Datastore MongoDB Stub pro Google App Engine SDK

Stanislav HELLER[1], Tomáš VOLF[2]

[1]*Fakulta informačních technologií VUT v Brně*
Božetěchova 1/2, 612 66 Brno
xhelle03@stud.fit.vutbr.cz
[2]*Ústav informačních systémů, FIT VUT v Brně*
Božetěchova 1/2, 612 66 Brno
ivolf@fit.vutbr.cz

Abstrakt. Datastore MongoDB Stub pro Google App Engine SDK umožňuje vývojářům využívat v prostředí Google App Engine lokálního úložiště v podobě databáze MongoDB pro vývoj a testování jejich aplikací. Datastore MongoDB Stub simuluje produkční prostředí Google App Engine, a to konkrétně chování úložiště High Replication Datastore. Samotné úložiště High Replication Datastore není uvolněno pro použití v lokálním vývojovém prostředí a je dostupné pouze v produkčním prostředí.

Klíčová slova: Google App Engine, MongoDB, datastore, stub, cloud, NoSQL.

1 Úvod

V posledních letech se vývoj aplikací stále více orientuje do prostředí cloudu, kde aplikace běží na distribuovaných výpočetních zdrojích, které jsou ve většině případů vlastněny a spravovány třetími stranami.

Jedním z těchto cloudových prostředí je Google App Engine[1], kterému se věnuje tento článek. Ačkoliv mají vývojáři možnost vyvíjet své aplikace lokálně, samotné úložiště, kam Google App Engine v produkčním prostředí data ukládá, poskytováno není. Pro vývoj a testování je možné využít jiných úložišť, která kromě ukládání vlastních dat také simulují chování úložiště Google. Tento článek představuje možnosti využití databáze MongoDB právě pro účely lokálního úložiště při vývoji a testování aplikací v prostředí Google App Engine.

Google App Engine není jediným produktem na trhu cloudových prostředí.

Společnost Microsoft nabízí produkt Windows Azure[2], který je primárně zaměřen pro využití s technologií .NET, ale poskytuje také prostředí pro PHP a Javu. Windows Azure je provozováno na operačním systému Windows, pro úložiště Microsoft nabízí Azure Table Storage nebo vlastní relační databázi Azure SQL Server.

Amazon nabízí produkt Amazon EC2[3] spadající do platformy Amazon Web Services. Jako úložiště lze zvolit SimpleDB nebo DynamoDB, případně pomocí Amazon Relation Database Service využít relačních databází MySQL, Oracle nebo Microsoft SQL Server.

[1] https://cloud.google.com/products/
[2] http://www.windowsazure.com/
[3] http://aws.amazon.com/ec2/

P. Šaloun (ed.), DATAKON 2013, Ostrava, 13-15.10.2013, pp. 1-10.

Společnost RedHat nabízí od roku 2011 produkt OpenShift[4]. OpenShift je otevřená platforma, veškeré služby lze použít i mimo cloud. Platforma nabízí široké množství programovacích jazyků (Java, Perl, PHP, Python, Ruby) a aplikační služby jako např. MongoDB, Mysql nebo PostgreSQL.

2 Google App Engine

Google App Engine [10] je platforma pro vývoj a hostování webových aplikací založená na jazyku Python. App Engine zjednodušuje vývoj cloudových aplikací – je navržen tak, aby mohl bez problému obsloužit mnoho současně probíhajících požadavků. Samotné aplikace se nemusí starat o zdroje, správu zdrojů zajišťuje právě App Engine, který v případě většího zatížení automaticky alokuje pro danou aplikaci další zdroje.

Základní prostor je vývojářům poskytnut zdarma do určité kvóty využití zdrojů, které zahrnují využití CPU, úložiště, příchozího a odchozího síťového provozu a dalších zdrojů. Pro malé projekty jsou tyto kvóty dostatečné, Google uvádí, že by měly postačit přibližně na 5 milionů shlédnutých stránek měsíčně. Pokud tento prostor vývojářům nestačí, mohou si zaplatit využití dalších zdrojů nad poskytnutou volnou kvótu zdrojů.

Google App Engine je orientován převážně na NoSQL databáze. Google Datastore je obecné označení nerelační databáze v Google App Engine. Dnes je nejčastěji využívána implementace High Replication Datastore (HRD) [1]. Termín Google Datastore bude dále využit výhradně pro označení databáze HRD. Samotné úložiště HRD však není uvolněno pro nasazení na lokální server; pro vývoj a testování proto slouží Software Development Kit (SDK), který simuluje produkční prostředí pro běh aplikace. V SDK se nacházejí rozhraní, která napodobují chování služeb App Enginu, tzv. stuby. SDK obsahuje také stuby pro ukládání dat, tzv. Datastore Service Stuby, které umožňují používat lokální úložiště (např. SQLite) a zároveň simulují chování HRD.

2.1 Datastore Service Stuby

Datastore File Stub

Datastore File Stub byl první implementací datastore stubu. Jeho název je poněkud zavádějící, protože všechna data ukládá do paměti RAM. File Stub lze přenastavit i tak, aby data kromě uložení do paměti RAM současně kopíroval na pevný disk do souboru.

Při malém objemu dat je ukládání dat do paměti RAM výhodou díky vysoké rychlosti zápisu a dotazování dat. Další výhodou File Stubu je fakt, že pro provoz nepotřebuje žádné externí knihovny či databáze.

Nevýhodou File Stubu je nutnost načtení obsahu databáze ze souboru do paměti při každém spuštění stubu. Navíc v paměti RAM ani v databázovém souboru nejsou podporovány indexy, proto je v případě filtrování nutné provést sekvenční prohledávání dat. Při větších objemech dat je výkon File Stubu nízký.

Datastore SQLite Stub

V roce 2010 byla zveřejněna implementace dalšího datastore stubu, kterým je Datastore SQLite Stub[5]. Jak již z názvu stubu vyplývá, jako úložiště je v tomto případě použita databáze SQLite. SQLite řeší problém s množstvím dat v paměti RAM a také rychleji

[4] https://www.openshift.com/

[5] https://sites.google.com/site/sqlitedatastorestub/

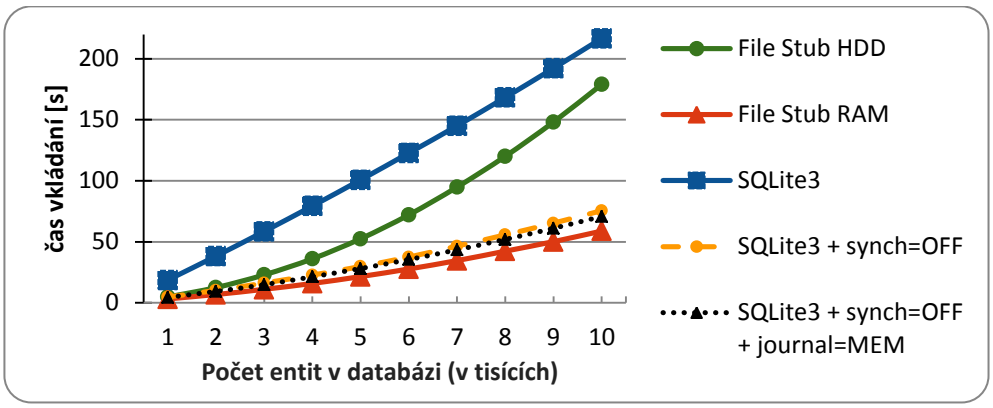
vyhledává v rozsáhlejší databázi. Pro malý objem dat je Datastore File Stub stále rychlejší, a to jak při čtení, tak i při zápisu do databáze.

Velkou nevýhodou Datastore SQLite Stubu je hromadné vkládání či upravování dat, protože vložení či úprava všech dat není obalena pouze jedinou transakcí, ale každá operace INSERT či UPDATE je obalena svou vlastní transakcí.

Srovnání Datastore Service Stubů

Pro porovnání obou stubů byl proveden test jejich zápisové rychlosti, který je zobrazen v grafu na obrázku 1. Test vkládal do jednotlivých úložišť vždy 10000 entit o průměrné velikosti 100 Bytů a byl proveden na stroji s následujícími parametry: 1,2 GHz CPU Intel Celeron, 16GB RAM, 80GB SSD disk, OS Linux – Ubuntu 12.04 LTS. Pro účely porovnání byly uvažovány následující nastavení datastore stubů (v závorce je uveden název datové řady v grafu na obrázku 1):

- Datastore File Stub s ukládáním dat na disk (File Stub HDD)
- Datastore File Stub s ukládáním dat pouze do paměti RAM (File Stub RAM)
- Datastore SQLite Stub s implicitním nastavením (SQLite3)
- Datastore SQLite Stub s vypnutou okamžitou synchronizací transakčních dat na disk (SQLite3 synch=OFF)
- Datastore SQLite Stub s vypnutou okamžitou synchronizací transakčních dat na disk a s ukládáním žurnálu pouze do paměti RAM (SQLite3 synch=OFF + journal=MEM)



Obr.1: Srovnání zápisové rychlosti datastore stubů s úpravou jejich nastavení

3 NoSQL databáze

NoSQL jsou takové databázové systémy, které nejsou relační. Ačkoliv NoSQL databáze vznikaly již v šedesátých letech dvacátého století, teprve nyní se začínají více prosazovat na poli databázových systémů. S problematikou NoSQL databází je úzce spjat problém CAP teorému.

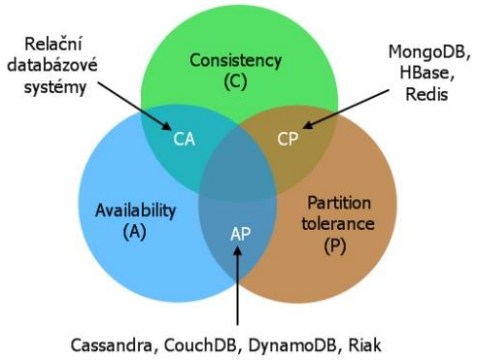
3.1 CAP teorém

CAP teorém zavedl a představil v roce 2000 Eric A. Brewer. Název teorému vychází z počátečních písmen tří vlastností, které v tomto teorému figurují [9]:

- **Konzistence *(consistency)*** zaručuje, že kdykoliv někdo přistupuje k databázi a čte z ní nějaká data, vždy bude číst nejaktuálnější data včetně těch, která byla změněna těsně před okamžikem čtení.
- **Dostupnost *(availability)*** znamená, že databáze pracuje bez výpadků a je vždy dostupná. Dostupnost se zpravidla řeší nasazením více serverů a databází se sdílenými daty, které se mezi sebou replikují. Toto distribuované úložiště pak navenek působí jako jediná databáze.
- **Tolerance k rozdělení *(partition tolerance)*** zajišťuje, aby i přes nedostupnost některé části databáze mohla být data stále čtena i zapisována a nedošlo tak ke ztrátě dat. Pokud je nějaká část databáze nedostupná, pak jsou data určená pro nedostupnou část zapsána do jiné, dostupné, části databáze. Poté, co je dříve nedostupná část databáze znovu zprovozněna, jsou jí doručena a jsou v ní zapsána data určená pro ni v době výpadku.

CAP teorém říká, že v žádném systému, kde jsou nějakým způsobem sdílena data, nelze zaručit, aby byly současně splněny všechny tři vlastnosti; vždy lze zaručit maximálně dvě z těchto vlastností [3]. Jak je vidět také na obrázku 2, z kombinací těchto dvou vlastností vznikají systémy s CA, AP a CP přístupy [6] při práci s daty:

- CA systémy využívají replikace pro zajištění dostupnosti konzistentních dat. Do kategorie CA systémů patří většina relačních databázových systémů.
- AP systémy upřednostňují dostupnost a toleranci k rozdělení před konzistencí. V těchto systémech se uplatňuje tzv. konečná konzistence *(eventual consistency)*, což znamená, že data budou v konečném čase konzistentní. Do kategorie AP systémů spadají například Cassandra, CouchDB, Dynamo či Riak.
- CP systémy upřednostňují konzistenci a toleranci k rozdělení jednotlivých uzlů distribuovaného úložiště před dostupností úložiště jako celku. Do této kategorie můžeme zařadit například HBase, MongoDB nebo Redis.



Obr.2: Vlastnosti CAP teorému a jejich kombinace – vytvořeno podle [2]

Seth Gilbert a Nancy Lynch v roce 2002 představili formální důkaz Brewerova CAP teorému [4].

3.2 Typy NoSQL databází

V závislosti na použitém datovém modelu se můžeme setkat s následujícími typy [4] NoSQL databází:

- **Úložiště typu klíč-hodnota *(key-value stores)*** ukládají do databáze hodnoty, které indexují pomocí zadaného klíče. Úložiště tohoto typu nepodporují žádné další, sekundární, klíče; hodnotu lze získat nebo změnit pouze pomocí tohoto zadaného klíče.
- **Sloupcově orientované úložiště *(column stores, extensible record stores)*** obsahují pro každý atribut vlastní rozšiřitelný sloupec, který sdružuje podobná data. Výhodou tohoto přístupu je efektivnější komprimace dat. Jednotlivé sloupce lze jednoduše rozprostřít mezi jednotlivé uzly distribuovaného úložiště. Distribuovat mezi jednotlivé uzly můžeme nejen sloupce, ale také bloky jednotlivých sloupců.
- **Dokumentově orientované úložiště *(document stores)*** umožňují ukládat různé typy dokumentů v databázi. Navíc podporují vnořené dokumenty a seznamy. Oproti úložištím typu klíč-hodnota podporují také sekundární indexy.

3.3 Výhody NoSQL databází

Hlavní výhodou využití NoSQL databází je jejich schopnost zpracovávat a ukládat nestrukturovaná data. NoSQL databáze jsou obecně rychlejší, protože je jejich datový model jednoduší než v případě relačních databází a protože nepodporují vlastnosti ACID (atomicita, konzistence, izolace a trvalost), které s sebou přináší režii navíc [8].

NoSQL databáze často slouží jako distribuovaná úložiště, což s sebou přináší další výhody, jako například škálovatelnost, (alespoň částečnou) dostupnost, snadné rozšíření, podporu velkokapacitních úložišť a další.

3.4 Nevýhody NoSQL databází

NoSQL databáze mají také své nevýhody. Díky přítomnosti nestrukturovaných dat není možné vytvořit obecný a univerzální dotazovací jazyk podobný jazyku SQL. NoSQL databáze neposkytují takovou spolehlivost jako databáze relační, protože nativně nepodporují vlastnosti ACID.

4 MongoDB

MongoDB[6] je platformově nezávislá dokumentová NoSQL databáze s otevřeným zdrojovým kódem vyvíjená společností 10gen od roku 2007. Z pohledu CAP teorému spadá MongoDB do kategorie CP systémů, upřednostňuje tedy konzistenci a toleranci k rozdělení nad dostupností databáze jako celku.

MongoDB [4] používá pro ukládání dat binární podobu formátu JSON označovanou BSON. Formát BSON podporuje pravdivostní hodnoty, číselné, řetězcové, binární typy a typy pro datum. V MongoDB lze ukládat i tzv. vnořené záznamy, kdy je do daného záznamu vložen jiný strukturovaný záznam. Vnořování záznamů je však omezeno maximální velikostí záznamu. Data dokumentu se kódují do formátu BSON již na klientovi. Systém dále podporuje specifikaci GridFS pro ukládání rozsáhlých binárních objektů, jako jsou obrázky a videa.

Databáze MongoDB využívá automaticky indexy, které jsou definovány nad jednotlivými dotazovanými položkami. Systém podporuje algoritmus MapReduce pro složité souhrny nad dokumenty. MongoDB nepodporuje transakční zpracování.

[6] http://www.mongodb.org

4.1 Rozdíly mezi Google Datastore a databází MongoDB

Ačkoliv jsou Google Datastore a MongoDB nerelačními databázemi, jsou mezi nimi velké rozdíly, na které je třeba vhodně reagovat při návrhu způsobu ukládání dat. Tam, kde není možné simulovat správné chování na úrovni databáze, je třeba simulaci správného chování zajistit na aplikační vrstvě. Z těch nejzásadnějších rozdílů můžeme zmínit následující:

- Google Datastore je objektová databáze, která ukládá serializované objekty entit do databáze BigTable (sloupcově orientovaná databáze [7]) pod klíč objektu; není tedy možné provést aktualizaci pouze určité části záznamu.
- MongoDB používá sadu datových typů, která vychází z jazyka JSON a kterou obohacuje o binární data, ObjectID, datum atp. Tato sada je z nemalé části odlišná od souboru typů v Google Datastore, který navíc podporuje geografické body, záznamy o uživatelích a další. Na druhou stranu Datastore neumí pracovat např. s vnořenými strukturovanými záznamy.
- MongoDB nepoužívá hierarchický klíč – neexistuje nativní způsob ukládání stromových struktur pomocí klíčů.
- MongoDB poskytuje možnost sekundárních indexů. Datastore je simuluje na úrovni BigTable (vytváří novou tabulku pro index).
- MongoDB neumožňuje indexování dvou seznamů současně – tzn., že není možné vytvořit složený index nad dvěma atributy, jejichž hodnoty jsou pole. Google Datastore tvorbu tohoto indexu umožňuje, ale označí jej za tzv. exploding index.
- Chování databázové projekce v Google Datastore se kvůli použití indexů zásadně liší od chování MongoDB, které v tomto případě kopíruje vlastnosti SQL databází. Největší rozdíl lze nalézt u projekce atributu, který obsahuje seznam hodnot – zde Google Datastore navrací zvláštní částečné entity obsahující vždy pouze jeden prvek seznamu, zatímco MongoDB celý seznam.
- MongoDB nepodporuje transakční zpracování, naopak Google Datastore transakční zpracování podporuje.

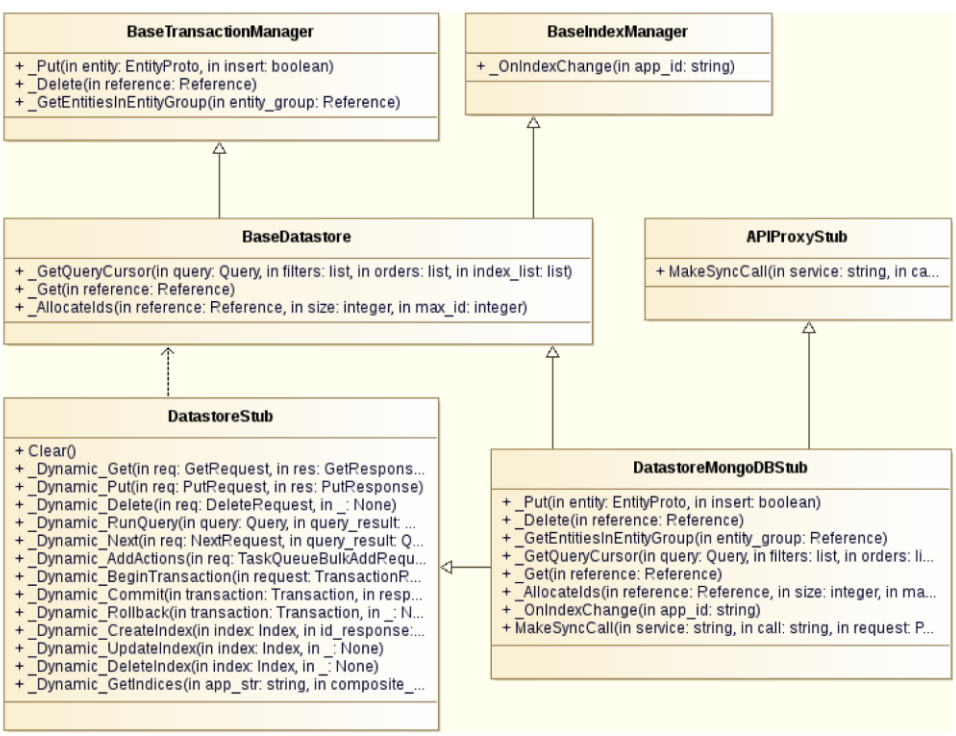
5 Datastore MongoDB Stub pro Google App Engine SDK

Základem každého service stubu je implementace metody `MakeSyncCall` (třídy `APIProxyStub`), která zajišťuje komunikaci všech knihoven vyšších vrstev s vlastním stubem.

Datastore MongoDB Stub[7] je navržen tak, aby mohl plně zastupovat stávající implementace v SDK. Navržený stub implementuje rozhraní třídy `BaseDatastore`, která zprostředkovává transakční vrstvu včetně simulace konzistence v konečném čase, a dědí všechny metody stubu od třídy `DatastoreStub`, která příchozí požadavky validuje a dále deleguje na nižší vrstvu, tedy právě na `BaseDatastore`. Tento přístup však s sebou přináší nutnost implementace speciálních kurzorů a dalších struktur pro simulaci chování Google Datastore.

Vazby Datastore MongoDB Stubu pro Google App Engine na již zmíněné třídy a celkový návrh architektury stubu jsou znázorněny ve zjednodušeném diagramu tříd na obrázku 3; pro přehlednost byly parametry některých metod zkráceny.

[7] https://github.com/hellerstanislav/appengine-datastore-mongodb-stub



Obr.3: Zjednodušený diagram tříd Datastore MongoDB Stubu

5.1 Mapování Google Datastore na MongoDB

V kapitole 4.1 byly popsány nejzásadnější rozdíly mezi databází MongoDB a Google Datastore, nyní stručně popíšeme způsoby řešení některých odlišností mezi těmito úložišti:

- **Reprezentace entit:** Entity jsou mapovány na MongoDB dokument. Každá entitní sada je uložena v kolekci s odpovídajícím jménem.
- **Klíč entity:** V Datastore je klíč hierarchický. Entita je ukládána do entitní sady, kam patří nejvyšší prvek v této hierarchii. Klíč entity je reprezentován jako seznam hierarchicky uspořádaných klíčů (od nejvyššího v hierarchii).
- **Převod datových typů:** Pro uložení hodnot datových typů Datastore, které MongoDB nepodporuje nativně, byl navržen strukturovaný formát hodnoty, kde je kromě samotné hodnoty uchovávána také informace o původním datovém typu hodnoty (aby byl možný zpětný překlad).
- **Neindexovatelné datové typy:** Google Datastore rozlišuje neindexovatelné datové typy, jsou jimi rozsáhlá textová data, binární data a vnořená entita s neindexovatelným obsahem. Každý dotaz na MongoDB obsahuje implicitní filtr, který zaručuje, že se ve výsledcích dotazů nebudou vyskytovat žádné hodnoty těchto datových typů.
- **Projekce** nad atributy se seznamem hodnot je simulována na aplikační vrstvě.
- **Transakce** jsou zajištěny na aplikační vrstvě v SDK. O simulaci transakcí se stará třída `BaseTransactionManager`, jehož funkcionalitu třída `DatastoreMongoDBStub` dědí.

– **Složený index nad seznamy** v tuto chvíli není podporován.

5.2 Experimenty s výkonností

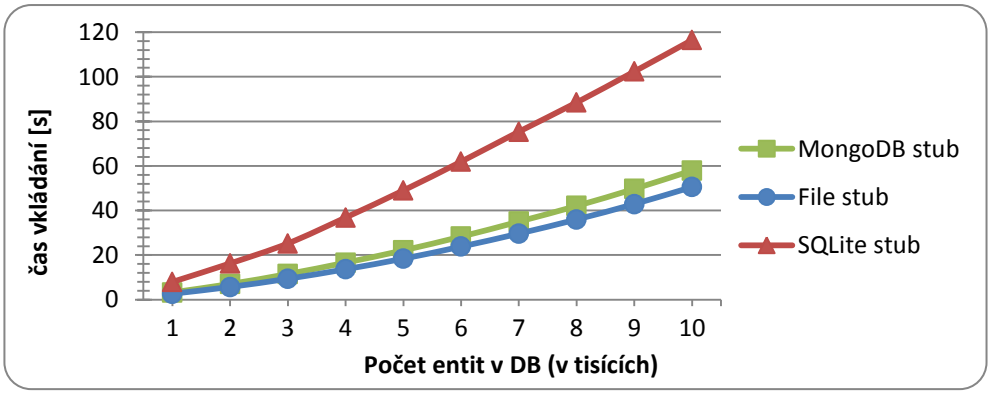
Vytvořený MongoDB Stub byl podroben srovnávacím testům rychlosti s File Stubem i SQLite Stubem. Pro účel testování byla využita entita s jedním řetězcem o velikosti 100 tisknutelných znaků, náhodným číslem datového typu integer, aktuálním datem a vnořenou entitou obsahující jedno náhodné číslo datového typu float.

Testy probíhaly nad náhodně vygenerovanými datovými sadami o velikosti 1000 až 10000 entit (po tisících) v deseti na sobě nezávislých sériích, které byly aritmeticky zprůměrovány. Testování proběhlo s vyprázdněnou pamětí RAM u MongoDB, a s optimalizovanými stuby pro co nejvyšší výkon (SQLite: zákaz synchronizací transakcí na disk a bez žurnálování; File Stub: ukládání dat jen do paměti RAM).

Všechny testy byly vykonány na stroji s konfigurací: 2,4 Ghz CPU Intel Pentium B980 (64bit), 4GB RAM DDR3 1600 MHz, 500 GB disk s rychlostí 7200 otáček za minutu, OS Fedora 17 (64bit), MongoDB 2.2.3 (64bit), Python 2.7.3 (64bit) a pymongo 2.4.

Test zápisu dat

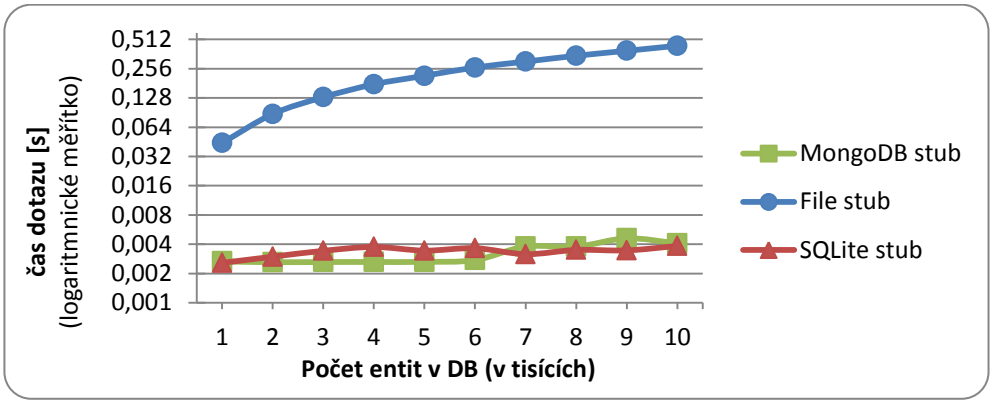
V tomto testu probíhalo vkládání dat do MongoDB bez čekání na zprávu o případné chybě a bez čekání na zápis do žurnálu. Graf na obrázku 4 ukazuje srovnání rychlosti vkládání dat pro jednotlivé stuby. Z grafu je patrné, že ukládání do MongoDB je jen o něco málo pomalejší než ukládání File Stubem do paměti RAM.



Obr.4: Srovnání zápisové rychlosti MongoDB Stubu s existujícími stuby

Test filtrování

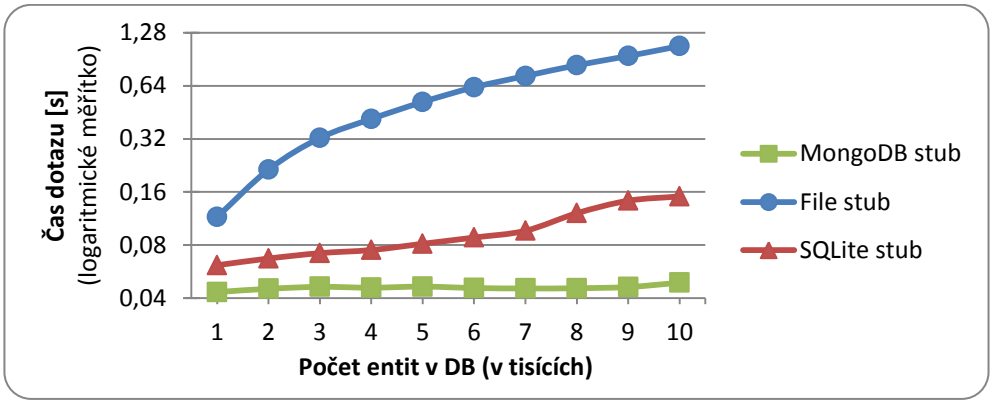
Test filtrování byl prováděn v každé sérii stokrát, výsledek byl aritmeticky zprůměrován. Graf na obrázku 5 ukazuje srovnání rychlosti filtrování entit, jejichž hodnota datového typu integer je menší nebo rovna hodnotě 100, přičemž maximální počet vybraných entit byl omezen na 100. Měřítko časové osy je logaritmické, protože rozdíly mezi výsledky MongoDB a SQLite Stubu jsou oproti File Stubu velmi malé. Z výsledků v grafu je vidět, že MongoDB Stub je srovnatelný s SQLite Stubem, což je velmi dobrý výsledek.



Obr.5: Srovnání rychlosti filtrování MongoDB Stubu s existujícími stuby

Test projekce

Pro tento typ testu byla vybrána projekce na atribut vnořené entity s číslem datového typu float. Tímto způsobem je testována schopnost stubu dotázat se jen na vybraný atribut entity a zároveň i rychlost dotazu na vnořenou entitu. Test byl rovněž omezen na 100 projekcí vybraného atributu. Z grafu na obrázku 6 je zřejmé, že tento typ testu byl nejúspěšnější právě pro MongoDB Stub. Měřítko časové osy grafu je opět logaritmické kvůli malým rozdílům ve výsledcích MongoDB a SQLite Stubu oproti File Stubu.



Obr.6: Srovnání rychlosti databázové projekce MongoDB Stubu s existujícími stuby

6 Závěr

V tomto článku byl prezentován Datastore MongoDB Stub pro Google App Engine SDK. Dále byla prezentována porovnání s ostatními existujícími stuby, jimiž jsou Datastore File Stub a Datastore SQLite Stub. V současné chvíli je MongoDB stub přizpůsoben pro 4 nejnovější verze SDK, konkrétně verze 1.7.4 až 1.7.7, a je plně integrovatelný do nového vývojového serveru devappserver2; současně je možné MongoDB stub využít také ve starém vývojovém serveru old_dev_appserver, do kterého je rovněž plně integrovatelný.

Literatura

1. Google App Engine: Python: Storing Data. Google Developers (verze 6. června 2013). Online: https://developers.google.com/appengine/docs/python/datastore/ [citováno 7. června 2013]. (anglicky)
2. MongoDB tutorials: Introduction to NoSQL. *W3resource.com*. Online: http://www.w3resource.com/mongodb/nosql.php [citováno 6. června 2013]. (anglicky)
3. Brewer, E. A.: Towards robust distributed systems. *The Nineteenth Annual ACM Symposium on Principles of Distributed Computing (PODC '00)*. Portland (2000), Oregon. Online: http://www.cs.berkeley.edu/~brewer/cs262b-2004/PODC-keynote.pdf [citováno 5. června 2013]. (anglicky)
4. Cattell, R.: Scalable SQL and NoSQL data stores. *ACM SIGMOD Rec*. 39, 4 (May 2011), 12-27. (anglicky)
5. Gilbert, S., Lynch, N.: Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web Services. *ACM SIGACT News* 33, 2 (June 2002), 51-59. (anglicky)
6. Han, J., et al.: Survey on NoSQL Database. In: *Proc. of the 6th International Conference on Pervasive Computing and Applications (ICPCA)*. IEEE Press, Piscataway, NJ (2011), 363-366. (anglicky)
7. Chang, F., et al.: Bigtable: A Distributed Storage System for Structured Data. *ACM Transactions on Computer Systems (TOCS)* 26, 2, Article 4 (June 2008), 4:1-4:26. (anglicky)
8. Leavitt, N.: Will NoSQL Databases Live Up to Their Promise? *Computer* 43, 2 (Februar 2010), 12-14. (anglicky)
9. Pokorný, J.: NoSQL Databases: A Step to Database Scalability in Web Environment. In: *Proc. of the 13th International Conference on Information Integration and Web-based Applications and Services (iiWAS '11)*. ACM, New York (2011), USA, 278-283. (anglicky)
10. Sanderson, D.: *Programming Google App Engine: Build and Run Scalable Web Apps on Google's Infrastructure*. O'Reilly Media, Inc., 1st edition. Sebastopol, California, USA, 2009. (anglicky)

Poděkování:
Tento článek byl podpořen granty MSM0021630528 a FIT-S-11-2.

Annotation:

Datastore MongoDB Stub for Google App Engine SDK

This paper deals with Google App Engine, which is a cloud platform for development and hosting web applications on Google infrastructure. It also discusses NoSQL databases, especially MongoDB database. High Replication Datastore, a production datastore for Google App Engine, is not available in local environment. The solution is the use of so-called datastore stubs, which store data to some available local datastore and simulate real production datastore. In Google App Engine, two datastore stubs are released: Datastore File Stub and Datastore SQLite Stub. In this paper Datastore MongoDB Stub is introduced, which is a new possibility to use with Google App Engine in development on local environment. Results of several experiments comparing performance of Datastore File Stub and Datastore SQLite Stub to the new Datastore MongoDB Stub are presenterd too.