ŽILINSKÁ UNIVERZITA V ŽILINE

FAKULTA RIADENIA A INFORMATIKY

AUTOREFERÁT
DIZERTAČNEJ PRÁCE

Žilina, apríl 2021 **Ing. Roman Čerešňák**

ŽILINSKÁ UNIVERZITA V ŽILINE
FAKULTA RIADENIA A INFORMATIKY

Ing. Roman Čerešňák

Autoreferát dizertačnej práce

SPRACOVANIE A VYHĽADÁVANIE V NEŠTRUKTÚROVANÝCH DATABÁZACH

na získanie akademického titulu „**philosophiae doctor**“ (**v skratke PhD.)**
v študijnom programe doktorandského štúdia
aplikovaná informatika

v študijnom odbore
informatika

Žilina, apríl 2021

Dizertačná práca bola vypracovaná v dennej forme doktorandského štúdia **na Katedre informatiky, Fakulte riadenia a informatiky Žilinskej univerzity v Žiline**.

Predkladateľ: **Ing. Roman Čerešňák**
Katedra informatiky
Fakulta riadenia a informatiky
Žilinská univerzita v Žiline

Školiteľ: **doc. Ing. Michal Kvet, PhD.**
Katedra informatiky
Fakulta riadenia a informatiky
Žilinská univerzita v Žiline

Oponent 1: **doc. Ing. Jarmila Škrinárová, PhD.**
Katedra informatiky
Fakulta prírodných vied
Univerzita Mateja Bela v Banskej Bystrici

Oponent 2: **prof. Ing. Marcel Harakaľ, PhD.**
Katedra informatiky
Akadémia ozbrojených síl generála Milana Rastislava Štefánika
AOS Liptovský Mikuláš

Autoreferát bol rozoslaný dňa: .....

Obhajoba dizertačnej práce sa koná dňa 19.8.2021 o 14 h. pred komisiou pre obhajobu dizertačnej práce schválenou pracovnou skupinou odborovej komisie v študijnom odbore **informatika** v študijnom programe **aplikovaná informatika**, vymenovanou dekanom Fakulty riadenia a informatiky Žilinskej univerzity v Žiline dňa .....

prof. Ing. Karol Matiaško, PhD.
predseda pracovnej skupiny odborovej komisie
v študijnom odbore **informatika**
v študijnom programe **aplikovaná informatika**

Fakulta riadenia a informatiky
Žilinská univerzita
Univerzitná 8215/1
010 26 Žilina

1. Úvod

Potreby ukladania a spracovania údajov v moderných internetových službách, ako sú sociálne siete, online nakupovanie, analýza údajov a vizualizácia, si vyžadovali nový druh úložných systémov, nazývaných databázy NoSql (nielen Sql). Takéto databázy používajú nové techniky, ktoré podporujú paralelné spracovanie a replikáciu údajov vo viacerých uzloch, aby sa zabezpečil zlepšený výkon a dostupnosť údajov [39].

Väčšina údajov, ktoré boli v minulosti vytvorené, boli uložené a v mnohých prípadoch sú stále ukladané do relačných databáz, akými sú Oracle, MySql, Microsoft Sql prípadne PostgreSql. S nárastom popularity NoSql databáz sa viaceré spoločnosti domnievali, že vlastnosti nerelačných databáz oproti vlastnostiam relačných databáz môžu pozitívnym spôsobom ovplyvniť prácu s údajmi. Za týmto účelom vytvorili niekoľko experimentov, ktorých cieľom bol presun dát z relačných databáz na nerelačné databázy. Hlavnými cieľmi, ktoré sme sa pri experimentoch snažili dosiahnuť bolo zachovanie referenčných integrít, dátových typov a jasne definovaných vzťahov medzi tabuľkami respektíve kolekciami. Mnohé zo štúdií, ktoré sa venovali presunu nahromadených dát v relačných databázach, vytvorili metodiky a všeobecné metódy, ktoré majú za cieľ presunúť dáta z jedného typu databáz na iný typ v snahe zachovania dátových typov a iných obmedzení [61] [38] .

Počas transformačného procesu je nutné zachovať nielen dostupnosť dát, ale taktiež stav, kedy môžu byť údaje ovplyvnené dátami, ktoré vstupujú do spusteného procesu. Medzi spomenuté operácie patria príkazy *Update*, ktoré menia hodnotu daného záznamu prípadne príkaz *Delete*, ktorý vymazáva záznam, a tým pádom, nie je potrebné danú hodnotu upravovať a následne ukladať do cieľovej databázy. V prípade ignorovania alebo nevytvorenia mechanizmu na úpravu dát v spustenom transformačnom procese by bolo nutné dáta po presunutí zo zdrojovej databázy na cieľovú znovu prehľadať a tento fakt by mal za následok nárast času potrebného na presun údajov [115].

Navyše nerelačné databázy, ktoré majú spĺňať kritérium bezproblémovej obsluhy takzvanej 3V (Velocity, Velocity, Variety) musia spoľahlivo a efektívne vyhľadávať údaje aj s narastajúcim množstvom dát v konkrétnej nerelačnej databáze. Čím ďalej tým viac narastajú požiadavky nie len na databázu, ale aj na iné odvetvia informačných technológií na rýchlejšie získavanie dát a zrýchlenie procesov. Práve zrýchlenie časového hľadiska a redukcia náročnosti na databázu sú pre nás kľúčovým faktorom. Ako ukážeme v ďalšej časti práce, súčasné riešenia neposkytujú dostatočne rýchlu odozvu na získanie objektov.

2. Súčasný stav riešenej problematiky

Distribuované spracovanie údajov je metóda, pri ktorej viac počítačov distribuovaných na rôznych miestach spracúva dáta cez komunikačnú sieť tvorenú oddelenými pc. Tento spôsob je diametrálne odlišný od spôsobu centralizovaného servera, ktorý riadi a poskytuje možnosti spracovania všetkým pripojeným systémom z jedného centrálneho servera.

Počítače, ktoré tvoria distribuovanú sieť na spracovanie údajov, sú umiestnené na rôznych miestach, ale sú vzájomne prepojené prostredníctvom bezdrôtových alebo satelitných spojení.

Masívny nárast údajov v posledných rokoch vedie k rastúcemu dopytu po spracovaní veľkých údajov v moderných dátových centrách, ktoré sú zvyčajne distribuované v rôznych geografických regiónoch, napríklad v 13 dátových centrách spoločnosti Google v 8 krajinách na 4 kontinentoch [52]. Analýza veľkých údajov preukázala svoj veľký potenciál pri odhaľovaní cenných poznatkov o údajoch s cieľom zlepšiť rozhodovanie, minimalizovať riziká a vyvíjať nové produkty a služby. Na druhej strane sa veľké údaje už premietli do vysokých nákladoch kvôli vysokému dopytu po výpočtových a komunikačných zdrojoch [123]. Výskumníci predpokladajú, že náklady na hardvér dátových centier budú neustále rásť, s čím súvisia aj ich vysoké výdavky na zvyšovanie efektivity a dopyt od zákazníka. Zo spomenutého dôvodu je nevyhnutné analyzovať problém minimalizácie nákladov na spracovanie veľkých údajov, ktoré sú distribuované na viacerých inštanciách dátových centier.

Na optimalizáciu cien sa pri optimálnych výpočtových jednotkách vyvinuli viaceré metódy. Ako hlavný nedostatok sme spozorovali zanedbanie návrhu veľkosti dátového centra na zníženie výpočtových nákladov úpravou počtu aktivovaných serverov prostredníctvom umiestnenia úloh [110]. Na základe zmeny veľkosti dátového centra sa navrhla štúdia zameriavajúca sa na geografické rozmiestnenie dátových centier [121] [68]. Problém s distribuovane spracovanými dátami súvisí aj s efektívne navrhnutou komunikáciou. Zaznamenali sme štúdie, ktoré sa venujú spomenutému problému v snahe zlepšenia údajov umiestnením úloh na server, na ktorých sa umiestnia vstupné údaje, aby sa zabránilo vzdialenému načítaniu záznamov [43].

Aj keď vyššie spomenuté štúdie priniesli určité pozitíva pri distribuovane spracovaných procesoch, diametrálne sa odlišujú od systému s nákladovo efektívnym spracovaním údajov a to z nasledujúcich dôvodov:

- ***plytvanie zdrojmi*** - pre dáta, ktoré sa v systéme vyskytujú je nutné zabezpečiť efektívnu správu. Niektoré dáta sa v systéme vyskytujú častejšie, k niektorým údajom sa pristupuje menej často, a preto nie je potrebné ich mať neustále k dispozícii. Tento fakt nám umožňuje efektívne manipulovať s dátami, ktoré nie sú často dopytované, a tým sa budú redukovať náklady, prípadne redukovať množstvo serverov potrebných na spracovanie,
- ***spojenie v sieťach*** - rýchlosť dátových centier je ovplyvnená viacerými faktormi, s ktorými súvisí aj ich nákladová vlastnosť [77]. Existujúca stratégia smerovania medzi dátovými centrami však nevyužíva rozmanitosť prepojení sietí dátových centier. Z dôvodu obmedzení kapacity ukladania a výpočtov nie je možné všetky úlohy ukladať na ten istý server, na ktorom sú uložené príslušné údaje. Je nevyhnutné, aby sa určité údaje stiahli zo vzdialeného servera,
- ***neefektivita údajov*** - v systéme sa vyskytujú údaje, ktorých replikovateľnosť môže závisieť aj od ich vhodného využívania. Často používané dáta vyžadujú väčšiu replikáciu z dôvodu ich kľúčovej úlohy ako dáta, ku ktorým sa počas existencie údajov často nepristupuje. Tým pádom by dochádzalo k redukcii zaťaženia jednotlivých serverov, a taktiež by sa redukovalo množstvo spotrebovaného úložného priestoru.

Aby sme prekonali nedostatky existujúcich riešení, zaoberáme sa problémom nákladov na spracovanie veľkých dát, a tiež efektívnu replikáciu údajov medzi viacerými servermi. Pri našej optimalizácii nákladov berieme do úvahy obmedzené množstvo výpočtových prostriedkov pri optimálnom množstve poskytnutých serverov. Našim cieľom je optimalizovať ukladanie veľkého množstva dát, eliminovať množstvo potrebných replikácií. Naše hlavné príspevky sú zhrnuté takto:

- Optimalizovať množstvo paralelných procesov, ktoré sú využívané pri spracovaní veľkého množstva údajov, a tým zamedziť nadmernému vyťaženiu serverov.
- Optimalizovanie množstva replikovaných údajov na základe váhy jednotlivých údajov, a tým optimalizovať množstvo dátových jednotiek potrebných na správu údajov.

3. Požiadavky na distribuované spracovanie dát, vyhľadávanie v nerelačných databázach a ciele práce

V súčasnosti rapídne narastá objem heterogénnych dát z distribuovaných zdrojov, čím vznikajú nové výzvy pri extrahovaní údajov. Takými aplikáciami môže byť modelovanie, simulácia, rozpoznávanie obrazov, vizualizácia a podobne v rôznych oblastiach, ako sú napr. biomedicína, astrofyzika, životné prostredie, aeronautika, automobilový priemysel, energetika prípadne materiálové vedy. Vzhľadom na veľkosť dát, ktoré sú často označované ako veľké, resp. extrémne, je potrebné navrhnúť metodológiu, robustné metódy a nástroje pre extrémne škálovateľnú analytiku v súčinnosti s distribuovanými architektúrami pre zber a manažovanie obrovského množstva dát, akými sú cloud-ové technológie a IoT. Aj keď sa v danej problematike urobil veľký krok vpred, stále existuje niekoľko problémov, ktoré by dokázali zlepšiť distribuované spracovanie dát.

Pri spracovávaní veľkého množstva údajov existuje niekoľko výziev, ktoré sme na základe logických súvislostí rozdelili do 10 kategórií:

1. ***Zber údajov*** - Úplne prvou výzvou pri spracovaní údajov je zber alebo získanie správnych údajov pre vstup.
2. ***Duplicita údajov*** - Keďže sa údaje zhromažďujú z rôznych zdrojov údajov, často sa stáva, že dôjde k duplicite údajov. Rovnaké záznamy a entity sa môžu počas fázy kódovania údajov vyskytnúť niekoľkokrát. Tieto duplicitné údaje sú nadbytočné a môžu viesť k nesprávnym výsledkom.
3. ***Nekonzistencia údajov*** - Z dôvodu zhromažďovania obrovského množstva údajov, neexistuje záruka, že informácie budú úplné alebo že všetky polia, ktoré potrebujeme, sú vyplnené správne. Údaje môžu byť navyše nejednoznačné.
4. ***Rozmanitosť údajov*** - Vstupné údaje, ktoré sa zhromažďujú z rôznych zdrojov, môžu obsahovať rôzne formy. Riadky a stĺpce relačnej databázy neobmedzujú údaje. Údaje sa líšia od aplikácie k aplikácii a od zdroja po zdroj. Väčšina týchto údajov je neštruktúrovaná a nezmestia sa do tabuľky alebo do relačnej databázy.
5. ***Integrácia údajov*** - Dátová integrácia znamená kombinovať údaje z rôznych zdrojov a prezentovať ich v jednotnom zobrazení. S rastúcou rozmanitosťou údajov a rôznymi formátmi údajov sa výzva integrovať údaje stáva čoraz väčšou.

6. ***Objem a ukladanie údajov*** - Pri spracovávaní veľkých dát je objem dát značný. Veľké dáta pozostávajú zo štruktúrovaných aj neštruktúrovaných údajov. Patria sem údaje dostupné na stránkach sociálnych sietí, záznamy spoločností, údaje zo zdrojov sledovania, údaje o výskume a vývoji, a oveľa viac. Prichádza výzva ukladať a spravovať tento obrovský objem dát. Ďalšou výzvou je, aké množstvo dát má byť obsiahnuté v RAM, aby bolo spracovanie rýchlejšie a využitie zdrojov inteligentné.
7. ***Zlý popis a meta dáta*** - Jedným z hlavných zdrojov vstupných údajov sú údaje, ktoré sa časom ukladajú do relačnej databázy. Problémom týchto dát je nesprávne naformátované a neexistuje meta popis úložiska, štruktúry a vzájomného vzťahu dátových entít. Tento scenár sa ešte zhoršuje, ak je objem dát veľký a samotná databáza je prepojená s inými databázami. Bez náležitej dokumentácie databázy je dosť ťažké vyťažiť správne vstupné údaje z databáz.
8. ***Úprava sieťových údajov*** - Dáta sú distribuované a navzájom spolu súvisia v zložitej štruktúre. Výzvou je upraviť štruktúru údajov alebo k nim pridať nejaké údaje. Internet je sieť, ktorá pozostáva z rôznych údajov, množstva aplikácií a webových stránok generujúce údaje, ktoré majú rôzne formy a vlastnosti. Schéma ich všetky prepája.
9. ***Bezpečnosť*** - V dátovom poli zohráva najdôležitejšiu úlohu bezpečnosť. Hackovanie údajov môže mať za následok únik údajov. Pre spoločnosť zaoberajúcu sa spracovaním údajov môže predstavovať zvýšené náklady. Hacker môže pri veľkom úsilí dokonca zmeniť alebo vymazať údaje, ktoré sme získali a spracovali.
10. ***Náklady*** - Náklady sú vecou úvahy. Keď sa množstvo údajov zvýši, potom sa náklady v každej fáze spracovania údajov zvyšujú postupne.

Ako ďalšiu súčasť práce tvorí vyhľadávanie v nerelačných databázach. Dopyt po dátach je v dnešnej dobe enormný. Na rozdiel od relačných databáz, je vyhľadávanie v nerelačných databázach ovplyvnené heterogénnosťou údajov, s čím súvisí viacero komplikácií ako je vyhľadávanie podľa určitého atribútu alebo vytváranie indexov nad jedným prípadne skupinou atribútov.

Na základe spomenutých výziev a požiadaviek na zefektívnenie procesu vyhľadávania v nerelačných databázach, sme stanovili nasledujúce ciele:

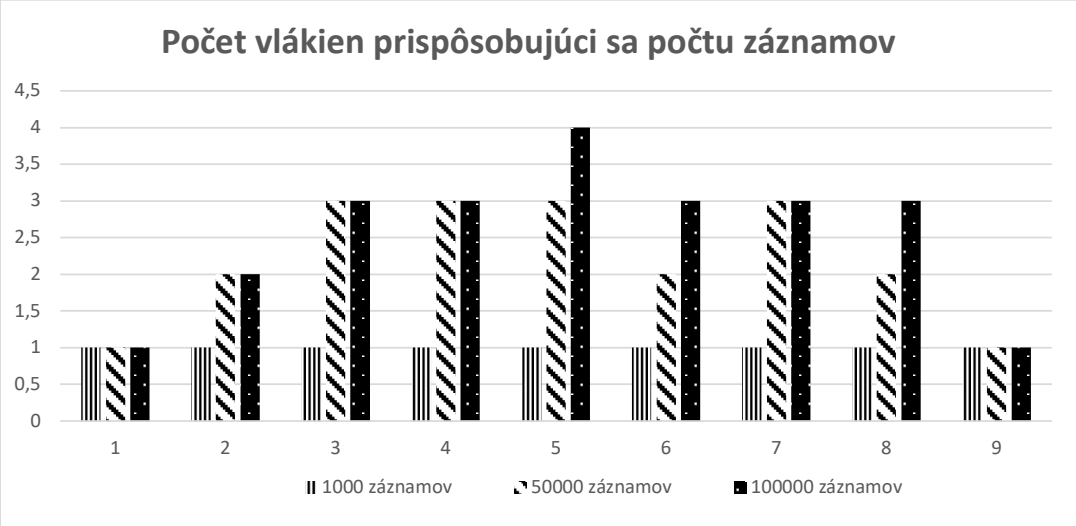
- Navrhneme architektúru a algoritmus, ktorý dokáže spracovávať dáta v reálnom čase a ovplyvňovať práve sa transformujúce hodnoty.
- Navrhneme postupy na redukciu rizika straty dát na jednotlivých výpočtových jednotkách.
- Navrhneme a implementujeme metódu na redukciu duplicít záznamov a experimentálne ju overíme
- Navrhneme metódu na automatické prispôsobovanie spracovania veľkého množstva dát, ktoré prichádzajú do systému.
- Vytvoríme modul na zachytávanie a mapovanie referenčných integrít dát medzi relačnými a nerelačnými databázami pri transformačných procesoch.
- Navrhneme a implementujeme rôzne varianty, ktoré zefektívnia proces vyhľadávania, ktoré nám pomôžu udržiavať štruktúru údajov a umožnia nám definovať indexy nad neúplnými dátami.

4. Riešenie paralelizmu

Veľkosť dát, ktoré prichádzajú do systému, nie je v súčasnosti ľahké odhadnúť. Pri veľkom počte prichádzajúcich dát, ktoré spracovávame distribuovane vstupujú do systému dáta vo veľmi malom objeme o veľkosti niekoľkých megabitov až po dáta o veľkosti rádovo gigabitov, terabitov, prípadne exabytov.

Na základe viacerých štúdií, ktoré sa zaoberali myšlienkou distribuovaného spracovania dát a paralelizmu, pri ktorej výskumníci zohľadňujú paralelizmus pri veľkom dôraze nákladov [35] a ochrane údajov [25], sa viacerí výskumníci obracajú na konštruovanie a správu paralelizmu na veľké cloudové spoločnosti ako je Google [29].

V mnohých prípadoch je spomenuté riešenie správne, avšak mechanizmus posudzovania paralelných procesov nemusí vyhovovať všetkým aplikáciám. Viaceré aplikácie nie sú závislé iba na počte dát, ale aj na vyťažení procesora, vyťažení vlákien vykonávajúcich výpočtové operácie a podobne.



Obrázok 1. Prispôsobovanie množstva vlákien veľkosti dát

Na účely experimentov sme vytvorili 3 súbory s rôznymi veľkosťami súborov a rôznymi objemami súborov. Vytvorené súbory obsahujú 1 000, 50 000 a 100 000 záznamov.

Ako je vidieť z obr. 9, hodnoty ktoré sme namerali pri veľkosti 1 000 záznamov ukazujú, že automatické škálovanie nebolo potrebné pre tieto účely, z dôvodu nízkej kapacity záznamov. Nami nastavená hodnota vyťaženia servera bola stanovená na hodnotu 80 percent, čo nebolo pri vstupnom toku dát prekročené, a preto nenastal žiadny nárast počtu výpočtových jednotiek. Okrem spomenutého faktu, hodnota 1 je vždy hodnota, ktorá predstavuje minimálny počet výpočtových jednotiek, ktorými server pri začiatku a konci procesu figuruje[RC3].

Pri porovnaní hodnôt, ktoré je možné pozorovať na obr. 9 nám s nárastom dát na hodnotu 50 000 záznamov narástol počet výpočtových jednotiek. Ako bolo spomínané vyššie v práci, tak vyťaženosť výpočtovej jednotky pri spracovaní 50 000 záznamov prekročila hranicu 80 percent, a tým spôsobila nárast počtu výpočtových jednotiek. Keďže ani počet výpočtových jednotiek rovnajúci sa hodnote 2 nestačil na zníženie vyťaženia servera, tak znovu bola pridaná ďalšia výpočtová jednotka. Pri počte rovnajúcej sa hodnote 3 sa vyťaženosť servera dostala do úrovne, kedy nebolo potrebné ani pridať a ani odobrať jednotky a server s optimálnym vyťažením dokázal spracovávať prichádzajúce dáta. Po spracovaní približne 2/3 záznamov došlo k opačnému efektu a množstvo výpočtových jednotiek potrebných na spracovanie údajov začal klesať až pokiaľ nebolo potrebné na spracovanie dát iba 1 jednotka. Tým pádom sa neoptimalizovala iba vyťaženosť servera, ale aj optimalizácia nákladov spojených s prevádzkovými nákladmi na spracovanie údajov.

5. Riešenie spoľahlivosti systému

Replikácia dát sa často používa ako prostriedok na zvýšenie dostupnosti dát úložného systému, ako je napríklad súborový systém Google. Ak na rôznych dátových uzloch existuje viac kópií bloku, kanály najmenej jednej prístupnej kópie sa zvýšia. Ak jeden dátový uzol zlyhá, dáta sú stále k dispozícii z replík. Náklady na správu sa výrazne zvýšia s rastúcim počtom replík. Mnoho replík nemusí výrazne vylepšiť dostupnosť [19], ale namiesto toho prinesie zbytočné výdavky. Kľúčovou otázkou je, aké je ideálne množstvo replík, ktoré zabezpečí optimálne fungovanie systému pri výpadku výpočtových uzlov.

Keď ponecháme všetky repliky aktívne, môžu sa ďalšie kópie použiť nielen na zlepšenie dostupnosti, ale aj na zlepšenie vyváženia záťaže a celkového výkonu, ak sú repliky primerane distribuované. Pri stanovení umiestnenia repliky musíme brať do úvahy aj veľkosť systému a počet výpočtových uzlov, na ktorom sa budú vykonávať výpočtové operácie.

Pravé takýmito otázkami sa zaoberalo viacero výskumníkov vo svojich štúdiách [RC5].

Na základe nášho úsudku sme vytvorili princíp, kedy stanovenie replikačného koeficientu určujeme na základe vyťaženie jednotlivých tabuliek. Replikačný koeficient sa musí odvíjať práve od reálnych hodnôt, ktoré sme získali z databázy. Podľa hodnoty celkového čítania záznamov sme vytvorili pravidlo, ktoré nám určuje, koľkokrát musia byť dáta, s ktorými pracujeme replikovať, aby nedošlo k potenciálnej strate záznamov.

Na základe odporúčaných literatúr [113], kde je uvádzaný replikačný koeficient na hodnotu 3 sme usúdili, že hodnota 3 je pre kľúčové hodnoty naozaj opodstatnená, avšak na základe nášho navrhnutého algoritmu, ktorý sleduje vyťaženosť hodnôt dochádza k stanoveniu replikačného koeficientu dát nasledovne:

- Algoritmus rozdelí záznamy podľa počtu čítania a zápisu do tabuľky.
    - Pre zápis sme definovali váhový koeficient rovný 1 z dôvodu, že pri zápise je v mnohých prípadoch v pozadí oproti operácii čítania dát do databázy.
    - Pre čítanie sme stanovili váhový koeficient rovný 2 z dôvodu, že čítanie je v mnohých prípadoch častejšia operácia ako operácia zápisu dát do databázy.

- Následne sa vypočíta medián záznamov na základe všetkých tabuliek pomocou vzorca:
- Algoritmus prechádza hodnotami získanými z databázy a vykonáva nasledujúcu operáciu:
    - ak operácia pre prvú tabuľku je menšia ako x tak sa hodnota replikačného koeficientu nastaví na hodnotu 2
    - v inom prípade sa replikačný koeficient nastaví na hodnotu 3
- Následne algoritmus vykoná krok vyššie pre všetky tabuľky a výsledkom bude rôznorodý replikačný koeficient pre nami poskytnutú dátovú schému.

6. Zvýšenie paralelizmu pri migrácii dát

Nakoľko operácia presunu údajov z databázy do distribuovaného systému trvá určitý čas a databáza stále plní úlohu primárneho úložiska, musíme zobrať do úvahy aj dáta, ktoré sa počas tohto procesu vyskytnú tzv. „dáta v reálnom čase“.

Na základe spomenutých nedostatkov a prichádzajúcich dát sme si stanovil pri skúmaní tejto problematiky nasledujúce ciele:

- Redukovať sekvenčné spracovanie záznamov na minimálnu hodnotu
- Aplikovať paralelný proces pri presune záznamov
- Redukovať množstvo dodatočných úprav v distribuovanom systéme
- V čo najväčšej miere ovplyvňovať práve sa transformujúce údaje, údajmi v reálnom čase

Nakoľko ovplyvňovanie jednotlivých procesov je ovplyvnené vzájomnými vzťahmi museli sme pri procese skontrolovať vzájomné vzťahy medzi tabuľkami navzájom a zistiť ktoré tabuľku a ich údaje môžu do procesu vstúpiť súčasne. Na základe počtu primárnych a cudzích kľúčov sme zostavili ich početnosť. Na základe hodnoty početnosti sme do procesu spúšťali tabuľky a údaje, ktoré mali najmenšiu hodnotu. Pri spustenom procese sme zachytávali hodnoty v reálnom čase pomocou transformačnej bariéry[RC2].

Hlavným cieľom transformačnej bariéry je sledovanie dvoch typov údajov. Prvým typom sú údaje, ktoré sú v systéme už dlho. Údaje sú uložené v databáze MySql. Počas tohto procesu ich môžu ovplyvňovať hodnoty, ktoré vstupujú do systému v čase zmeny alebo prenosu hodnôt z jedného typu databázy do druhého. Počas tohto procesu môžu existovať určité

kolízie, ktoré by bolo potrebné vykonať po transformačnom procese, čo by negatívne ovplyvnilo čas potrebný na transformáciu hodnôt. Kolízia je situácia, kedy dáta, ktoré sa práve transformujú sú ovplyvňované hodnotami alebo príkazmi, ktoré práve transformované dáta ovplyvňujú. Môžu aktualizovať práve transformované dáta, môžu vymazávať konkrétnu hodnotu, ba dokonca môžu vymazať celú tabuľku. Kolízia spôsobuje rýchlejšiu transformáciu, pretože môže redukovať vykonanie operácií pri transformačnom procese. Synchronizačná bariéra rieši vplyv údajov akumulovaných v systéme na hodnoty systému.

7. Vytváranie verzií dát počas migračných procesov

Nakoľko distribuované spracovanie slúži v mnohých prípadoch na správu veľkých dát (Big Data) je pre nás nevyhnutné, aby sa proces v prípade poruchy alebo výpadku systému vrátil do bodu zlyhania.

Na základe spomenutých nedostatkov a možnosti výpadku systému sme si stanovili pri skúmaní tejto problematiky nasledujúce ciele:

- Redukovať opätovné spustenie toho istého procesu v prípade zlyhania
- Aplikovať vytváranie verzií dát

Hlavnou myšlienkou je vytvorenie verzionovacieho systému schopného efektívneho návratu do bodu možnej chyby alebo práca s rôznymi dátovými verziami uloženými v nerelačnej databáze MongoDB.

Proces transformácie, pri ktorom zmena údajov z relačnej databázy MySql na MongoDB s väčším počtom tabuliek a údajov, môže trvať niekoľko minút až niekoľko hodín. V prípade chyby, ktorá sa vyskytne počas procesu transformácie, mohlo byť potrebné vymazať všetky hodnoty z nerelačnej databázy a podprogramov výhod a následne spustiť tento proces. Vyvinuli sme riešenie schopné pracovať podobným spôsobom s verziovanými systémami Github, BitBucket alebo Gitlab[RC7]. Spomenuté systémy pracujú na porovnaní konkrétnych verzií zdrojového kódu s lokálnym úložiskom umiestneným v počítači a úložiskom na serveri obsahujúcom zdrojový kód.

Pokiaľ ide o túto logiku, vytvorili sme bránu „verzie“. Brána funguje na nasledujúcom princípe. Algoritmus zisťuje, či sú v zdrojovej nerelačnej databáze určité špecifické hodnoty. V prípade prázdnej databázy sa hodnoty ukladajú automaticky a nie je potrebné vykonať nijakú opravu údajov.

V opačnom prípade je potrebné sledovať hodnoty objektov už umiestnených v databáze. Je užitočné sledovať hodnoty objektov v kolekcii, ale hodnoty objektov je možné priebežne meniť, mazať jednotlivé objekty alebo pridávať ďalšie atribúty. Na základe týchto problémov sme použili The Document Versioning Pattern priamo v nerelačnej databáze MongoDB 4.

8. Bezpečnosť spracovania dát v reálnom čase

Bezpečnosť uložených, respektíve poskytnutých údajov je v súčasnosti veľmi rozšírený a dôležitý aspekt pri manipulácii s dátami. V súčasnosti sa každý deň uskutočňuje množstvo útokov v snahe získať, respektíve odcudziť dáta, a tak negatívnym spôsobom ovplyvniť správne fungovanie aplikácie alebo databázy.

So zanedbaním bezpečnosti, respektíve s poľavením bezpečnosti pri správe údajov bolo už publikovaných mnoho článkov od viacerých výskumníkov. Pri tomto procese si viacerí výskumníci uvedomili, že so zvýšením bezpečnosti narastá aj dĺžka manipulácie s údajmi. Za zaujímavý aspekt sme považovali spravovať dáta, a následne tým redukovať stratu údajov pri rôznych útokoch alebo poškodení dát [RC6].

Na základe štúdie [65], ktorej budúci výskum chceli autori venovať návrhu presunu dát sme sa rozhodli aj my vytvoriť metódu presúvania dát z databázy do dátového skladu, a v prípade potreby z dátového skladu do databázy na základe pravidiel. Tento presun sme podmienili dodatočným bezpečnostným prvkom.

Hodnoty, ktoré sme zaznamenali, sme vkladali do relačnej tabuľky MySql. Aj keď sme skúšali aj nerelačnú databázu DynamoDB na základe pevnej štruktúry sme sa rozhodli použiť práve relačnú databázu MySql.

Hodnoty, ktoré nám prichádzajú do tabuliek sú povolené iba pre konkrétnu aplikáciu, ktorá vykonáva monitorovanie cesty jednotlivých dopravných prostriedkov, v našom prípade sa jedná o automobily. Za týmto účelom sme vytvorili rolu, konkrétne IAM Role, ktorá povoľuje vkladať záznamy iba aplikácii [RC8].

Hlavnou myšlienkou rozšírenia tabuľky bola kontrola stavu operácií jednotlivých záznamov. Máme na mysli operáciu aktualizovania (*update*) a zmazania (*delete*) záznamov. Za týmto účelom sme vytvorili nad databázou notifikáciu. Vždy chceme, aby každá operácia update a delete bola kontrolovaná pred vykonaním jednotlivých operácií. Ak aplikácia vykonáva spomenuté operácie update a delete, tak sa najskôr skontroluje ich *úloha/rola*, ktorá je priradená do aplikácie. Následne je skontrolovaná unikátna hodnota z tabuľky *autentifikátor*

(authentificator). Každá úloha (rola) ma vytvorených 5 náhodných token-ov, ktorých hlavička sa posiela automaticky pri zavolaní operácie update a delete. Týchto 5 náhodných vytvorených token-ov je závislých aj na čase vytvorenia záznamu a dochádza k ich automatickej úprave každých 12 hodín. Úprava token-u funguje dovtedy, pokiaľ nedôjde k neoprávnenému pokusu o úpravu dát. Ak sa takáto situácia vyskytne, tak sú tokeny pregenerované automaticky a je vyvolaná zmena na vytvorenie novej role pre aplikáciu. Upravenie notifikácie je druhou samozrejmou vecou, ktorá je vykonaná automaticky pri neoprávnenej snahe o získanie dát.

9. Ovplyvňovanie nahromadených dát reálnymi dátami

Transformačný proces, v ktorom dochádza k zmene štruktúry zo zdrojovej databázy na cieľovú databázu v mnohých prípadoch trvá od niekoľkých minút až po desiatky hodín. Počas tohto procesu môžu prichádzajúce dáta výrazne ovplyvniť práve sa transformujúce hodnoty, a tým výrazne redukovať čas potrebný na dodatočnú úpravu po skončení transformačného procesu. Tento fakt bol povšimnutý výskumníkmi [109] [24], ktorí navrhli efektívne metódy ako vyriešiť problém s reálnymi dátami. Podľa nášho názoru je implementované riešenie prvým krokom ako správne zohľadniť prichádzajúce dáta, a tým ovplyvniť konečný výsledok v cieľovej databáze, ale myslíme si, že k spomenutému ovplyvneniu dochádza až príliš neskoro [RC8].

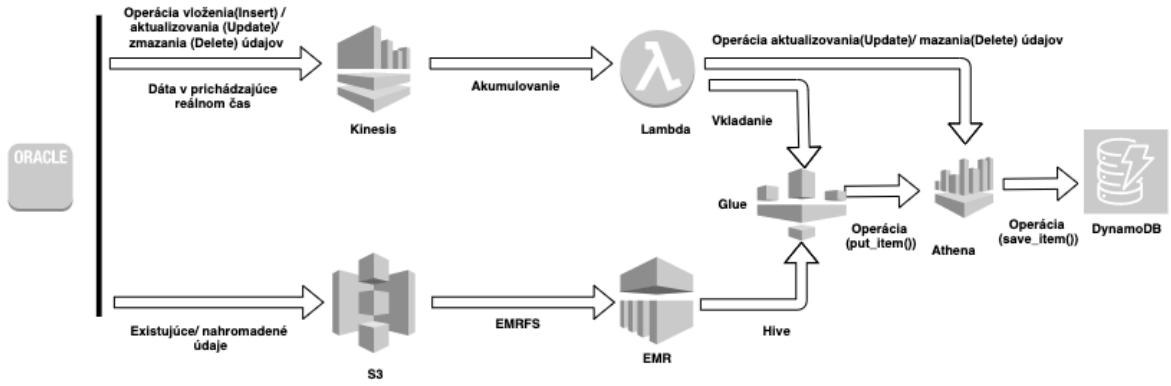
Na základe nášho úsudku, sme sa rozhodli posunúť ovplyvňovanie dát namiesto úplného konca procesu, to znamená pred uložením do databázy, do času, keď sa nahromadené dáta aktuálne transformujú a môžu byť ovplyvnené prichádzajúcimi dátami. Náš navrhnutý princíp, ktorý je popísaný nižšie úplne neodstránil metódu, ktorá bola spomenutá, ale výrazne ju redukoval[RC1]. Náš prínos je zameraný na situáciu, ktorá vychádza zo zanedbania zachytávania dát v reálnom čase pri migrácií dát z relačnej databázy na nerelačnú. V tomto procese zohrávajú dáta, ktoré do procesu prichádzajú počas prebiehajúcej transformácie dát významnú úlohu, a to nie len z dôvodu možnosti ovplyvňovania dát, ktoré sú nahromadené a už dochádza k ich transformácii, ale najmä z dôvodu nutnej úpravy dát v cieľovej databáze. Ako bolo publikované [20] operácia vyhľadávania nie je v nerelačných databázach rovnako efektívna ako pri databázach relačných.

Zo spomenutého dôvodu sme vytvorili architektúru, ktorá umožňuje pri transformačnom procese zachytávať dáta v reálnom čase, a tým zamedziť dodatočnej úprave po presune dát z relačnej databázy Oracle na nerelačnú databázu DynamoDB 6.

Táto architektúra sa skladá z 2 častí:

- *Vrchná vetva* (Dáta v reálnom čase)
- *Spodná vetva* (Nahromadené dáta)

Dáta v reálnom čase sú dáta, ktoré do systému vstúpili počas spusteného transformačného procesu, to znamená, že sa do databázy neukladajú, ale sú automaticky presúvané do transformačného procesu. Dáta, ktoré pôsobili v databáze Oracle dlhší čas, napríklad niekoľko dní, mesiacov alebo rokov, nazývame nahromadené dáta. Tieto dáta je nutné počas zmeny podrobiť určitému typu zmeny dát. Dáta mali štruktúru, respektíve boli uložené v relačnej databáze a musia byť z dôvodu efektívnosti zmenené, aby došlo k ich efektívnej manipulácii a spomenutý formát vyhovoval databázovému typu, pre ktorý je transformačný proces určený, keďže existujú väčšie množstvá nerelačných databáz.



Obrázok 2. Architektúra ovplyvňovania údajov

10. Porovnanie rýchlosti základných príkazov medzi relačnými a nerelačnými databázami

Pri prvotnom skúmaní sme potrebovali získať komplexnejší pohľad k operáciám z relačných/nerelačných databáz výberu dát (*select/find()*), vloženia záznamov (*insert/insert()*), aktualizovaniu hodnôt (*update/update()*) a vymazaniu záznamu (*delete/remove()*).

S nutnosťou preskúmania viacerých relačných a nerelačných databáz sme potrebovali získať hodnoty, ktoré budú otestované a namerané na rovnakých typoch databáz, v rovnakom počítači, respektíve serveri a pri rovnakých konfiguračných nastaveniach, aby nedošlo k žiadnemu skresleniu hodnôt a my sme mohli vyvodiť potrebné závery.

K aktuálnej problematike sa vyjadrilo viacero odborníkov a publikovali mnohé štúdie [71] [84], avšak mnohé z publikovaných štúdií sa venovali porovnaniu iba relačných, nerelačných, prípadne porovnávali jednu relačnú a nerelačnú databázu.

Z dôvodu komplexnosti riešenia sme získali dáta o dopravných spojeniach, ktoré sme následne vložili do viacerých relačných a nerelačných databáz a otestovali sme spomenuté operácie vyššie [RC4].

V snahe získať komplexnejšie výsledky sme museli zistiť trvanie pre jednotlivé operácie akými sú operácie vloženia (*insert*), aktualizovania (*update*), zmazania (*delete*) a výberu (*select*) údajov. Aj keď sme pri skúmaní našli viaceré výsledky spomenutých operácií nikdy sme nenašli výsledky, ktoré by porovnávali nami vyžadované databázové typy alebo boli porovnávania vykonané pri rôznej konfigurácií servera.

Z toho dôvodu sme sa rozhodli nakonfigurovať všetky databázové typy, ktoré sme uviedli v hlavičke tabuľky 12. Všetky tieto databázy boli nakonfigurované na rovnakú a defaultnú hodnotu, ktorá je odporúčaná na oficiálnych stránkach každej zmienej databázy. Aby sme zamedzili istým výkyvom, resp. nežiadúcim skresleniam, vykonávali sme príkazy iba pri spustení 1 programu. Všetky experimenty sme uskutočnili na počítači MacBook pro rok 2015 s 8 GB RAM a procesorom I5.

11. Metóda na zefektívnenie vyhľadávania v NoSQL databázach

Pri skúmaní danej problematiky sme sa zoznámili s možnosťou správy často dopytovaných dát a ich presunu do dát v pamäti akými sú databázy Redis alebo Memcached. Tieto databázy poskytujú účinný spôsob ako zefektívniť proces vyhľadávania v pamäti. Za nedostatok však môže byť považované ich pamäťové ohraničenie.

Ako nedostatok, ktorý sme počas výskumu zaznamenali je neefektívna manipulácia s dátami, ktoré boli spracované pomocou rôznych metód a ktoré boli publikované v článku [49] a následne presunuté z primárnej databázy do pamäte.

Na základe spomenutého nedostatku manažovania dát, kedy sú dáta presúvané z databázy do vyrovnávacej pamäte (cache), sme vytvorili spôsob, akým manažujeme nie len hodnoty čítania dát z databázy, ale manipulujeme resp. presúvame hodnoty do medzipamäte už pri zápise hodnôt, a tým redukujeme čas potrebný na opätovné načítanie záznamu hneď po jeho zápise.

Predstavili sme dva moduly, ktoré nám pomáhajú zefektívniť vyhľadávanie v nerelačných databázach.

Modul s medzipamäťou s údajmi a modul s elastickými údajmi. DCM (Data Cached Module) slúži ako sklad dátových úložísk, ktorého úlohou je prenos dát do medzipamäte. DEM (Data Elastic Module) slúži na automatické prispôsobenie veľkosti dát.

Pre DCM model sme implementovali :

- *Side-cache* - tento princíp nám pomáha pri vysokom preťažení počas čítania informácií z pamäte.
- *Medzipamäť na čítanie* - DAX(*Amazon DynamoDB Accelerator*) je vyrovnávacia pamäť na čítanie
- *Medzipamäť na zápis* - Podobne ako semi-pamäť na čítanie, aj semi-pamäť na zápis dát pracuje v súlade s databázou a aktualizuje semi-pamäť, keď sa dáta zapisujú do pamäte základných údajov. DEM:

Vytvorený modul slúži na zväčšovanie údajov v medzipamäti. Nerelačná databáza DynamoDB plní úlohu širokého úložiska dát a v prípade prenosu dát do databázy v pamäti sa dáta môžu pohybovať od niekoľkých megabajtov do gigabajtov. Tento spomínaný problém momentálne rieši vytvorený modul.

Monitorovanie metrík sme nakonfigurovali v cloudovej službe Amazon pomocou služby Amazon CloudWatch. Uvedená služba umožňuje editovať, pridávať a odstraňovať nové výpočtové jednotky v prípade horizontálneho alebo vertikálneho upravovania. Výhodnou charakteristikou tejto metódy je horizontálne škálovanie, ktoré v prípade veľkého počtu dátových prenosov vyvolá varovanie pred veľkým preťažením a skript na načítanie informácií z iných replík v službe CloudWatch. Horizontálna replika je časť skriptu vykonávaná automaticky počas konfigurácie databázy v pamäti DAX nasledujúcim skriptom.

12. Metóda na zrýchlenie vyhľadávania dát v NoSQL databáze MongoDB

Ako hlavný problém, ktorý sme pri skúmaní problematiky vyhodnotili za kľúčovú zložku je dátová štruktúra. Pri relačných databázach je tento problém vyriešený s kontrolou dátovej schémy pri vkladaní záznamov, ale tento aspekt nie je v nerelačných databázach implementovaný pre ich kľúčovú vlastnosť (flexibilnosť).

S neudržiavaním dátovej štruktúry súvisia viaceré problémy, akými sú dopyt dát (keďže nevieme, aké dáta a či nejaké dáta môžeme očakávať), kontrola dátových typov (získame pri

dopyte reťazec alebo číslo), prípadne vytvorenie sekundárnych indexov (nachádzajú sa vo všetkých objektoch rovnaké dáta, sú prázdne alebo nie).

Viacero výskumníkov prinieslo efektívne spôsoby [73] ako spoľahlivo pracovať s flexibilnými a heterogénnymi dátami v nerelačných databázach ako je MongoDB alebo DynamoDB. Vo viacerých prípadoch naozaj nie je nutné kontrolovať tieto hodnoty, čo redukuje rýchlosť vyhľadávania. Z dôvodu zrýchlenia získavania dát sme vytvorili spôsob, akým tento proces dokážeme zrýchliť, a taktiež udržať pevnú dátovú štruktúru.

Hlavným dôvodom viacerých výskumníkov zaoberajúcich sa nerelačnými databázami je degradácia vyhľadávania v nerelačných databázach. My sme sa rozhodli tento problém riešiť odlišným spôsobom, ktorý je založený na 2 databázach. Prvým typom databázy je nerelačná databáza MongoDB, ktorá bude slúžiť ako primárne dátové úložisko a druhým typom je relačná databáza Oracle, ktorá pre nás slúži ako nástroj na držanie istej štruktúry pre nerelačnú databázu. Z dôvodu voľnej štruktúry v nerelačnej databáze sme zaviedli v relačnej databáze dátový model (ktorý je zobrazený na obr. 46), ktorý nám poskytuje udržiavanie všetkých dátových typov na 1 mieste [RC9].

Problém, ktorý sa vyskytuje pri vyhľadávaní je spojený s nekompatibilitou príkazov medzi relačnými a nerelačnými databázami 9. Keď dôjde k situácii vkladania záznamov do nerelačnej databázy, tak pomocou skriptu sú názvy atribútov a ich dátové typu uložené do tabuľky *attribute* a názov štruktúry do tabuľky *structure*. Po úspešnom uložení hodnôt do nerelačnej databázy je vrátený záznam v konkrétnom bloku.

V prípade získavania hodnôt z nerelačnej databázy je príkaz na výber dát presunutý priamo do programu, kde dochádza k nasledujúcemu prístupu:

- príkaz je automaticky získaný a následne prekonvertovaný z nerelačného dotazu na dotaz relačný,
- príkaz získa objekt, prípadne objekty, ktoré vyhovujú kritériu a následne sú nám vrátené všetky atribúty *objectId*, ktoré spĺňajú kritérium,
- množstvo záznamov, ktoré získame z JSON formátu sa v nerelačných databázach často líši typom a aj množstvom atribútov, preto sme tento aspekt museli vyriešiť prehľadaním celého JSON-u ako je ukázané v 5 príkaze,
- tieto záznamy boli automaticky vložené do relačnej databázy za účelom vytvorenia stálej štruktúry. Pre účely jednoduchšej inkrementácie primárneho kľúča sme vytvorili 3 sekvencie,
- následne sú všetky prvky v nerelačnej databáze prehľadne vložené ako výstup.

13. Výsledky a vyhodnotenie

V tejto časti popíšeme metódy, ktoré sme počas doktorandského štúdia dosiahli, a taktiež ich ohraničenie.

1. ***Metóda synchronizačnej bariéry*** - Bariéra slúži na úpravu prichádzajúcich dát (real-time dáta) do systému počas toho, keď sa už začal proces s nahromadenými dátami, ktoré boli v systéme už dlhšiu dobu. Hlavnou úlohou synchronizačnej bariéry je znížiť počet úprav, ktoré by museli byť vykonané po skončení procesu. V každej situácii, kedy operácia príde do procesu, je vyvolaná bariéra, ktorá skontroluje, či operácia s dátami ovplyvní práve sa transformujúce dáta. Spomenutá metóda má výrazný vplyv na dĺžku transformačného procesu a významne redukuje počet dodatočných úprav dát v cieľovej databáze.
2. ***Verzionovacia metóda*** - Verzionovanie slúži na účely zvýšenia spoľahlivosti systému. V ľubovoľnom okamihu môže byť transformačný proces, ktorý spracováva veľké množstvo údajov prerušený, čo by viedlo k opätovnému spracovaniu údajov. Na základe verzionovania sme vytvorili proces, ktorý pri spustení zistí, či sa daný objekt alebo hodnota objektu už v procese vyskytla a v prípade, že už v procese naozaj bola, transformačný proces pomocou tejto metódy preskočí danú hodnotu objektu, čo ma za následok zrýchlenie procesu a zníženie vyťaženia servera.
3. ***Metóda paralelizmu výpočtu*** - Vytvorená metóda má za účel sa automaticky prispôsobovať zvýšenému respektíve zníženému dopytu na výpočtové jednotky. Na základe veľkosti dát, ktoré do systému prichádzajú a rýchlosti spracovania, sú vyvolané respektíve redukované paralelné procesy. Vytvorené paralelné procesy sú riadené hlavným vláknom, ktoré deleguje jednotlivé spracovanie. V prípade, ak sa jednotlivé procesy ovplyvňujú, tak sa hlavné vlákno postará o vytvorenie vlastného potomka, ktorý vstúpi do procesu ako mediátor procesu a vyrieši daný problém s kolíziou.
4. ***Metóda replikačného koeficientu*** - Vytvorená metóda slúži na posudzovanie replikačného koeficientu pre dáta, ktoré sú distribuovane spracovávané. Na základe prístupových hodnôt, ako je počet čítaní, počet zapisovaní do každej tabuľky, ktoré sme získali z databázy pomocou príkazu, vieme na základe pravidiel definovať potrebnosť resp. užitočnosť dát.

Z dôvodu neustálych zmien, čo sa týka čítania a zapisovania, sme vytvorili mechanizmus automatických úprav pre jednotlivé tabuľky na základe meniacich sa hodnôt či už sa jedná o masívne čítanie, prípadne zmazanie tabuľky.

5. ***Metóda redukcie duplicít*** - Vytvorená metóda slúži ako z názvu vyplýva z redukcie množstva prijatých hodnôt do systému. Z dôvodu akceptovania hodnôt z viacerých zdrojov sme boli nútení sledovať všetky prichádzajúce hodnoty do systému, a v prípade výskytu rovnakých dát vytvárať referencie na objekty respektíve hodnoty pre viaceré zdroje.
6. ***Metóda paralelizmu procesov*** - Presun dát je vždy spojený s istým čakaním. Z dôvodu závislosti na jednotlivých tabuľkách a vykonávania procesu sekvenčne sme vytvorili algoritmus, ktorý na začiatku zistí funkčnú závislosť na jednotlivých tabuľkách pri zistení závislosti primárnych, cudzích a kompozitných kľúčov a zistí, ktoré tabuľky môžu byť spracované paralelne. V tom prípade dochádza k spusteniu všetkých tabuliek paralelne a nie sekvenčne.
7. ***Metóda bezpečnosti práce s údajmi*** - Bezpečnosť dát patrí k dôležitej súčasti našej práce. Namiesto ukladania dát do viacerých zón dostupnosti sme sa rozhodli ukladať dáta do viacerých regiónov, čo nám redukuje možnosť straty údajov pri distribuovane spracovaných záznamoch. Taktiež sme implementovali spôsob, akým používateľ má možnosť pracovať s dátami a tým redukovali nežiaduci vstup aplikácie alebo používateľa k našim dátam.
8. ***Metóda manažmentu dát*** - Navrhnutá metóda má za účel efektívne spravovať aktívne a pasívne hodnoty, a tým redukovať množstvo dát uložených v primárnej databáze. Za týmto účelom sme vytvorili metódu, ktorá na základe časových hodnôt a dopytu k jednotlivým dátam presúva dáta obidvomi smermi, a to z databázy do dátového skladu alebo z dátového skladu do databázy. Tento manažment priniesol viaceré výhody akými sú menšie vyťaženie servera, rýchlejšie dopytovanie a menšia strata údajov, v prípade výpadku.
9. ***Metóda vytvorenia štruktúry dát*** - Hlavným účelom navhrnutej metódy je uchovávať pomocou štruktúry v relačnej databáze štruktúru objektov v nerelačných databázach. Na základe dopytovaných hodnôt, ktoré sa môžu v nerelačných metódach často odlišovať vieme redukovať množstvo hodnôt, ktoré spĺňajú kritérium na základe napísaného príkazu. Pre zrýchlenie tohto procesu sme vytvorili sekundárny index, ktorý nám v tomto ohľade pomôže ešte viacej zrýchliť proces vyhľadávania v relačnej databáze.

10. ***Metóda strojového učenia*** - Vytvorená metóda má za účel zrýchliť proces vyhľadávania v databázach. Prvotnou myšlienkou bolo zrýchliť vyhľadávanie v nerelačných databázach ako je *MongoDB* a *DynamoDB*, ale tento model sme implementovali aj pre relačné databázy *Oracle* a *MySql*. Algoritmus funguje na 2 princípoch. Prvým princípom je sledovanie príkazu, ktorý používateľ píše a predčasne spustí tento proces. Druhým princípom je predpokladať na základe strojového učenia aké príkazy boli spúšťané a výsledky presunúť do pamäte, a tým pádom príkaz nespustiť priamo nad spustenou databázou, ale nad databázou v pamäti.

14. Záver

V predkladanej práci sme riešili problémy, ktoré súvisia so stále narastajúcou potrebou rýchlejšej a efektívnejšej manipulácie s dátami, ktoré sú ovplyvnené dátami v reálnom čase. Pri tomto procese sme ukladali dáta do nerelačných databáz akými boli MongoDB a DynamoDB, s čím súvisí aj zvyšujúci nárok na rýchlosť získavania údajov.

Prvá časť práce je viac-menej teoretickou časťou. V práci sa venujeme štandardným (konvenčným) riešeniam, ktoré súvisia s viacerými výzvami, ktoré sme museli zohľadniť pri riešení distribuovane spracovaných dát a taktiež výzvy, ktoré sa týkali procesu vyhľadávania v nerelačných databázach. Zistili sme, že v mnohých prípadoch sú dáta, ktoré prichádzajú do systému počas transformačného procesu zanedbané a následne až po ukončení procesu je s dátami manipulované. Tým pádom dochádza k vyhľadávaniu dát a následnej úprave po skončení transformačného procesu, čo v konečnom dôsledku zvyšuje čas potrebný na celkovú úpravu. Ako ďalší skúmaný problém sme videli v neštandardizovanom spôsobe manipulácie s heterogénnymi dátami, ktoré pri vyhľadávaní spôsobujú značné problémy.

Analýzou existujúcich riešení, algoritmov a systémov sme zistili, že neexistujú komplexné riešenia, ktoré by dokázali na základe univerzálnych modulov riešiť viaceré výzvy pri distribuovanom spracovaní dát. Pri skúmaní riešení, ktoré sa venujú vyhľadávaniu dát, keďže práve nerelačné databázy slúžia aj na ukladanie rozsiahlych dát sme taktiež navrhli viaceré spôsoby, ktoré v konečnom dôsledku dokážu zrýchliť proces vyhľadávania v nerelačných databázach, ale dokonca aj v relačných databázach.

Kľúčovú zložku v našom výskume tvorí zvyšovanie paralelizmu spracovania údajov. Spomenutý aspekt bol súčasťou mnohých diskusných príspevkov a fór, kde sa výskumníci snažili odhaliť ideálnu hodnotu či už vyťaženia servera, počtu vlákien alebo množstva procesov, ktoré sú ideálne pre spracovanie veľkého množstva údajov. Na základe diskusií sme

vytvorili princíp automatického prispôsobovania výpočtových jednotiek pri zachovaní optimálnych podmienok pri správnom zaťažení servera a pri efektívnej správe nákladov.

Významný spôsob ovplyvňovania dát vidíme v návrhu synchronizačnej bariéry, ktorá dokáže efektívne vstupovať do procesu práve sa transformujúcich dát (transformujú sa dáta, ktoré boli v procese dlhšiu dobu) a tým pádom na základe operácie aktualizovania, prípadne zmazania údajov, dokáže urýchliť tento proces v časovom rozmedzí od niekoľko sekúnd až po desiatky minút. Pomocou tohto spôsobu sa na základe dát, ktoré vstúpili do systému môžu zmazať hodnoty od 1 záznamu až po hodnoty celej transformujúcej sa tabuľky až po hodnotu celej kolekcie.

Bezpečnosť pri transformačných procesoch je v súčasnosti veľmi žiadaná, a preto aj zachovanie bezpečnosti či už transformačného procesu (vyriešili sme vytvorením algoritmu na verzionovanie dát), bezpečnosť už uložených dát (vyriešili sme pomocou automatického presúvania údajov medzi databázou a dátovým skladom) alebo potlačením duplicít pri transformácii dát (vyriešili sme sledovaním vstupných hodnôt a vytváraním referencií) zohrali pri našom výskume dôležitú zložku.

Vyhľadávanie pomocou príkazu Select v relačných databázach, respektíve príkaz Find v nerelačných databázach považujeme taktiež za dôležitú súčasť nášho výskumu. Na základe väčšej flexibility ukladania záznamov v nerelačných databázach sme vytvorili viaceré metódy, ktorých účelom je získať dáta z dátového úložiska rýchlejšie, vytvoriť istý spôsob udržiavania pevnej štruktúry dát tak ako to poznáme z relačných databáz, a taktiež efektívnu manipuláciu s vyrovnávacou pamäťou. Ako veľký posun v tomto smere sme vytvorili modul pomocou strojového učenia, ktorý dokáže sledovať rozpísaný príkaz v relačnej databáze Oracle a nerelačných databázach MongoDB a DynamoDB a ešte pred potvrdením príkazu predčasne presunúť dáta do vyrovnávacej pamäti, čo nám v mnohých prípadoch urýchlilo proces získavania záznamov.

Počas transformačného procesu, kedy dochádzalo k presunu údajov z relačného typu databáz (pomocou stanovených pravidiel) do nerelačného úložiska bol proces oproti opačnému spôsobu relatívne jednoduchší ako pri presune dát opačným smerom. Tento problém súvisel s voľnosťou štruktúry v nerelačných databázach. Pri tomto presune sme museli efektívnym spôsobom riešiť aj problém súvisiaci s nekompatibilnosťou dátových typov a hlavne neúplnosťou vyžadovaných údajov a taktiež referenčnou integritou.

Sumarizácia dosiahnutých cieľov:

- Navrhli sme architektúru, ktorá dokáže v reálnom čase zachytávať prichádzajúce dáta, a tým ovplyvniť práve sa transformujúce údaje.
- Navrhli sme postup ako efektívne manipulovať s výkonnosťou systému, a tým efektívne sa prispôsobovať vyťaženiu servera.
- Navrhli sme metódu na automatické prispôsobovanie počtu replikácií v distribuovanom systéme na základe databázových procesov.
- Vytvorili sme metódu na redukciu duplicít v reálnom čase, a tým sme redukovali aj množstvo úložného priestoru.
- Vytvorili sme architektúru, ktorá dokáže efektívne pracovať s meniacou sa dátovou štruktúrou v nerelačných databázach.
- Vytvorili sme princíp a spôsob manažovania hodnôt z nerelačných databáz MongoDB a DynamoDB pomocou vyrovnávacej pamäte.
- Navrhli a implementovali sme metódu kontroly a správy manažmentu referenčných integrít pri transformačnom procese.
- Navrhli sme mechanizmus strojového učenia, ktorý na základe predchádzajúcich dopytov efektívne manažuje presun dát z relačných databáz Oracle a MySql a nerelačných databáz MongoDB a DynamoDB.
- Experimentálne sme overili výkonnosti vytvorených riešení a postupov na malých (1GB dát), stredných (100 GB dát), veľkých (1 TB dát) a veľmi veľkých (100 TB dát) vzorkách dát.

Vytvorené riešenia boli testované a použité pre prostredie v rozsiahlych dopravných prostriedkoch, avšak poskytnuté riešenia nie sú pre tento proces od nich závislé, keďže sa jedná o všeobecné algoritmy nezávislé na vstupných dátach alebo objemu dát.

Zoznam vlastných publikácií

V texte budeme používať skratky RC1 až RC10 ktoré označujú vlastné publikácie.

[1]. AFC
Influencing migration processes by real-time data / Roman Ceresnak, Karol Matiasko, Adam Dudas.
In: Proceedings of the 28th conference of Open innovations association FRUCT [electronic]. - ISSN 2305-7254. - 1. vyd. - [S.l.]: FRUCT Oy, 2021. - ISBN 978-952-69244-4-1 (online). - s. 48-54.
[Čerešňák Roman - Matiaško Karol - Dudáš Adam]

[2]. AFD
Comparison of distributed data transformation and comparing query performance in relational and non-relational database / Roman Čerešňák, Michal Kvet.
In: ICETA 2019 [electronic, print] : 17th IEEE International conference on emerging elearning technologies and applications : Information and communication technologies in learning : proceedings. - 1. vyd. - Denver: Institute of Electrical and Electronics Engineers, 2019. - ISBN 978-1-7281-4967-7. - s. 108-114 [online].
[Čerešňák Roman - Kvet Michal]

[3]. AFD
Comparison of apache technology in distributed environment / Roman Ceresnak, Michal Kvet.
In: Information and digital technologies 2019 [electronic] : proceedings of the international conference. - 1. vyd. - Danvers: Institute of Electrical and Electronics Engineers, 2019. - ISBN 978-1-7281-1401-9. - s. 80-85 .
[Čerešňák Roman - Kvet Michal]

[4]. AFD
Comparison of query performance in relational a non-relation databases / Roman Čerešňák, Michal Kvet.
In: TRANSCOM 2019 [electronic] : conference proceedings. - ISSN 2352-1465. - 1. vyd. - Amsterdam: Elsevier Science, 2019. - s. 170-177
[Čerešňák Roman - Kvet Michal]

[5]. AFC
Using replication method to increase reliability in distributed information systems / Roman Čerešňák, Karol Matiaško - Conference Reliability Engineering and Computational Intelligence
[Čerešňák Roman - Matiaško Karol]

[6]. AFC
Synchronization Barrier in Database Migration Processes / Roman Čerešňák, Karol Matiaško - Conference ICETA 2021
[Čerešňák Roman - Matiaško Karol]

[7]. AFC
Versioning Data During Migration Processes in Cloud Environment / Roman Čerešňák, Karol Matiaško - Conference ICETA 2021
[Čerešňák Roman - Matiaško Karol]

[8]. AFC
Various proposed approaches eliminating duplicate data in a system / Roman Čerešňák, Karol Matiaško - Scientific journal Communications
[Čerešňák Roman - Matiaško Karol]

[9]. AFC
Improvement of Searching Data in MongoDB with the Usage of Oracle Database / Roman Čerešňák, Adam Dudáš - Conference SSD
[Čerešňák Roman - Dudáš Adam]

Zoznam použitej literatúry a ďalších použitých zdrojov

[1]. 2014. Proceedings of the ACM SIGMOD International Conference on Management of Data.

[2]. Abad, C., Lu, Y. and Campbell, R., 2011. DARE: Adaptive Data Replication for Efficient Cluster Scheduling. 2011 IEEE International Conference on Cluster Computing,.

[3]. Abawajy, J. and Deris, M., 2014. Data Replication Approach with Consistency Guarantee for Data Grid. IEEE Transactions on Computers, 63(12), pp.2975-2987.

[4]. Agrawal, S., Chaudhuri, S. and Narasayya, V., 2000. Automated selection of materialized views and indexes for SQL databases.

[5]. Agrawal, S., Chaudhuri, S. and Narasayya, V., 2001. Materialized view and index selection tool for Microsoft SQL server 2000. ACM SIGMOD Record, 30(2), p.608.

[6]. Ahanger, G. and Little, T., 2001. Data semantics for improving retrieval performance of digital news video systems. IEEE Transactions on Knowledge and Data Engineering, 13(3), pp.352-360.

[7]. Ameri, P., 2016. Challenges of index recommendation for databases [With specific evaluation on a NoSQL database].

[8]. Ananda, A. and Poo, G., 1995. Distributed systems: Concepts and design. Computer Communications, 18(7), pp.521-522.

[9]. Aouiche, K. and Darmont, J., 2009. Data mining-based materialized view and index selection in data warehouses. Journal of Intelligent Information Systems, 33(1), pp.65-93.

[10]. Barbierato, E., Gribaudo, M. and Iacono, M., 2014. Performance evaluation of NoSQL big-data applications using multi-formalism models. Future Generation Computer Systems, 37, pp.345-353.

[11]. Binani, S., Gutti, A. and Upadhyay, S., 2016. SQL vs. NoSQL vs. NewSQL- A Comparative Study. Communications on Applied Electronics, 6(1), pp.43-46.

[12]. Bjeladinovic, S., 2018. A fresh approach for hybrid SQL/NoSQL database design based on data structuredness. Enterprise Information Systems, 12(8-9), pp.1202-1220.

[13]. Board, R., 1993. Distributed Database Systems. IASSIST Quarterly, 16(3), p.4.

[14]. Boicea, A., Radulescu, F. and Agapin, L., 2012. MongoDB vs Oracle -- Database Comparison. 2012 Third International Conference on Emerging Intelligent Data and Web Technologies,.

[15]. Bruno, N. and Chaudhuri, S., 2005. Automatic physical database tuning. Proceedings of the 2005 ACM SIGMOD international conference on Management of data - SIGMOD '05,.

[16]. Callan, J., Lu, Z. and Croft, W., 2017. Searching Distributed Collections With Inference Networks. ACM SIGIR Forum, 51(2), pp.160-167.

[17]. Caprara, A., Fischetti, M. and Maio, D., 1995. Exact and approximate algorithms for the index selection problem in physical database design. IEEE Transactions on Knowledge and Data Engineering, 7(6), pp.955-967.

[18]. Casado, R. and Younas, M., 2014. Emerging trends and technologies in big data processing. Concurrency and Computation: Practice and Experience, 27(8), pp.2078-2091.

[19]. Celesti, A., Fazio, M., Romano, A., Bramanti, A., Bramanti, P. and Villari, M., 2018. An OAIS-Based Hospital Information System on the Cloud: Analysis of a NoSQL Column-Oriented Approach. IEEE Journal of Biomedical and Health Informatics, 22(3), pp.912-918.

[20]. Čerešňák, R. and Kvet, M., 2019. Comparison of query performance in relational a non-relation databases. Transportation Research Procedia, 40, pp.170-177.

[21]. Cervin, A., Eker, J., Bernhardsson, B. and Årzén, K., 2002. Feedback-feedforward scheduling of control tasks. Real-Time Systems, 23(1/2), pp.25-53.

[22]. Chen, F., Deng, P., Wan, J., Zhang, D., Vasilakos, A. and Rong, X., 2015. Data Mining for the Internet of Things: Literature Review and Challenges. International Journal of Distributed Sensor Networks, 11(8), p.431047.

[23]. Choice Reviews Online, 1997. The Computer science and engineering handbook. 35(04), pp.35-2165-35-2165.

[24]. Chung, W., Lin, H., Chen, S., Jiang, M. and Chung, Y., 2013. JackHare: a framework for SQL to NoSQL translation using MapReduce. Automated Software Engineering, 21(4), pp.489-508.

[25]. Cidon, A., Stutsman, R., Rumble, S., Katti, S., Ousterhout, J. and Rosenblum, M., 2013. MinCopysets : Derandomizing Replication In Cloud Storage.

[26]. Dayalan, M., 2018. MapReduce: Simplified Data Processing on Large Cluster. International Journal of Research and Engineering, 5(5), pp.399-403.

[27]. DB, S., 2021. SQL Vs Nosql Database Differences Explained With Few Example DB. [online] Thegeekstuff.com. Available at: <https://www.thegeekstuff.com/2014/01/sql-vs-nosql-db/> [Accessed 16 January 2021].

[28]. Deari, R., Zenuni, X., Ajdari, J., Ismaili, F. and Raufi, B., 2018. Analysis And Comparision of Document-Based Databases with Relational Databases: MongoDB vs MySQL. 2018 International Conference on Information Technologies (InfoTech),.
[29]. Díaz-Galiano, M., Martín-Valdivia, M., Montejo-Ráez, A. and Ureña-López, L., 2007. Improving Performance of Medical Images Retrieval by Combining Textual and Visual Information. 2007 Sixth Mexican International Conference on Artificial Intelligence, Special Session (MICAI),.
[30]. Díaz, M., Martín, C. and Rubio, B., 2016. State-of-the-art, challenges, and open issues in the integration of Internet of things and cloud computing. Journal of Network and Computer Applications, 67, pp.99-117.
[31]. Elmore, A., Das, S., Agrawal, D. and El Abbadi, A., 2021. Zephyr : Live Migration in Shared Nothing Databases for Elastic Cloud Platforms Categories and Subject Descriptors.
[32]. Erevelles, S., Fukawa, N. and Swayne, L., 2016. Big Data consumer analytics and the transformation of marketing. Journal of Business Research, 69(2), pp.897-904.
[33]. Faerber, F., Kemper, A., Larson, P., Levandoski, J., Neumann, T. and Pavlo, A., 2017. Main Memory Database Systems. Foundations and Trends® in Databases, 8(1-2), pp.1-130.
[34]. Fan, W. and Bifet, A., 2013. Mining big data. ACM SIGKDD Explorations Newsletter, 14(2), pp.1-5.
[35]. Fan, X., Weber, W. and Barroso, L., 2007. Power provisioning for a warehouse-sized computer. Proceedings of the 34th annual international symposium on Computer architecture - ISCA '07,.
[36]. Fox, P. and Friston, K., 2012. Distributed processing; distributed functions?. NeuroImage, 61(2), pp.407-426.
[37]. Freiknecht, J. and Papp, S., 2018. Apache Kafka. Big Data in der Praxis, pp.449-456.
[38]. Ghotiya, S., Mandal, J. and Kandasamy, S., 2017. Migration from relational to NoSQL database. IOP Conference Series: Materials Science and Engineering, 263, p.042055.
[39]. González-Aparicio, M., Ogunyadeka, A., Younas, M., Tuya, J. and Casado, R., 2017. Transaction processing in consistency-aware user's applications deployed on NoSQL databases. Human-centric Computing and Information Sciences, 7(1).
[40]. Goudarzi, M., 2019. Heterogeneous Architectures for Big Data Batch Processing in MapReduce Paradigm. IEEE Transactions on Big Data, 5(1), pp.18-33.
[41]. Govindan, S., Sivasubramaniam, A. and Urgaonkar, B., 2011. Benefits and limitations of tapping into stored energy for datacenters. Proceeding of the 38th annual international symposium on Computer architecture - ISCA '11,.
[42]. Grolinger, K., Hayes, M., Higashino, W., L'Heureux, A., Allison, D. and Capretz, M., 2014. Challenges for MapReduce in Big Data. 2014 IEEE World Congress on Services,.
[43]. Gu, L., Zeng, D., Guo, S. and Ye, B., 2015. Joint optimization of VM placement and request distribution for electricity cost cut in geo-distributed data centers. 2015 International Conference on Computing, Networking and Communications (ICNC),.
[44]. Guo, Y., Rao, J., Jiang, C. and Zhou, X., 2017. Moving Hadoop into the Cloud with Flexible Slot Management and Speculative Execution. IEEE Transactions on Parallel and Distributed Systems, 28(3), pp.798-812.
[45]. Hirzel, M., Soulé, R., Schneider, S., Gedik, B. and Grimm, R., 2014. A catalog of stream processing optimizations. ACM Computing Surveys, 46(4), pp.1-34.
[46]. Hueske, F. and Walther, T., 2019. Apache Flink. Encyclopedia of Big Data Technologies, pp.51-58.
[47]. IEEE Security & Privacy Magazine, 2011. Usenix 2011 Trade Advertisement. 9(1), pp.c2-c2.
[48]. Imran, S. and Hyder, I., 2009. Security Issues in Databases. 2009 Second International Conference on Future Information Technology and Management Engineering,.
[49]. Jacobs, I. and Bean, C., 1963. Fine Particles, Thin Films and Exchange Anisotropy (Effects of Finite Dimensions and Interfaces on the Basic Properties of Ferromagnets). Spin Arrangements and Crystal Structure, Domains, and Micromagnetics, pp.271-350.
[50]. Janech, J., Tavac, M. and Kvet, M., 2019. Versioned database storage using unitemporal relational database. 2019 IEEE 15th International Scientific Conference on Informatics,.
[51]. Jin, H., Cheocherngngarn, T., Levy, D., Smith, A., Pan, D., Liu, J. and Pissinou, N., 2013. Joint Host-Network Optimization for Energy-Efficient Data Center Networking. 2013 IEEE 27th International Symposium on Parallel and Distributed Processing,.
[52]. Kamal, J., Murshed, M. and Buyya, R., 2016. Workload-aware incremental repartitioning of shared-nothing distributed databases for scalable OLTP applications. Future Generation Computer Systems, 56, pp.421-435.
[53]. Kamburugamuve, S., Fox, G., Leake, D. and Qiu, J., 2013. Survey of distributed stream processing for large stream sources.

[54]. Karnitis, G. and Arnicans, G., 2015. Migration of Relational Database to Document-Oriented Database: Structure Denormalization and Data Transformation. 2015 7th International Conference on Computational Intelligence, Communication Systems and Networks,.
[55]. Katsifodimos, A. and Schelter, S., 2016. Apache Flink: Stream Analytics at Scale. 2016 IEEE International Conference on Cloud Engineering Workshop (IC2EW),.
[56]. Khan, O., Burns, R., Plank, J., Pierce, W. and Huang, C., 2019. Rethinking erasure codes for cloud file systems: Minimizing I/O for recovery and degraded reads.
[57]. Khare, A., 2016. A Review of NoSQL Databases , Types and Comparison with Relational Database.
[58]. Kleinrock, L., 1985. Distributed systems. Communications of the ACM, 28(11), pp.1200-1213.
[59]. Kleppmann, M., 2018. Samza. Encyclopedia of Big Data Technologies, pp.1-8.
[60]. Klettke, M., Störl, U. and Scherzinger, S., 2015. Schema extraction and structural outlier detection for JSON-based NoSQL Data Stores.
[61]. Kuderu, N. and Kumari, V., 2016. Relational Database to NoSQL Conversion by Schema Migration and Mapping. International Journal of Computer Engineering in Research Trends, 3(9), p.506.
[62]. Kulkarni, S., Bhagat, N., Fu, M., Kedigehalli, V., Kellogg, C., Mittal, S., Patel, J., Ramasamy, K. and Taneja, S., 2015. Twitter Heron. Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data,.
[63]. Kuznetsov, S. and Poskonin, A., 2014. NoSQL data management systems. Programming and Computer Software, 40(6), pp.323-332.
[64]. Kvet, M. and Matiasko, K., 2017. Time as the Important Factor of the DataRetrieval – Table Type Classification. Advances in Intelligent Systems and Computing, pp.492-502.
[65]. Kvet, M., 2019. Data Distribution in Ad-hoc Transport Network. 2019 International Conference on Information and Digital Technologies (IDT),.
[66]. Kvet, M., Matiaško, K. and Kvet, M., 2014. Complex time management in databases. Open Computer Science, 4(4).
[67]. Kvet, M., Toth, S. and Krsak, E., 2019. Concept of temporal data retrieval: Undefined value management. Concurrency and Computation: Practice and Experience, 32(13).
[68]. Le Merrer, E. and Le Scouarnec, N., 2016. Efficient User Opt-Out from Block Stores. 2016 IEEE International Conference on Cloud Engineering Workshop (IC2EW),.
[69]. Lei, H., Blount, M. and Tait, C., 1999. DataX: an approach to ubiquitous database access. Proceedings WMCSA'99. Second IEEE Workshop on Mobile Computing Systems and Applications,.
[70]. Li, C., 2010. Transforming relational database into HBase: A case study. 2010 IEEE International Conference on Software Engineering and Service Sciences,.
[71]. Li, S., Jiang, H. and Shi, M., 2017. Redis-based web server cluster session maintaining technology. 2017 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD),.
[72]. Liu, X., Iftikhar, N. and Xie, X., 2014. Survey of real-time processing systems for big data. Proceedings of the 18th International Database Engineering & Applications Symposium on - IDEAS '14,.
[73]. Liu, Y., Wang, Y. and Jin, Y., 2012. Research on the improvement of MongoDB Auto-Sharding in cloud environment. 2012 7th International Conference on Computer Science & Education (ICCSE),.
[74]. Liu, Z., Lin, M., Wierman, A., Low, S. and Andrew, L., 2015. Greening Geographical Load Balancing. IEEE/ACM Transactions on Networking, 23(2), pp.657-671.
[75]. Lourenço, J., Cabral, B., Carreiro, P., Vieira, M. and Bernardino, J., 2015. Choosing the right NoSQL database for the job: a quality attribute evaluation. Journal of Big Data, 2(1).
[76]. Manegold, S., 2002. Understanding, modeling, and improving main-memory database performance.
[77]. Marshall, I. and Roadknight, C., 1998. Linking cache performance to user behaviour. Computer Networks and ISDN Systems, 30(22-23), pp.2123-2130.
[78]. Mehmood, N., Culmone, R. and Mostarda, L., 2017. Modeling temporal aspects of sensor data for MongoDB NoSQL database. Journal of Big Data, 4(1).
[79]. Mo, X. and Wang, H., 2012. Asynchronous Index Strategy for high performance real-time big data stream storage. 2012 3rd IEEE International Conference on Network Infrastructure and Digital Content,.
[80]. Naheman, W. and Jianxin Wei, 2013. Review of NoSQL databases and performance testing on HBase. Proceedings 2013 International Conference on Mechatronic Sciences, Electric Engineering and Computer (MEC),.
[81]. Nasiri, H., Nasehi, S. and Goudarzi, M., 2019. Evaluation of distributed stream processing frameworks for IoT applications in Smart Cities. Journal of Big Data, 6(1).
[82]. Okman, L., Gal-Oz, N., Gonen, Y., Gudes, E. and Abramov, J., 2011. Security Issues in NoSQL Databases. 2011IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications,.
[83]. Oliver, A., 2014. Storm or Spark: Choose your real-time weapon.

[84]. Pala, I., 2014. BMC Medical Informatics and Decision Making. BMC Medical Informatics and Decision Making, 14(1).
[85]. Patel, J., 2015. From Data to Insights @ Bare Metal Speed. Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data,.
[86]. Peng, D. and Dabek, F., 2019. Large-scale incremental processing using distributed transactions and notifications.
[87]. Qader, M., Cheng, S. and Hristidis, V., 2018. A Comparative Study of Secondary Indexing Techniques in LSM-based NoSQL Databases. Proceedings of the 2018 International Conference on Management of Data,.
[88]. Qian, L., Luo, Z., Du, Y. and Guo, L., 2009. Cloud Computing: An Overview BT - Cloud Computing. pp.626-631.
[89]. Qureshi, A., Weber, R., Balakrishnan, H., Guttag, J. and Maggs, B., 2009. Cutting the electric bill for internet-scale systems. ACM SIGCOMM Computer Communication Review, 39(4), pp.123-134.
[90]. Rawat, D. and Alshaikhi, A., 2018. Leveraging Distributed Blockchain-based Scheme for Wireless Network Virtualization with Security and QoS Constraints. 2018 International Conference on Computing, Networking and Communications (ICNC),.
[91]. Reniers, V., Rafique, A., Van Landuyt, D. and Joosen, W., 2017. Object-NoSQL Database Mappers: a benchmark study on the performance overhead. Journal of Internet Services and Applications, 8(1).
[92]. Sagiroglu, S. and Sinanc, D., 2013. Big data: A review. 2013 International Conference on Collaboration Technologies and Systems (CTS),.
[93]. Scavuzzo, M., Di Nitto, E. and Ceri, S., 2014. Interoperable Data Migration between NoSQL Columnar Databases. 2014 IEEE 18th International Enterprise Distributed Object Computing Conference Workshops and Demonstrations,.
[94]. Schneider, S., Hirzel, M., Gedik, B. and Wu, K., 2012. Auto-parallelizing stateful distributed streaming applications. Proceedings of the 21st international conference on Parallel architectures and compilation techniques - PACT '12,.
[95]. Sharma, K. and Attar, V., 2016. Generalized Big Data Test Framework for ETL migration. 2016 International Conference on Computing, Analytics and Security Trends (CAST),.
[96]. Shidong Huang, Lizhi Cai, Zhenyu Liu and Yun Hu, 2012. Non-structure Data Storage Technology: A Discussion. 2012 IEEE/ACIS 11th International Conference on Computer and Information Science,.
[97]. Silva, B., Diyan, M. and Han, K., 2018. Big Data Analytics. Deep Learning: Convergence to Big Data Analytics, pp.13-30.
[98]. Smith, G., 1991. Modeling security-relevant data semantics. IEEE Transactions on Software Engineering, 17(11), pp.1195-1203.
[99]. Srinivas, S. and Nair, A., 2015. Security maturity in NoSQL databases - are they secure enough to haul the modern IT applications?. 2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI),.
[100]. Tan, Y., Ko, R. and Holmes, G., 2013. Security and Data Accountability in Distributed Systems: A Provenance Survey. 2013 IEEE 10th International Conference on High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing,.
[101]. Tidke, S., 2017. MonogDB: Data management in NoSQL. Privacy and Security Policies in Big Data, pp.64-91.
[102]. Toshniwal, A., Taneja, S., Shukla, A., Ramasamy, K., Patel, J., Kulkarni, S., Jackson, J., Gade, K., Fu, M., Donham, J., Bhagat, N., Mittal, S. and Ryaboy, D., 2014. Apache Storm. Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data,.
[103]. Valentin, G., Zuliani, M., Zilio, D., Lohman, G. and Skelley, A., n.d. DB2 advisor: an optimizer smart enough to recommend its own indexes. Proceedings of 16th International Conference on Data Engineering (Cat. No.00CB37073),.
[104]. Vavilapalli, V., Murthy, A., Douglas, C., Agarwal, S., Konar, M., Evans, R., Graves, T., Lowe, J., Shah, H., Seth, S., Saha, B., Curino, C., O'Malley, O., Radia, S., Reed, B. and Baldeschwieler, E., 2013. Apache Hadoop YARN: Yet another resource negotiator. Proceedings of the 4th annual Symposium on Cloud Computing,.
[105]. Vohra, D., 2016. Apache Sqoop. Practical Hadoop Ecosystem, pp.261-286.
[106]. VOLUME-8 ISSUE-10, AUGUST 2019, REGULAR ISSUE, 2019. Database Migration using Data Synchronization and Transactional Replication. 8(10), pp.2730-2734.
[107]. W, K., 2010. SQL vs. NoSQL.
[108]. Wei, Q., Veeravalli, B., Gong, B., Zeng, L. and Feng, D., 2010. CDRM: A Cost-Effective Dynamic Replication Management Scheme for Cloud Storage Cluster. 2010 IEEE International Conference on Cluster Computing,.

[109]. Wei, Z., Dejun, J., Pierre, G., Chi, C. and van Steen, M., 2008. Service-oriented data denormalization for scalable web applications. Proceeding of the 17th international conference on World Wide Web - WWW '08,.
[110]. Xia, Q., Liang, W. and Xu, Z., 2014. Data Locality-Aware Query Evaluation for Big Data Analytics in Distributed Clouds. 2014 Second International Conference on Advanced Cloud and Big Data,.
[111]. Xiong, M., Ramamritham, K., Haritsa, J. and Stankovic, J., n.d. MIRROR: a state-conscious concurrency control protocol for replicated real-time databases. Proceedings of International Workshop on Advance Issues of E-Commerce and Web-Based Information Systems. (Cat. No.PR00334),.
[112]. Xplenty, 2017. The SQL vs NoSQL Difference: MySQL vs MongoDB.
[113]. Yang, C., Fu, C. and Hsu, C., 2009. File replication, maintenance, and consistency management services in data grids. The Journal of Supercomputing, 53(3), pp.411-439.
[114]. Yang, S., 2012. Move into the Cloud, shall we?. Library Hi Tech News, 29(1), pp.4-7.
[115]. Yang, W., Liu, X., Zhang, L. and Yang, L., 2013. Big Data Real-Time Processing Based on Storm. 2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications,.
[116]. Zaharia, M., Das, T., Li, H., Shenker, S. and Stoica, I., 2020. Discretized streams: An efficient and fault-tolerant model for stream processing on large clusters.
[117]. Zaharia, M., Xin, R., Wendell, P., Das, T., Armbrust, M., Dave, A., Meng, X., Rosen, J., Venkataraman, S., Franklin, M., Ghodsi, A., Gonzalez, J., Shenker, S. and Stoica, I., 2016. Apache Spark. Communications of the ACM, 59(11), pp.56-65.
[118]. Zaharia, M., Xin, R., Wendell, P., Das, T., Armbrust, M., Dave, A., Meng, X., Rosen, J., Venkataraman, S., Franklin, M., Ghodsi, A., Gonzalez, J., Shenker, S. and Stoica, I., 2016. Apache Spark : A unified engine for big data processing. Communications of the ACM, 59(11), pp.56-65.
[119]. Zahid, A., Masood, R. and Shibli, M., 2014. Security of sharded NoSQL databases: A comparative analysis. 2014 Conference on Information Assurance and Cyber Security (CIACS),.
[120]. Zeng, X., Garg, S., Barika, M., Zomaya, A., Wang, L., Villari, M., Chen, D. and Ranjan, R., 2020. SLA Management for Big Data Analytical Applications in Clouds. ACM Computing Surveys, 53(3), pp.1-40.
[121]. Zhang, L., Han, T. and Ansari, N., 2016. Revenue Driven Virtual Machine Management in Green Datacenter Networks Towards Big Data. 2016 IEEE Global Communications Conference (GLOBECOM),.
[122]. Zhao, G., Huang, W., Liang, S. and Tang, Y., 2013. Modeling MongoDB with Relational Model. 2013 Fourth International Conference on Emerging Intelligent Data and Web Technologies,.
[123]. Zhao, Y., Calheiros, R., Bailey, J. and Sinnott, R., 2016. SLA-based profit optimization for resource management of big data analytics-as-a-service platforms in cloud computing environments. 2016 IEEE International Conference on Big Data (Big Data),.
[124]. Zhu, C., Zhou, H., Leung, V., Wang, K., Zhang, Y. and Yang, L., 2017. Toward Big Data in Green City. IEEE Communications Magazine, 55(11), pp.14-18.