisanje keša) između svih pojedinačnih testova, osim naravno, Memcached servera u konfiguraciji S1.

I klijent (aplikacija) i server (Memcached/MySQL) se izvršavaju na istom serveru u cilju izbjegavanja uticaja mreže na mjerenja. Klijent sve upite izvršava sukcesivno u jednoj niti. U okviru ovog rada nije vršeno mjerenje performansi u višenitnom okruženju.

U cilju dobijanja realnih rezultata projektovan je sistem koji grubo aproksimira berzanski sistem trgovanja koji generiše relativno veliki broj operacija na relativno maloj količini podataka. Odabrana je veličina zapisa od 250 znakova uz mjerenje performansi na 10000 i 100000 operacija.

Pri upisu podataka aplikacija generiše 10000, odnosno 100000 unosa sekvencijalno. Pri čitanju podataka aplikacija pristupa slučajnim izborom 100000, odnosno 1000000 unosa (deset puta više od broja kreiranih unosa), dok se pri ažuriranju vrši izmjena 10000, odnosno 100000 podataka slučajnim izborom.

Omogućeno je mjerenje vremena izvršavanja umetanja, čitanja i izmjene podataka. Podacima je moguće pristupiti na četiri prethodno navedena načina. Pri čitanju navodi se broj umetnutih podataka i broj podataka koje treba pročitati. Moguće je čitati više podataka nego što postoji u bazi, odnosno neke podatke više puta, jer se podacima pristupa pseudo slučajnim izborom ključa iz opsega postojećih ključeva. Pri izmjeni postojećih podataka se takođe navodi broj postojećih podataka i broj podataka koje treba izmijeniti.

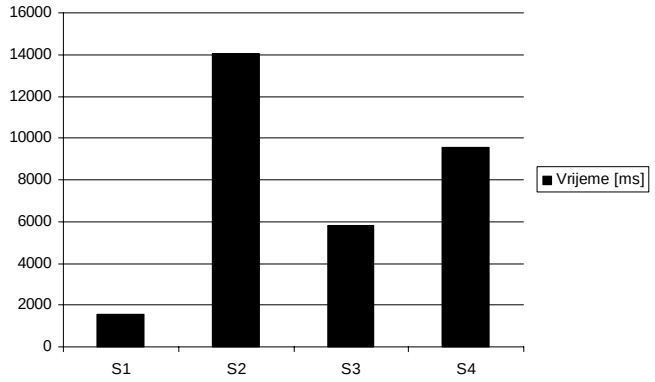
B. *Rezultati za 10000 unosa*

U tabeli “Tabela I” su dati rezultati mjerenja za 10000 upisa i izmjena te 100000 čitanja za sve konfiguracije. Svako mjerenje je izvršeno deset puta, a u tabeli su date srednje vrijednosti.

TABELA I. MJERENJA ZA 10000 UNOSA

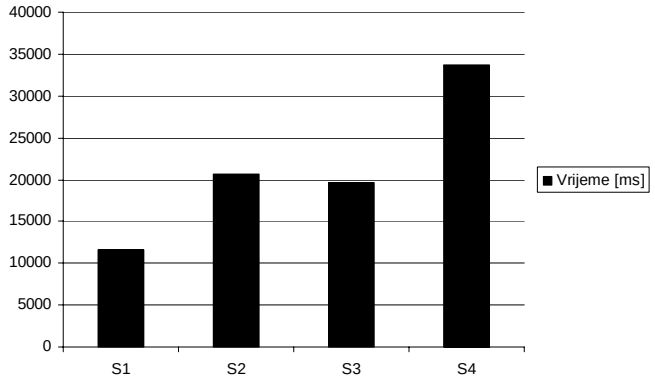
Vrijeme [ms]	***S1***	***S2***	***S3***	***S4***
INSERT	1559	14033	5817	9542
READ (10x)	11646	20616	19697	33750
UPDATE	1594	15752	7383	9483

Na slici Sl. 2 je dat grafički prikaz vremena upisa 10000 unosa.



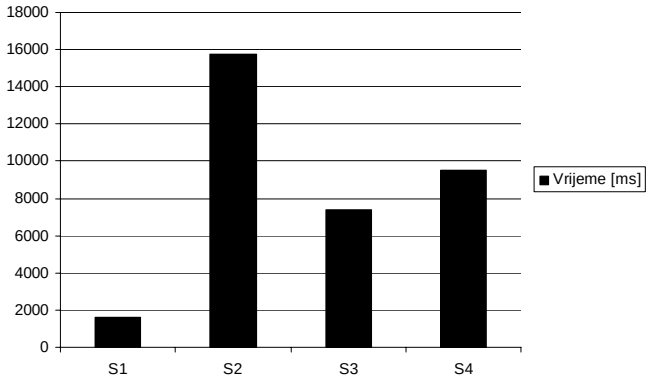
Slika 2. Vremena upisa 10000 unosa

Na slici Sl. 3 je dat grafički prikaz vremena čitanja 100000 unosa po slučajnom izboru.



Slika 3. Vremena čitanja 100000 unosa

Na slici Sl. 4 je dat grafički prikaz vremena izmjene 10000 unosa po slučajnom izboru.



Slika 4. Vremena izmjene 10000 unosa

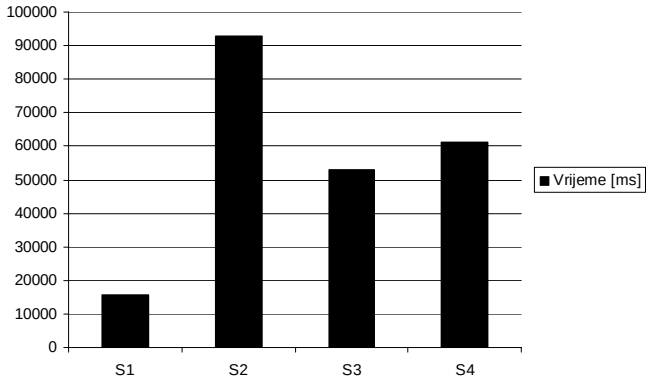
C. Rezultati za 100000 unosa

U tabeli "Tabela II" su dati rezultati mjerenja za 100000 upisa i izmjena te 1000000 čitanja za sve konfiguracije. Svako mjerenje je izvršeno deset puta, a u tabeli su date srednje vrijednosti.

TABELA II. MJERENJA ZA 10000 UNOSA

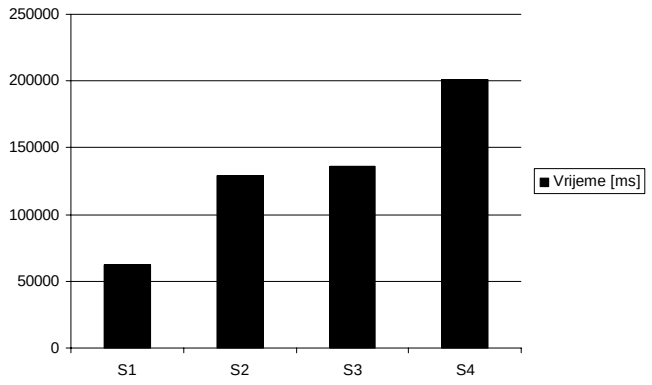
Vrijeme [ms]	***S1***	***S2***	***S3***	***S4***
INSERT	15505	92992	52970	61358
READ (10x)	62605	128776	136398	200876
UPDATE	14161	108434	60022	68491

Na slici Sl. 5 je dat grafički prikaz vremena upisa 100000 unosa.



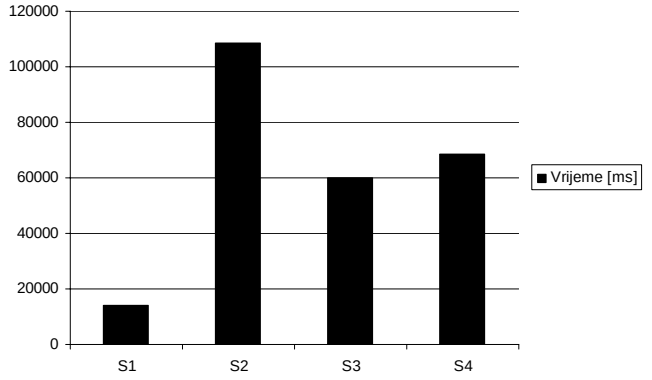
Slika 5. Vremena upisa 100000 unosa

Na slici Sl. 6 je dat grafički prikaz vremena čitanja 100000 unosa po slučajnom izboru.



Slika 6. Vremena čitanja 1000000 unosa

Na slici Sl. 7 je dat grafički prikaz vremena izmjene 100000 unosa po slučajnom izboru.



Slika 7. Vremena izmjene 100000 unosa

D. Diskusija rezultata

Iz rezultata prikazanih tabelarno i grafički u prethodnom tekstu vidljivo je da sam Memcached ima najbolje performanse za sve slučajeve upotrebe. Takvi rezultati su bili i očekivani imajući u vidu činjenicu da se podaci drže u RAM-u i da je mehanizam pristupa podacima izuzetno jednostavan.

Upotreba eksternog Memcached servisa uz postojeći MySQL server nudi osjetno bolje performanse za čitanje podataka, dok je kod upisa i ažuriranja podataka vrijeme potrebno za operacije veće nego kod samog MySQL servera. I ova mjerenja su očekivana jer se vrijeme troši i na rad sa Memcached servisom i sa MySQL serverom kod upisa i izmjena, dok se kod čitanja performanse gube samo dok je keš "hladan". Velika prednost ove konfiguracije je ta što ne samo da nema vezanosti za određenu verziju MySQL-a nego nema vezanosti za MySQL ili bilo koji drugi sistem za upravljanje bazama podataka. Naravno, ova fleksibilnost nosi sa sobom i skup specifičnih problema pri realizaciji sistema.

Izuzetno dobre rezultate ostvaruje integrisani Memcached plugin u okviru MySQL servera. Na osnovu mjerenja se može zaključiti da je ova konfiguracija ili osjetno brža ili zanemarivo sporija u odnosu i na sam MySQL i na MySQL sa eksternim Memcached servisom. Gubitaka performansi gotovo da i nema, a projektanti sistema se lišavaju brige o dodatnom sloju za keširanje[11]. Naravno, veliko ograničenje ove konfiguracije je neophodnost upotrebe MySQL servera u verziji 5.6 ili novijoj, te upotrebe InnoDB tabela za pohranu podataka. Ukoliko ova ograničenja ne predstavljaju problem, teško je pronaći razlog ne korišćenju integrisanog Memcached plugin-a.

VI. ZAKLJUČAK

Integracija Memcached API pristupa u MySQL serveru verzije 5.6 omogućava izvršavanje ravnopravnog poređenja i analize mjerenja dva pristupa poboljšanju performansi tradicionalnog SQL servera upotrebom jedne od NoSQL tehnologija. Iako su rezultati prema kojima jednostavna RAM zasnovana baza podataka ima najbolje performanse bili očekivani, osjetno smanjenje vremena izvršavanja svih testiranih vrsta upita, uz jednostavnost i transparentnost u radu, čini upotrebu integrisanog Memcached API-ja u MySQL serveru izuzetno kvalitetnim rješenjem za široku klasu problema realnih sistema.

LITERATURA

[1] A. Silberschatz, H. Korth, and S. Sudarshan, Database System Concepts, 4th ed. McGrawHill, 2012.

[2] S. Marić and D. Brđanin, Relacione baze podataka. Univerzitet u Banjoj Luci, Elektrotehnički fakultet, Banja Luka, 2012.

[3] "DB-Engines Ranking", http://db-engines.com/en/ranking, posjećeno: 25-Dec-2013.

[4] A. Lith and J. Mattsson, "Investigating storage solutions for large data-A comparison of well performing and scalable data storage solutions for real time extraction and batch insertion of data." Chalmers University of Technology, Sweden, 2010.

[5] C. Strauch, "NoSQL Databases." Stuttgart Media University, 2011.

[6] M. Fowler and P. J. Sadalage, NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence. Addison-Wesley, 2012.

[7] "About Memcached", http://memcached.org/about, posjećeno: 23-Nov-2013.

[8] "Using MySQL with memcached" , http://dev.mysql.com/doc/refman/5.6/en/ha-memcached.html, posjećeno: 25-Dec-2013.

[9] "Architecture of InnoDB and memcached", http://dev.mysql.com/doc/refman/5.6/en/innodb-memcached-intro.html, posjećeno: 25-Nov-2013.

[10] "Using MySQL and memcached with Java", http://dev.mysql.com/doc/refman/5.1/en/ha-memcached-interfaces-java.html, posjećeno: 23-Nov-2013.

[11] "Benefits of the InnoDB / memcached Combination", https://dev.mysql.com/doc/refman/5.6/en/innodb-memcached-benefits.html, posjećeno: 25-Dec-2013.

ABSTRACT

In this paper we present a comparison of two architectures for caching access to MySQL DBMS by using Memcached service. We have performed single-thread performance measurements of standalone Memcached, standalone MySQL, MySQL cached by external Memcached and Memcached integrated in MySQL server.

PERFORMANCE OF MEMCACHED API ACCESS TO MYSQL SERVER

Jelena Jokić, Mihajlo Savić, Slavko Marić