Distribuovaný databázový systém na MySQL

Peter Schmidt[1], Pavol Jurík[2]

Abstrakt
V tomto príspevku sa zameriame na problémy spojené s distribuovanými databázami spoločnosti, ktorá prevádzkuje sieť obchodov a svoju infraštruktúru má postavenú na open-source technológiách. Svoje údaje uchováva a spracováva pomocou databázy MySQL. Jej cieľom je vytvoriť funkčný a spoľahlivý IS so všetkými výhodami distribuovaného databázového systému (DDBS) na platforme MySQL.

Kľúčové slová
MySQL, distribuovaná databáza, replikácia, distribuovaný IS

Abstract
In this paper, we focus on problems associated with distributed databases of a company that operates a network of stores and has built its infrastructure on open-source technologies. It stores and processes its data using a MySQL database. Its goal is to create a functional and reliable IS with all the advantages of a distributed database system (DDBS) on the MySQL platform.

Key words
MySQL, distributed database, replication, distributed IS

JEL classification
L8

1 Úvod

Globalizácia je hlavným trendom začiatku 21. storočia, čomu výrazne napomáhajú informačné technológie, presnejšie povedané, bez masívneho rozvoja IT za posledných 20 rokov by ku globalizácii v tejto miere, akú vidíme dnes, nemohlo dôjsť. Spoločnosti sa rozširujú, vytvárajú pobočky nielen v materskej krajine, ale po celom svete.

Pre mnohé podniky a organizácie, ktoré majú svoje pobočky vo viacerých mestách, štátoch či na rôznych svetadieloch je nevyhnutné, aby mali svoje obchodné údaje k dispozícii v maximálnej možnej miere, pričom dôležitou podmienkou je aj ich bezpečnosť. Veľa spoločností tento problém rieši využívaním cloudových služieb, avšak údaje strategického významu si nechávajú vo vlastnej réžii. Tento krok je pochopiteľný, nakoľko vo viacerých vyspelých štátoch legislatíva nedokáže ochrániť citlivé údaje na cloudoch, ktoré ako jedno z distribuovaných riešení využíva virtualizáciu a zdieľanie hardvéru. Možnosť využívania vlastnej hardvérovej infraštruktúry na geograficky vzdialených miestach s využívaním dostupných riešení bezpečnej komunikácie viedlo viacero spoločností k vytvoreniu vlastného distribuovaného riešenia.

V tomto príspevku sa zameriame na problémy spojené s distribuovanými databázami spoločnosti, ktorá prevádzkuje sieť obchodov nielen na Slovensku, ale aj v ďalších troch

[1] Ekonomická univerzita, Fakulta hospodárskej informatiky, Katedra aplikovanej informatiky, Dolnozemská cesta 1, 852 35 Bratislava, peter.schmidt@euba.sk

[2] Ekonomická univerzita, Fakulta hospodárskej informatiky, Katedra aplikovanej informatiky, Dolnozemská cesta 1, 852 35 Bratislava, pavol.jurik@euba.sk

európskych krajinách. Napriek prítomnosti na štyroch regionálnych trhoch nejde o veľkého nadnárodného giganta, nakoľko ich sortiment je špecifický a určený živnostníkom a podnikateľom. V každej krajine má od troch do piatich pobočiek. Ekonomická situácia spoločnosti napriek tomu, že umožňuje veľmi pomalý rast, neumožňuje obrovské investície do distribuovanej informatickej infraštruktúry. Cloudové riešenia sú pre nich vhodné len sčasti a centralizované spracovanie dát sa ukázalo ako náročné a málo flexibilné. Pri centralizovanom spracovaní dochádzalo k spomaleniu práce kvôli preťaženiu serverov a častým výpadkom na miestach s horším internetovým pripojením.

2 Základné pojmy

Distribuovaný systém môžeme definovať nasledovne:

- je systém viacerých autonómnych prvkov, ktoré spolupracujú na dosiahnutí spoločného cieľa. Zo systému sú vylúčené tie siete, v ktorých serverové uzly fungujú bez spoločného účelu (Burns & Wellings, 2009);
- je systém autonómnych výpočtových prvkov, ktoré sa javia používateľovi ako jednotný systém. Táto definícia sa zaoberá dvoma charakteristickými črtami distribuovaných systémov:
  - ide o súhrn výpočtových prvkov, z ktorých každý môže nezávisle fungovať;
  - používatelia veria, že pracujú s jednotným systémom. To znamená, že serverové uzly musia medzi sebou nebadane spolupracovať (Steen & Tanenbaum, 2017);
- je systém, v ktorom sú jednotlivé prvky prepojené sieťou. Tieto prvky medzi sebou komunikujú prostredníctvom zasielania údajov a správ. Cieľom distribuovaného systému je spolupráca všetkých serverových uzlov k dosiahnutiu spoločného cieľa;
- pri distribuovaných systémoch je potrebné, aby boli všetky dáta používateľovi k dispozícii aj pri prerušení spojenia, tzn. je vhodné vytvoriť také distribuované systémy, ktoré pri výpadku spojenia vytvoria zálohy odoslaných transakcií. Tieto systémy fungujú prostredníctvom siete VPN (Kultan, Schmidt, & Mukhambetova, 2019).

2.1 Informačné systémy z hľadiska prístupu k údajom môžeme klasifikovať nasledovne:

Centralizované IS:

- predpokladajú vytvorenie jedinej centrálnej databázy na hlavnom serverovom uzle, ku ktorému budú mať prístup ostatní používatelia. Na získanie prístupu k údajom je potrebné sieťové pripojenie. Spoľahlivosť a dostupnosť v systéme je závislá od dostupnosti hlavného serverového uzla, nakoľko výpadok centrálneho serverového uzla spôsobí nefunkčnosť celého systému (Kultan, Schmidt, & Mukhambetova, 2019);
- sú najintuitívnejšie a najjednoduchšie pochopiteľné. Spravidla používajú architektúru klient/server, kde sú klienti pripojení priamo k centrálnemu serveru. Toto je najčastejšie používaný typ systému v mnohých organizáciách, kde klient posiela požiadavku na firemný server a prijíma odpoveď. Keďže celý systém pozostáva z centrálneho uzla a mnohých klientov, všetci klienti sa synchronizujú s globálnymi hodinami centrálneho uzla. Zlyhanie centrálneho uzla spôsobí zlyhanie celého systému. Keď je server vypnutý, neexistuje iná entita na odosielanie požiadaviek či prijímanie odpovedí (Berty Technologies, 2019);
- v centralizovanom systéme sú všetci používatelia pripojení k centrálnemu uzlu. Centrálny uzol ukladá údaje, ku ktorým majú prístup ostatní používatelia, na základe oprávnení. Nesporná výhoda centralizovaného systému je ľahká správa

a údržba. „Upgrade“ sa vykonáva na jednom mieste. Jednoduchšie je aj riešenie zabezpečenia systému.

Decentralizované IS:

- databáza je rozdelená na jednotlivé serverové uzly po častiach. Táto architektúra má niektoré vlastnosti distribuovaného spracovania;
- ide o typ systému, ktorý si získal veľkú obľubu vďaka úspechu blockchainu. Mnoho organizácií sa snaží aplikovať tieto systémy do praxe. Každý uzol robí vlastné rozhodnutie. Konečné správanie systému je súhrn rozhodnutí jednotlivých uzlov. Neexistuje žiadna centrálna entita, ktorá prijíma a odpovedá na žiadosti. Skladá sa z uzlov a sieťového prepojenia. Každý uzol je nezávislý, má rôzne nastavené globálne hodiny. Má viac ako jednu centrálnu jednotku, ktorá môže prijímať dáta z iného uzla. Porucha jedného centrálneho uzla nespôsobí zlyhanie celého systému, ale iba časti. Všetky uzly sú rovnocenné, žiadny uzol nemá nadradenosť nad ostatnými, ani podradenosť. Môžeme hovoriť o systéme na báze peer-to-peer (GeeksforGeeks, 2019);
- decentralizovaný systém môže byť rovnako zraniteľný ako centralizovaný systém. Je však odolnejší voči poruchám, pretože v prípade zlyhania jedného alebo viacerých uzlov môžu ostatní používatelia naďalej systém využívať.

Distribuované IS:

- skladajú sa z uzlov, ktoré sú územne rozptýlené a chovajú sa ako jeden celok. Distribuované systémy sú opakom centralizovaných systémov, na rozdiel od centralizovaného systému je sieť fragmentovaná na jednotlivé serverové uzly, ktoré spolu kooperujú. Funkcie, riadenie aj logika spracovania sú riešené distribuovane (Kultan, Schmidt, & Mukhambetova, 2019);
- je systém, ktorý sa chová ako celok. Zlyhanie uzla nemá výrazný vplyv na celý systém, teda ak jeden uzol zlyhá, celý systém bude naďalej fungovať. Každý uzol rozhoduje samostatne, konečné správanie je súhrn rozhodnutí jednotlivých uzlov. Neexistuje hlavná, centrálna entita, ktorá prijíma a odpovedá na žiadosti;
- je podobný decentralizovanému systému v tom, že nemá jediného centrálneho vlastníka a eliminuje centralizáciu. Používatelia majú v systéme rovnaký prístup k údajom, avšak používateľské oprávnenia môžu byť v prípade potreby povolené. Najlepším príkladom rozsiahleho, distribuovaného systému je internet. Distribuovaný systém umožňuje používateľom zdieľať vlastníctvo údajov. Hardvérové a softvérové zdroje sú tiež rozdelené medzi používateľov, čo môže v niektorých prípadoch zlepšiť výkon systému. Distribuovaný systém je zabezpečený pred nezávislým zlyhaním komponentov, čo môže značne zlepšiť jeho dostupnosť (Berty Technologies, 2019).

Základom všetkých informačných systémov je miesto, kde sa uchovávajú informácie. Toto miesto je databáza, ktorá v závislosti od charakteru dát môže byť relačná, textová, grafická a pod. V súčasnosti sa stále najviac využívajú relačné databázy SQL (Select Query Language). Aby databáza patrila do kategórie SQL databáz, musí jej systém riadenia bázy dát (DBMS) vedieť interpretovať jazyk SQL a spracovávať jednotlivé dopyty, ktoré prichádzajú od používateľov. Nakoľko jazyk SQL je štandardizovaný, jednotlivé produkty sa líšia hlavne cenou. Samozrejme cena odzrkadľuje aj kvalitu DBMS, ale sú na trhu aj kvalitné systémy zdarma, ako napríklad obľúbený MySQL alebo Maria DB, ktoré sú aj navzájom kompatibilné.

Pre malé a stredne veľké spoločnosti je voľba DBS na báze MySQL alebo Maria DB logickou voľbou.

2.2 Homogénny versus heterogénny distribuovaný databázový systém

Distribuovaný databázový systém môžeme rozdeliť na:

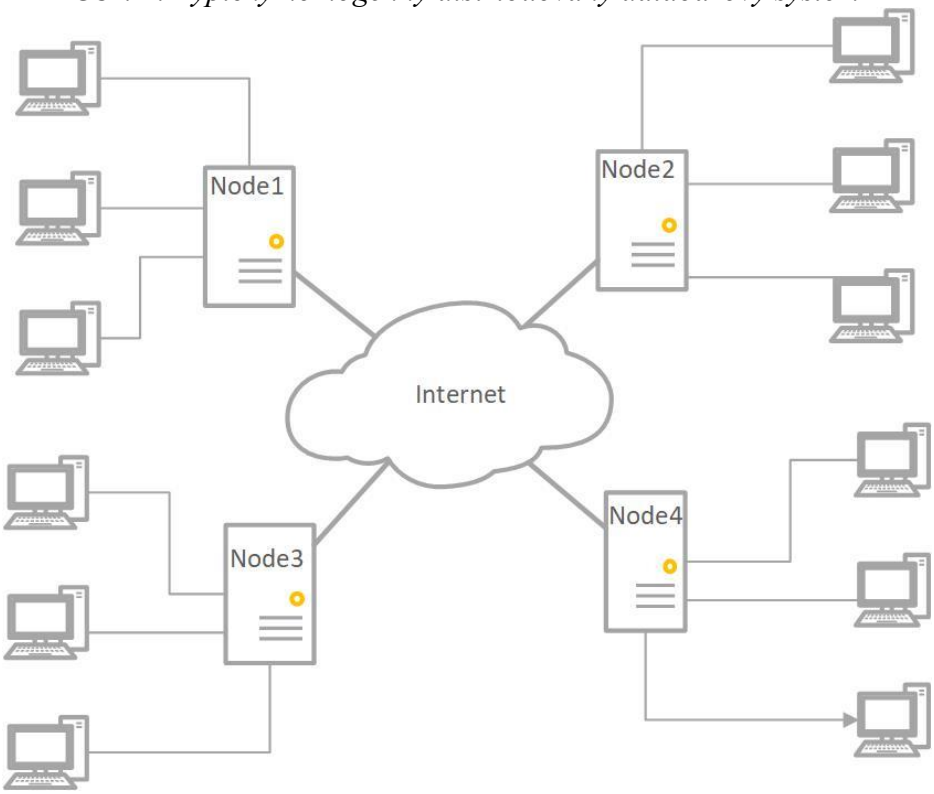
Homogénny DDBS (Homogeneous Distributed database system):

- Všetky uzly majú identický databázový softvér
- Každý uzol vie o ostatných
- Dohodnú sa na spolupráci pri spracovaní používateľských dotazov
- Lokálne uzly sa vzdávajú časti svojej autonómie, pokiaľ ide o ich právo meniť schémy alebo softvér DBMS
- Tento softvér musí tiež spolupracovať s ostatnými uzlami pri výmene informácií o transakciách, aby bolo možné spracovanie transakcií vo viacerých uzloch.

Heterogénny DDBS (Heterogeneous Distributed database system)

- Rôzne schémy a rôzny databázový softvér
- Uzly o sebe nemusia navzájom vedieť
- Môžu poskytovať obmedzené možnosti spolupráce pri spracovaní transakcií
- Rozdiely v schémach môžu predstavovať problém pri spracovaní transakcií
- Rozdielnosť softvéru môže byť prekážkou pri spracovaní najmä tých transakcií, ktoré pristupujú k viacerým uzlom (Chakraborty, 2018).

Obr. 1: Typický homogénny distribuovaný databázový systém



Zdroj: Vlastné spracovanie

Pre našu štúdiu neuvažujeme o heterogénnych DDBS, nakoľko heterogenita značne sťažuje proces replikácie. Mimochodom heterogénne DDBS sa najčastejšie vyskytujú pri GRID computingu a pri Cloudoch. Pre nás je najlogickejšia voľba homogénny DDBS.

2.3 Problém

Aj keď open-source DBS vyvíjajú pomerne veľké komunity vývojárov, v určitých oblastiach zákonite zaostávajú. Nakoľko MySQL je už pod správou Oracle, je logické že si materská firma nebude tvoriť konkurenta, ktorý je navyše zdarma, takže vývoj MySQL je takpovediac pozvoľný. Prejavuje sa to hlavne v možnostiach duplikácie a replikácie dát, ktoré sú kľúčové, ak by sme MySQL chceli použiť ako distribuovaný DBS.

2.4 Čo je replikácia dát?

Máme jednu kópiu databázy a z nejakého dôvodu potrebujeme ďalšiu kópiu. Ak máme na jednom uzle DB a na druhom uzle jej kópiu, môžeme hovoriť o duplikáte na homogénnom DDBS. Poznámka: *Viacerí používatelia nerobia rozdiel medzi duplikátom DB a zálohou DB. Rozdiel je v tom, že záloha sa vytvára k určitému času a po zapísaní sa do nej už neukladajú žiadne zmeny, vďaka tomu sa dá DB k danému času zo zálohy reštaurovať. Duplikát DB je funkčná, živá DB, na ktorej sa prejavia všetky zmeny, ktoré sa vykonali v originálnej DB na základe duplikačnej politiky.*

Replikácia je sofistikovanejšia a má rôzne formy, rôzne porovnávacie osi ako mieru synchronizácie, počet záznamových serverov, formáty zmien a pod.

Vývojári implementujú do svojich DDBS určitú podporu replikácie dát. Kým v starších verziách MySQL replikácia podporovaná nebola, tak už pri verzii MySQL 5.7 je podpora zjavná. Napriek týmto snahám, ešte nemôžeme povedať, že vieme na báze MySQL 5.7 vytvoriť DDBS. Na správne fungovanie DDBS má spoľahlivá a rýchla replikácia dát kľúčový vplyv.

2.5 Aké máme možnosti replikácie dát?

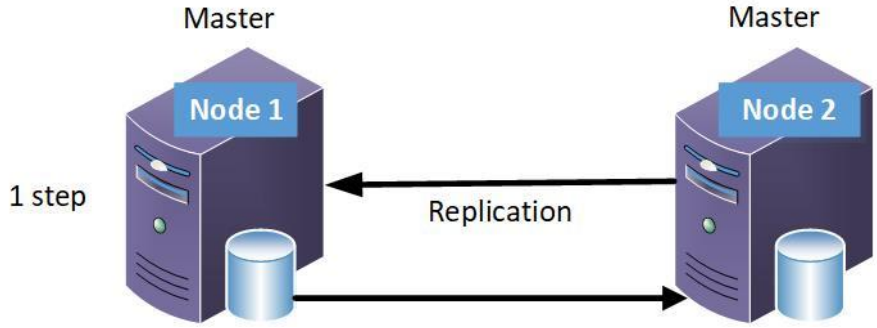
V princípe rozoznávame Master-Slave a Master-Master replikáciu. Replikácia **Master-Slave classic** znamená, že sa všetky zmeny uložia na jeden server, potom sa skopírujú do množstva replík, avšak tento spôsob je typicky klient-server.

Master-master true znamená, že sa zmeny uložia na viac master uzlov „súčasne“ tak, že sa z jedného na druhý replikujú. Je jasné, že keď máte jeden originál a z neho niekoľko replík, ktoré by mali (ideálne okamžite) tento originál kopírovať, potom je pomerne jednoduché si predstaviť, že tento model je vhodný napr. pri prvotnej tvorbe replík, keď je náš DDBS tvorený rovnakými databázami, čiže rovnaké schémy a rovnaké dáta, bez fragmentácií.

Problémy začínajú pri Master-Master replikácii, a nie len v prípade MySQL, ale vo všeobecnosti. Čisto teoreticky sa zamyslime, čo by sa stalo, keby sme sa pokúsili spustiť rovnakú transakciu na dvoch uzloch súčasne, ktorá by mala zmeniť rovnaké údaje, ale rôznymi spôsobmi. Je zrejmé, že tieto dve zmeny nemôžeme použiť súčasne. V okamihu, keď na jednom uzle začneme niečo meniť, na druhom uzle ešte nič nie je. Nastáva konflikt. Jedna z transakcií bude musieť byť vrátená späť. Práve na vyriešenie takýchto situácií je nevyhnutné mať uzly synchronizované.

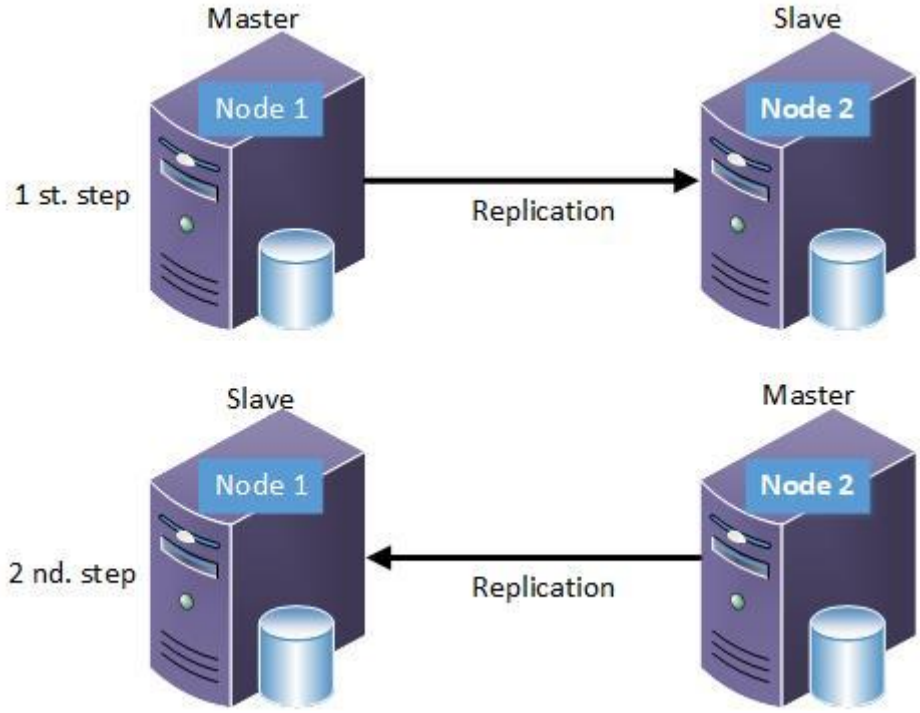
Multi-master replication je metóda replikácie databázy, ktorá umožňuje ukladanie údajov skupinou počítačov a aktualizáciu ktorýmkoľvek členom skupiny. Všetci členovia reagujú na dopyty týkajúce sa údajov o klientoch. Replikačný systém s viacerými hlavnými servermi je zodpovedný za šírenie úprav údajov vykonaných každým členom do zvyšku skupiny a za riešenie akýchkoľvek konfliktov, ktoré by mohli vzniknúť medzi súbežnými zmenami vykonanými rôznymi členmi. Multi Master je podobná topológii Master / Slave, s tým rozdielom, že oba uzly sú súčasne master aj slave. To znamená, že medzi uzlami bude kruhová replikácia. Odporúča sa nakonfigurovať obidva servery na protokolovanie transakcií z replikačného vlákna (log-slave-updates), ale ignorovať svoje vlastné už replikované transakcie (nastaviť replicate-same-server-id na 0), aby sa zabránilo nekonečným cyklom v replikácii. Toto je potrebné nakonfigurovať aj pri povolenom GTID.

Obr. 2: Master – Master cross replication



Zdroj: Vlastné spracovanie

Obr. 3: Master – Slave cross replication

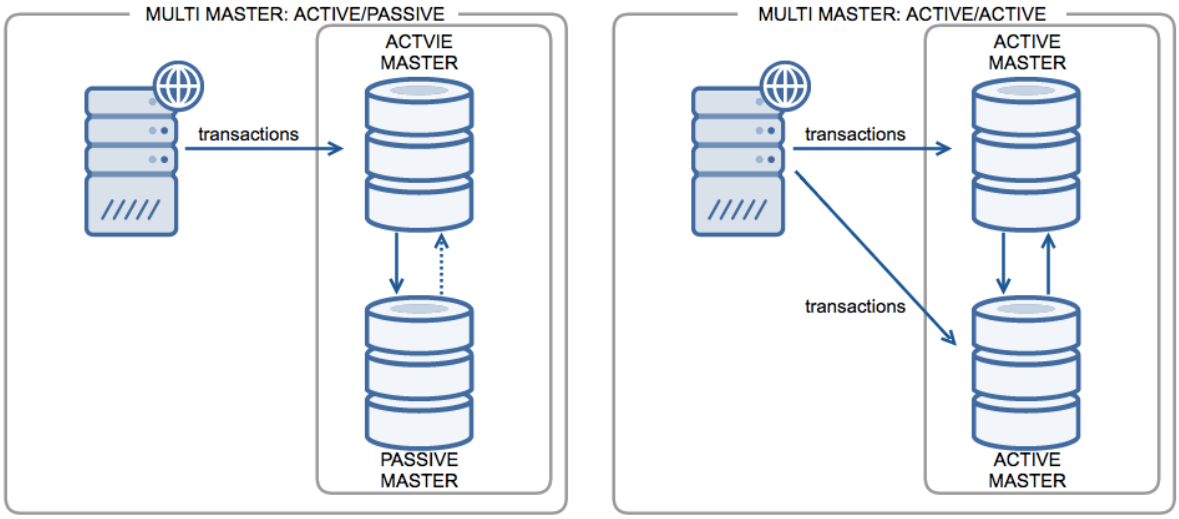


Zdroj: Vlastné spracovanie

Multi master topológie je možné nakonfigurovať tak, aby mali buď takzvané aktívne/pasívne nastavenie, kde je možné zapisovať iba do jedného uzla a druhý uzol je v aktívnom pohotovostnom režime. Iný spôsob nastavenia je aktívne / aktívne nastavenie, kde sú oba uzly pripravené na zapisovanie (Sharif, 2020).

Primárnym účelom replikácie s viacerými master uzlami je zvýšená dostupnosť a rýchlejšia doba odozvy servera. Komunikácia a replikácia v systémoch Multi-master sa často vykonáva pomocou typu konsenzuálneho algoritmu, ale dá sa implementovať aj pomocou vlastných algoritmov.

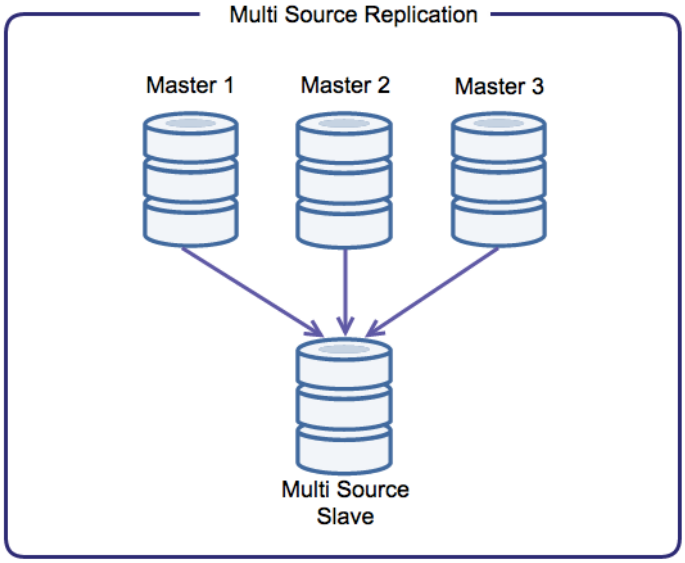
Obr. 4: Multi master replikácia



Zdroj: (Sharif, 2016)

Multi source Replication – viaczdrojová replikácia je podporovaná už od MariaDB 10.0 a MySQL 5.7. V zásade to znamená, že replika je povolená pre replikáciu z viacerých Master. Ak to chceme povoliť, replika nemôže mať viac vzorov, ktoré by sa zapisovali do tej istej schémy, pretože by to viedlo ku konfliktom množiny zápisov.

Obr. 5: Multi Source Slave replikácia

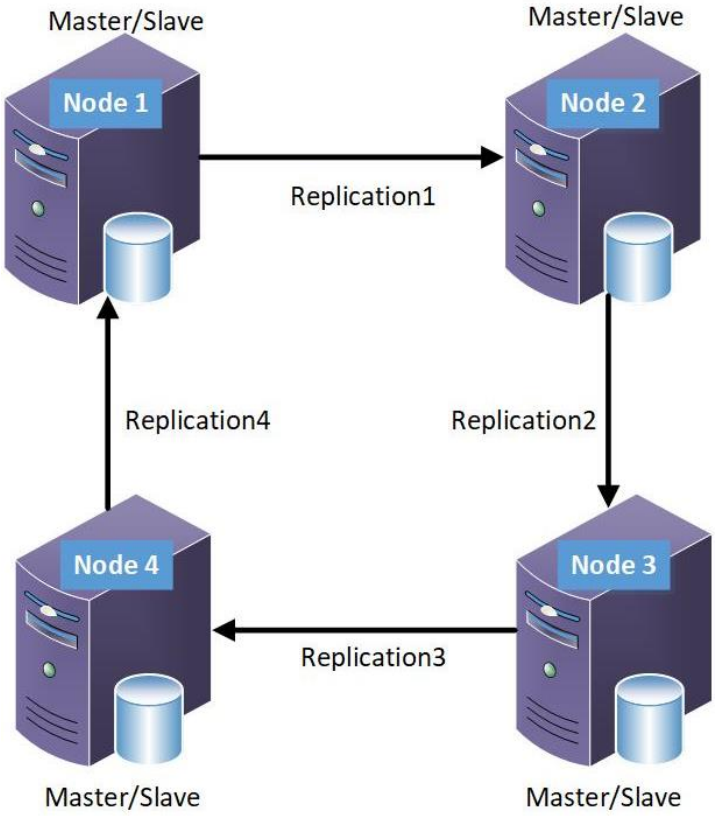


Zdroj: (Sharif, 2016)

Kruhová replikácia (Circular replication) – rozdiel oproti replikácii dvoch uzlov master-master je v tom, že ak máte viac ako dva uzly, replikácia prebieha v kruhu, t. j. pri štyroch uzloch replikácia prebieha z uzla1 do uzla2, z uzla2 do uzla3, z uzla3 do uzla4 a z uzla4 do uzla1 (Timme, n. d.).

Problémom tejto replikácie, pri ktorej je potrebné si vytvoriť replikačného používateľa (user for replication), je jej veľmi nízka spoľahlivosť aj keď je replikácia nadefinovaná priamo v DBMS (SRBD). Situácia sa veľmi nezlepší, aj keď si vytvoríme identické uzly.

Obr. 6: Circular replication on 4 nodes



Zdroj: Vlastné spracovanie

2.6 Aké údaje prenášať medzi jednotlivými uzlami počas replikácie?

Môžeme prenášať samotné dopyty (query) alebo môžeme prenášať iba zmenené reťazce. Jedno aj druhé dokáže uskutočniť ľubovoľný systém DBMS, v ktorom existujú dopyty, ktoré generujú veľký počet zmien, čiže aktualizáciu veľkého množstva údajov. Vynára sa otázka, čo konkrétne budeme kopírovať? Samotné dopyty môžeme riadiť tam a späť medzi uzlami, alebo môžeme replikovať iba zmenené údaje. Ak by sme sa rozhodli pre dopyty, tak sa nám na ich uchovávanie bude hodiť napr. textový súbor, ale ak si vyberieme prenos jednotlivých záznamov, tak budeme pravdepodobne potrebovať dočasnú tabuľku (temporary table).

Pripomeňme si, že replikácia žije na logickej úrovni, ktorá nemá nič spoločné s úrovňou fyzického úložiska. Ak chceme zachovať synchrónny stav databáz na všetkých serveroch súčasne, musíme použiť replikáciu.

2.7 Riešenie

Vrátime sa k pôvodnému problému, ktorý sme chceli v tomto článku riešiť. Spomínaná spoločnosť má v každej krajine, kde je zastúpená od 3 do 5 pobočiek. Na zabezpečenie chodu pobočiek/predajní potrebuje distribuované riešenie svojich databáz, aby všetky pobočky mali prehľad o stave zásob tovarov v danej krajine, prípadne dostupnosti tovarov vo svojich zahraničných pobočkách. Kvôli jednoduchosti sú na jednotlivých firemných uzloch úplné repliky celých databáz, pričom sa pravidelne vytvárajú zálohy. Výhoda tohto prístupu je veľká redundancia a vysoká dostupnosť dát aj keď nefunguje konektivita medzi uzlami. V prípade výpadku internetového pripojenia, spoločnosť zaviedla jednoduché ale účinné opatrenie, ktoré zabraňuje zmene záznamov, ktorých autorom/vlastníkom nie je daná pobočka. Všetky ostatné činnosti pobočky môžu vykonávať. Po obnovení konektivity je nutné znovu dosiahnuť konzistentný stav medzi jednotlivými uzlami, spustením replikácií. Aby sa obmedzili zlyhania

navrhla sa fully mesh topológia replikácie. Tento prístup síce zvyšuje réžiu siete, ale dokáže zabezpečiť udržanie, alebo vrátenie DB do konzistentného stavu po výpadku siete.

3 Záver

V tejto štúdii sme sa snažili poukázať na problémy pri tvorbe distribuovaného systému na databázovej platforme, ktorá primárne nebola na takéto účely vyvinutá. Vzniká otázka, prečo nepoužiť databázový systém určený priamo na distribuované spracovanie dát. Odpoveď je veľmi jednoduchá, databázové technológie určené na distribuované spracovanie dát sú veľmi drahé a v žiadnom prípade nie sú jednoduché. Okrem ich kúpnej ceny treba počítať s vysokými nákladmi na ich údržbu. Výhodou prezentovaného riešenia je, že náklady na nákup softvérových prostriedkov boli 0,- eur, nakoľko MySQL databáza je dostupná zdarma. Sme presvedčení, že aj hybridné riešenie, ktoré využíva naprogramovanú replikačnú logiku so správnou konfiguráciou DBMS, dokáže vytvoriť funkčný a spoľahlivý distribuovaný databázový systém, ktorý sa dá používať v bežnej prevádzke. Navrhnutý systém, je schopný sa po výpadku siete, alebo niektorých uzlov, po obnovení komunikácie sám dostať do konzistentného stavu, čiže repliky na každom uzle budú totožné. Podmienkou spoľahlivého fungovania je zakázať manuálnu manipuláciu s databázou.

Príspevok bol spracovaný v rámci riešenia grantovej úlohy KEGA 019EU-4/2020 *Podpora dištančného vzdelávania prostredníctovm virtuálnej katedry*.

Literatúra

[1] Berty Technologies. (2019, June 20). Centralized vs Decentralized vs distributed Systems · Berty Technologies. Retrieved May 22, 2021, from https://berty.tech/blog/decentralized-distributed-centralized

[2] Burns, A., & Wellings, A. (2009). *Real-Time systems and programming languages: Ada, Real-Time Java and C/Real-Time POSIX*. Addison-Wesley.

[3] Chakraborty, A. (Director). (2018, January 30). *DBMS - distributed database system* [Video file]. Retrieved May 22, 2021, from https://www.youtube.com/watch?v=aUyqZxn12sY

[4] GeeksforGeeks. (2019, April 30). Comparison - CENTRALIZED, decentralized and distributed systems. Retrieved May 22, 2021, from https://www.geeksforgeeks.org/comparison-centralized-decentralized-and-distributed-systems/

[5] Kultan, J., Schmidt, P., & Mukhambetova, M. (2019). Distributed data processing in the economy. *Proceedings of the International Conference on Economics, Management and Technology in Enterprises 2019 (EMT 2019)*. doi:10.2991/emt-19.2019.23

[6] OLACOM. (n.d.). Mysql replication of individual tables. Retrieved from https://olacom.ru/mozilla-firefox/mysql-replikaciya-otdelnyh-tablic-nastroika-replikacii-master-slave-v-mysql/

[7] Sharif, A. (2020, October 01). The difference between mysql multi-master and multi-source replication. Retrieved May 16, 2016, from https://severalnines.com/database-blog/difference-between-mysql-multi-master-and-multi-source-replication#:~:text=Multi%20source%20replication%20is%20supported,conflicts%20in%20the%20write%20set.

[8] Steen, M. V., & Tanenbaum, A. S. (2017). *Distributed systems*. (Udne sted): Maarten van Steen.

[9] Timme, F. (n.d.). Setting up master-master replication on four nodes with mysql 5 on debian etch. Retrieved May 22, 2021, from https://www.howtoforge.com/setting-up-master-master-replication-on-four-nodes-with-mysql-5-on-debian-etch