T4 Replikácia a sharding (partitioning)

a) Uloženie dát

Denník – kontrolné body a pohľady

b) Replikácia

Sada replík – prim., sekund. replika a rozhodca

c) Sharding

Shard kľúč a typy rozdelenia dát

MongoDB používa replikáciu a sharding na poskytovanie vysoko dostupného systému duplikáciou a distribúciou dát.

Architektúra MongoDB

- **mongod** - ako systém na pozadí spracováva všetky **požiadavky** na dáta, riadi dátový formát a vykonáva operácie manažovania
- **mongos** - je proces, ktorý určuje, kde požadované dáta sú umiestnené v klastri so **shardingom** a smeruje otázky na správny server (rúter).
- **mongo** – zabezpečuje interaktívnu komunikáciu s komponentami MongoDB.

Treba si uvedomiť, že v MongoDB zápis dát môže byť fyzicky realizovaný až niekoľko sekúnd po insert príkaze.

a) Uloženie dát

Denník – kontrolné body a pohľady

Denník je sekvenčný binárny **log** súbor **transakcií**, ktorý slúži na **obnovenie** databázy do platného stavu v prípade náhleho zlyhania alebo vypnutia servera. Údaje sú najprv zapisané do denníka a až potom do dátových súborov.

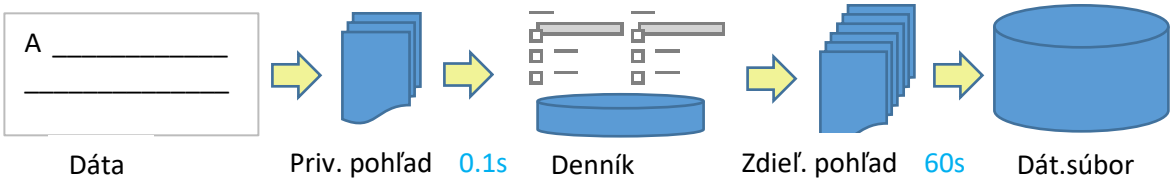
MongoDB používa **kontrolné body** (checkpoints) na poskytnutie konzistentného zobrazenia údajov z disku a zotavenie sa od posledného kontrolného bodu. MongoDB nastavuje pre údaje kontrolné body na WiredTiger v intervale **60 sekúnd** (pozri nižšie) alebo keď už bolo zapísaných 2 GB dát do denníka.

WiredTiger je od verzie 3.2 predvoleným úložným motorom (storage engine). WiredTiger automaticky odstráni staré denníkové súbory, aby udržiaval iba súbory potrebné na obnovenie od posledného kontrolného bodu.

Pri práci s denníkom MongoDB používa dva interné pohľady údajov

- **privátny pohľad** (menší), ktorý sa používa na zápis do **denníkov** a
- **zdieľaný pohľad** (väčší), ktorý sa používa na zápis do **dátových súborov**

a uskutočnia sa 4 až 5 zápisov.



MongoDB **častejšie** píše do denníka ako do dátových súborov:

1. MongoDB operácie zápisu realizuje najprv do **privátneho** pohľadu.
2. MongoDB potom prepíše po skupinách zmeny v privátnom pohľade do súborov **denníka** na disku v adresári denníka približne každých **0.1** sekúnd. Zápisy do denníka sú atomické, zabezpečujúce konzistentnosť denníkových súborov na disku.
3. Po ukončení zapísu do denníka MongoDB aplikuje zmeny z denníka na **zdieľaný** pohľad.
4. Nakoniec MongoDB aplikuje zmeny v zdieľanom pohľade na **dátové súbory** v predvolených intervaloch **60** sekúnd.
5. Pre replikáciu po realizácii databázových operácií na **primárnom MongoDB** sa zaznamenávajú operácie do primárneho **oplogu**.

Pozreme sa teraz najprv na **kopírovanie** (replikácia) a potom na **rozdelenie** (sharding) dát.

b) Replikácia

Replikácia zabezpečuje nadbytočnosť, redundanciu dát ich kopírovaním na viac **uzlov**. Cieľom replikácie duplikovaním dát nie je iba ich ochrana proti nepriaznivým udalostiam, ale aj zvýšenie efektívnosti pre veľký počet **čítaní** údajov.

Množinu uzlov nazývame **klaster**, ak všetky uzly majú rovnaký hardvér, ináč hovoríme o **gride**.

Sada replík – prim., sekund. replika a rozhodca

MongoDB pôvodne podporoval dva typy replikácie:

- tradičnú replikáciu pomocou master/slave (jeden master a veľa slaves) - od verzie 3.2 už nie
- **sadu replík** (replica set)
    - primárna/master replikácia (člen, uzol, server)
    - sekundárna/slave replikácia
    - rozhodca

denník transakcií (log) vs **denník** prevádzkovania (oplog)

Sada replík je **mongod klaster** s maximálne 50 **členmi**, z ktorých iba 7 má hlasovacie právo. V sade je jeden primárny/master a niekoľko sekundárnych/slave replikácií. Ak master **vypadne**, jeden zo sekundárnych uzlov sa automaticky povýši na veliteľa hlasovaním členov v priebehu voľby. **Master** uzol udržuje ohraničený súbor operácií (**oplog,** prevádzkový denník), v ktorom sú uložené zápisy do databázy usporiadané podľa času. **Slaves** replikujú dáta pomocou tohto súboru.

Rozhodca nemá kópiu údajov a nemôže sa stať primárnym. Môže iba hlasovať vo voľbách za primárneho člena.

Poznamenáme, že sekundárni členovia skopírujú oplog primárneho člena a vykonajú všetky operácie v **asynchrónnom** režime. V oploge sú udržiavané iba tie operácie, ktoré zmenia údaje. Pridaním každej novej operácie do oplogu sa staršie vymažú.

Aj keď v MongoDB **čítanie** môže byť presmerované na sekundárnych členov, **zápisy** sú vždy smerované na **primárny** uzol. Vďaka tomu nemôže dojsť k tomu, aby sa dva uzly súčasne snažili aktualizovať rovnaký súbor dát. Dáta uložené na primárnom uzle sú vždy konzistentné – v celku sa hovorí iba o **prípadnej konzistencii**.

Garantovanie konzistentnosti dát na sekundárnych uzloch je možné pomocou špeciálneho parametra zápisu (write concern), ktorý zabezpečí, že sa zápis systémom označuje za **úspešný** iba vtedy, ak sú úspešne zapísané, skopírované aj na všetkých členov. Systém informuje aplikáciu o úspešnom zápise dát na primárneho člena alebo aj na všetkých v závislosti od konfigurácie.

Počet hlasovacích členov v sade replík musí byť nepárny 7, aby sa nenastala *remiza* pri výbere primárneho uzla hlasovaním.

Nasledujúci príkaz zabezpečí, aby dáta boli úspešne zapísané aspoň na dvoch členoch.

```
> db.testovacka.insert( {a : 11,  b : ”laska”},  {writeConcern: {w:2} } )
```

Sada replikácií môže mať až 11 stavov: STARTUP, PRIMARY, SECONDARY, RECOVERING, ARBITER, ...

c) Sharding

Ako sme videli, replikácia rozdistribuuje veľký počet čítaní medzi uzlami, aby sa zvýšila účinnosť **čítania**.

MongoDB využíva **pamäť** hlavne na zníženie doby databázových operácií, veď čítanie dát z pamäte je viac tisíc krát rýchlejšie ako čítanie z disku. Pracovné údaje v MongoDB by sa mali pre ideálny prípad zmestiť do pamäte a mali by sa skladať z najčastejšie používaných dát a indexov.

Chyba stránky (page fault) môže nastať, keď údaje, ktoré nie sú ešte v pamäti, MongoDB prístupní, teda je nútený načítať ich z fyzického disku. V tomto prípade buď existuje voľné miesto v pamäti do ktorého systém priamo načíta požadovanú stránku, alebo v absencii voľného miasta sa stránky v pamäti prepíšu na disk aby sa požadovaná stránka mohla byť načítaná do pamäte, čo spomaľuje proces.

Ďalším príkladom nepriaznivého vplyvu na výkon je dopyt, ktorý má preskenovať všetky dokumenty, veľkosť ktorých presahuje pamäť servera, lebo to tiež vedie k premiestňovaniu dokumentov z pamäti na disk.

Škálovateľnosť, rozšíriteľnosť systému je jeho schopnosť **efektívne** pracovať aj pri **rastúcom** množstve práce, dát. MongoDB škálovateľnosť rieši horizontálnym rozšírením dátových sád, tzn. **rozdelením kolekcií** medzi servermi, čomu sa hovorí **sharding** alebo **partitioning**, rozdeľovanie. Servery sa nazývajú **shard**y (črep, úlomok) a každý server je zodpovedný za manipuláciu s vlastnými dátami bez zaťaženia ostatných.

Každý shard je nezávislá **databáza**. Shardy spolu tvoria jednu logickú databázu, v ktorej počet operácií pripadajúcich na jeden shard je redukovaný. Napríklad pri vložení dát do databázy iba tie shardy sú oslovené, na ktoré sa ukladajú dáta. Táto redukcia je tým väčšia, čím je klaster väčší, čo vedie k zvýšeniu priepustnosti a horizontálnej kapacity. Predpokladajme, že máme kolekciu veľkosti 800 GB. Ak počet shardov je 8 resp. 40, jednotlivé shardy sa majú starať približne iba o 100 resp. 20 GB dát.

O nasadení shardingu je možné uvažovať v nasledujúcich prípadoch:

- veľkosť dátovej sady je obrovská a približuje sa ku kapacite jedného systému
- ak pre aktívnu prácu nastavené limity sú už dosiahnuté, škálovateľnosť MongoDB môže pomôcť, veď pamäť je používaná MongoDB pre rýchle načítanie dát
- ak aplikácia vykoná prevažne zápisy, sharding môže byť použitý na rozdelenie dát na viacerých serveroch.

replica	vs	**shard**
(člen, uzol, server, primárny a sekundárny člen, rozhodca)		(komponenta, mongod, sada replík, map manager, konfig. server)

Komponenty klastera so shardingom sú

- shardy
  Shard je komponenta, kde podmnožina dát z kolekcie je uložená a komponenta môže byť buď **mongod**, ako inštancia MongoDB, alebo **sada replík**. Dáta zo všetkých shardov tvoria kompletnú sadu dát. Nie každá kolekcia musí byť rozdelená.
- **mongos** (shard **map** manager) - udržuje informácie umožňujúce aplikácie sa pripojiť k **správnej** databáze na základe hodnoty **shard kľúča** (pozri nižšie).
- konfiguračné servery - konfiguračný server je taký **mongod**, ktorý obsahuje **metadáta** klastra, opisujúce jeho **organizáciu** a stav.

Shard kľúč a typy rozdelenia dát

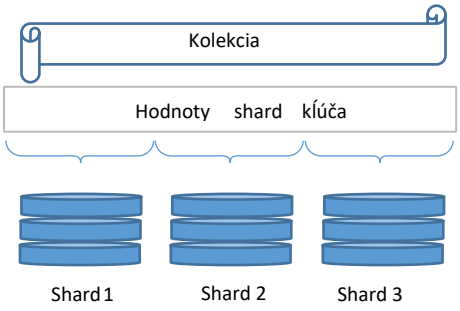
Shard kľúč

Pripomíname, že kľúč zodpovedá atribútu/om a ObjectId zabezpečuje jedinečnosť dokumentov.

V MongoDB údaje kolekcie sú rozdelené pomocou **shard kľúča**, ktorý môže byť ľubovoľný indexovaný jednoduchý alebo kompozitný kľúč kolekcie, definovaný pre každý dokument kolekcie. Shard kľúče sú imutabilné a po inzercii nemôžu byť zmenené.

Rozdelenie dát môže prebiehať buď na základe

- **rozsahu** (range) keď hodnoty shard kľúča sú zatriedené do neprekrývajúcich sa pásiem, rozsahov
- **haš indexu**, keď rozdelenie údajov z kolekcie medzi shardami je viac náhodné vďaka kryptografickému hašu MD5 alebo SHA1-SHA3.



Dáta medzi shardami sa posielajú v **dávkach** (chunk), rozmer ktorých je 64MB.

Sharding vplýva nie len na rýchlosť zápisu, ale aj na rýchlosť čítania. Najrýchlejšie dopyty sú tie, ktoré mongos presmeruje k jedinému shardu pomocou shard kľúča a konfiguračného servera. Ak dopyt neobsahuje shard kľúč, mongos musí preskenovať všetky shardy.

Odporúčania

Replikácia

Replikácie sa používa, ak vysoká dostupnosť je jednou z požiadaviek. Odporúča sa v každom nasadení MongoDB použiť sadu replikácií s najmenej tromi uzlami.

Sharding

Sharding sa používa, ak dáta sa už nezmestia na jeden uzol, v dôsledku čoho sa rovnomerne rozdelujú medzi členmi klastra na báze vhodne zvoleného shard kľúča. Odporúča sa v každom nasadení MongoDB použiť tri konfiguračné servery a začať rozdelenie pred tým, ako dáta dosiahli 256GB.

Príklady

Konfigurácia siete musí umožňovať komunikáciu medzi všetkými členmi klastra. Standardný MongoDB port je 27017.
Príklad 1 – sada replík

- Najprv sa vytvoria vhodné **adresáre** pre budúce dáta a
- po spustení pomenovaných **mongod** inštancií so špeciálnymi parametrami na
    - port
    - replica set
    - oplog
- sa príkazom **mongo** pripojí k jednej inštancii mongod zadaním jej portu
- a **inicializuje** sa replica set.

Príklad 2 – klaster so shardingom

Microsoft Sharding

Shard elasticita je schopnosť vykonať škálovanie v oboch smeroch - horizontálne aj vertikálne.

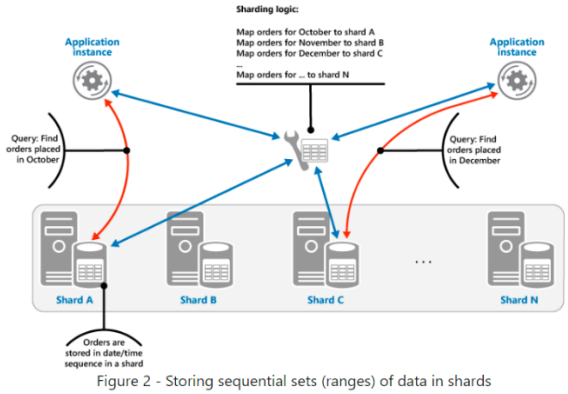


Figure 2 - Storing sequential sets (ranges) of data in shards