UNIVERZITA J. E. PURKYNĚ V ÚSTÍ NAD LABEM
Přírodovědecká fakulta

NOSQL DATABÁZOVÉ SYSTÉMY

Jiří Fišer

Ústí nad Labem 2020

Předmět:	NoSQL databázové systémy
Studijní program:	Aplikovaná informatika
Klíčová slova:	NoSQL databáze, distribuované databázové systémy
Anotace:	Základní principy NoSQL databází (CAP, distribuované systémy), základní druhy typy NoSQL systémů s praktickou prací v REDIS, MongoDB a Neo4j.

Jazyková korektura nebyla provedena, za jazykovou stránku odpovídá autor.

Autor: Jiří Fišer

Obsah

Základní charakteristika NoSQL databází

SQL databáze (relační)

- založené na relačním kalkulu (efektivní INNER JOIN)
- formální schéma tabulek
- důraz na transakční integritu (ACID)
- omezená horizontální škálovatelnost

NoSQL databáze

- nevyužívají relační kalkul (INNER JOIN může být podporován, ale je pomalý)
- schéma je neformální, či parciální
- jen omezená integrita (důsledek CAP, viz dále)
- důraz na horizontální škálovatelnost (replikace, sharding)

ÚKOLY

Jakou roli hraje INNER JOIN v implementaci relačních vztahů? Co je transakční integrita?

Co obecně neplatí o NoSQL databázích

~~No to SQL (Ne jazyku SQL)~~

cílem není nahradit SQL databáze. NoSQL databáze jsou optimalizovány pro jiné typy aplikací (i když v praxi existuje významný překryv)

~~Not only SQL (Něco víc než SQL)~~

I SQL databáze nabízejí přístupy ležící mimo relační jádro SQL (fulltext, ukládání strukturovaných dat, např. JSON)

~~Without SQL~~

Ne všechny databáze nevyužívající relační přístup jsou zahrnovány do pojmu NoSQL (např. objektové či XML)

ÚKOLY

Zjistěte, jaké nerelační datové typy nabízí váš oblíbený databázový systém (MySQL, PostgreSQL, apod.). Diskutujte, proč, nevyhovují relačnímu kalkulu.

Kdy jsou NoSQL databáze vhodné?

- Write once/read many = vložená data nejsou modifikována (resp. modifikována jen zřídka)
- nepředvídatelný nárůst množství zpracovávaných dat (exponenciální růst)
- častá změna schématu

Úkol: Navrhněte scénáře využití databáze, pro něž jsou typické předchozí body.

ŠKÁLOVATELNOST

vertikální škálovatelnost (scaling up)

- výkonnější hardware (rychlejší procesor, více paměti)

  \+ stačí standardní přístupy

  – technologická a finanční omezení

horizontální škálovatelnost (scaling out)

využití distribuovaného systému (clusteru)

\+ možnost využití cloudu

\- vzrůst režie, nespolehlivost sítě

ÚKOLY

Jak vertikální škálovatelnost ovlivňuje Moorův zákon? V jakých oblastech platí IT, a kde spíše ne?

CAP TEORÉM

NoSQL databáze nemohou dosáhnout absolutní konzistence (ACID) ve světě distribuovaných systémů.

C = c*onsistency* = každé čtení získá poslední zapsanou hodnotu nebo je signalizována chyba

A = *availability* (dostupnost) = všechny požadavky jsou čtení/zápis jsou vždy obslouženy (s určitou maximální latencí)

P = *partition tolerance* = systém funguje i při rozpadu na více částí (tj. i když jsou ztraceny či zpožděny zprávy mezi uzly)

Důsledek CAP teorému: **V distribuovaném systému lze dosáhnout vždy maximálně dvou vlastností z CAP.**

Klíčový důsledek: Pokud se předpokládá odolnost proti rozdělení (která je v distribuovaném světě nezbytná), pak je nutné si vybrat mezi (dobrou) dostupností a (dostatečnou) konzistencí.

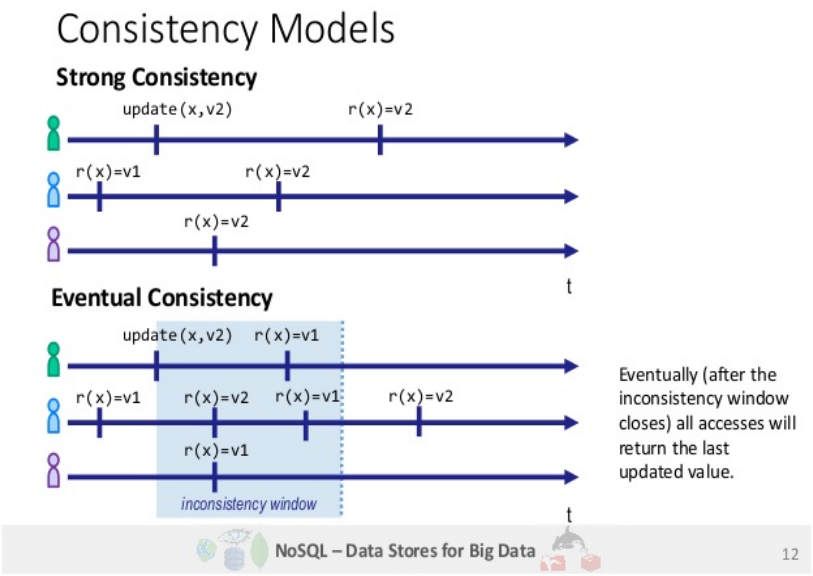
- tj. např. lepší dostupnosti, pokud netrváme na dokonalé konzistenci
- vyšší odolnosti vůči rozpadu, za cenu nižší dostupnosti = vyšší latence (apod.)

ÚKOLY

1. *Platí CAP teorém i v nedistribuovaných systémech?*
2. *Přečtěte si článek o CAP teorému od Erica Brewera na* https://www.infoq.com/articles/cap-twelve-years-later-how-the-rules-have-changed/.

BASE MODEL

- systém je **převážně dostupný** (**B**asically **A**vailable), běžné jsou částečné výpadky (vyšší latence), ale celkový výpadek je nepravděpodobný (výrazně nepravděpodobnější než u systému centralizovaného)
- systém je dynamický a **nedeterministický** (tzv. **S**oft state)
- **občasná (ne)konzistence** (**E**ventual consistency) = systém je občas nekonzistentní tzv. Nekonzistentní okno (resp. obráceně nastávají situace, kdy je konzistentní)



ÚKOLY

Navrhněte scénáře, ve kterých je občasná nekonzistence tolerovatelná a ve kterých nikoliv.

DISTRIBUCE DAT

sharding (rozdělení)
různé části dat (shards = střep) jsou rozmístěny na různé uzly v clusteru => zvyšuje se kapacita a při vhodném rozmístění i dostupnost dat

- rovnoměrné rozdělení dat na uzlech (předpokládáme homogenní cluster)
- minimalizace počtu uzlů, na které musí uživatel přistoupit, aby data získal (jednoduchá distribuční funkce)
- optimalizace fyzického umístění serveru vzhledem k umístění klientů (topologie sítě, geografická lokalizace)

replikace
kopie dat jsou umístěny na některých uzlech clusteru => zvýšení dostupnosti, propustnosti a odolnosti proti rozpadu clusteru

ÚKOLY

1. *Obě techniky (sharding a replikace) jsou tzv. ortogonální. Co to znamená?*
2. *Promyslete vhodné rozdělení případně replikaci dat na případných serverech v budovách univerzitního kampusu.*

Distribuce na nehomogenních uzlech

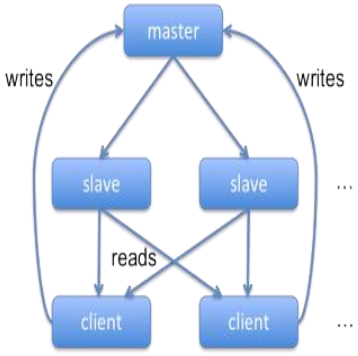
- žhavá (aktuální) data jsou umístěna na rychlých, ale malokapacitních úložištích (nové počítače s SSD disky a rychlým Ethernetem)
- chladná (historická) na pomalejších, ale velkokapacitních úložištích (starší počítače s pevnými disky)

Otázka: Jak tento přístup souvisí s archivací dat a datovou bezpečností?

REPLIKACE

MASTER-SLAVE REPLICATION

- zápis se provádí na jediný uzel (master), který se následně postará o replikaci dat na sekundární (slave) servery.
- čtení zajistí libovolný server (typicky je to slave)



výhody:
dochází jen k write-read konfliktům (občasná a dočasné nekonzistence)
nevýhody:
úzké hrdlo při zápisu
při rozpadu sítě je nutno zvolit nového mastera

PEER-TO-PEER REPLICATION

- všechny uzly jsou si rovny (ale některé jsou si ještě rovnější)

hrozí **write-to-write** kolize
např. souběžný update s dvěma různými novými hodnotami (na dvou uzlech)

efektivní řešení:
využití **kvóra** (je nutný určitý počet uzlů, na nichž se operace provedou, aby byla validní).
Např. je-li replikační faktor N (= počet kopií), pak lze využít těchto minimálních kvór:

W > N/2 (zvítězí maximálně jeden konfliktní zápis)
W+R > N (operace čtení se provede až po operaci zápisu)

Problematika replikace dat v DBMS je složitá problematika. Zajímavý úvod najdete ve článku https://academy.datastax.com/planet-cassandra/data-replication-in-nosql-databases-explained

ÚKOLY

Zjistěte, jaké replikační řešení nabízí vaše oblíbená relační databáze. Diskutujte typ replikace a použitelnost v praxi.

KLASIFIKACE NOSQL DATABÁZÍ

databáze klíč-hodnota: hodnoty odkazované pomocí klíčů (obdoba slovníků)

dokumentové databáze: ukládání strukturovaných dokumentů (typicky [B]JSON)

sloupcové databáze: něco mezi dokumenty a SQL = každý řádek může mít jiné sloupce (které mohou tvořit skupiny)

grafové databáze: relace representované grafy (obtížněji distribuovatelné)

ÚKOLY

Nalezněte příklady databází všech typů a z jejich WWW stránek zjistěte, co nabízejí.

DATABÁZE TYPU KLÍČ-HODNOTA

obdoba slovníků

nejstarší typ NoSQL databází (BerkeleyDB vznikla dříve než SQL)

tento typ databází se může zaměřovat na:

- masivní distribuovatelnost (Riak)
- masivní propustnost (Redis)

navíc je lze dělit podle charakteru odkazovaných hodnot

- BLOB (databáze nevyužívá vnitřní strukturu)
- elementární typy (řetězce, čísla, seznamy, množiny, atd.)

REDIS

Redis is an open source (BSD licensed), in-memory data structure store, used as a database, cache and message broker. It supports data structures such as strings, hashes, lists, sets, sorted sets with range queries, bitmaps, hyperloglogs, geospatial indexes with radius queries and streams.

(citováno z WWW stránek)

in-memory databáze (data jsou uložena v operační paměti), je však možná automatická perzistence (ukládání do trvalého úložiště) včetně podpory hierarchické master-slave replikace.

klíčem může být libovolný bytový řetězec (např. textový řetězec v libovolném ale předem dohodnutém kódování)

hodnotou může být:

1. bytový řetězec (Redis nepodporuje jiné elementární objekty), na některé bytové řetězce však lze aplikovat i aritmetické operace (musí representovat zápis celého čísla)
2. kolekce bytových řetězců (viz charakteristika)

Specifickými rysy REDIS jsou důsledně atomické operace (lze ji tak použít pro komunikaci mezi procesy) a automatická expirace hodnot (využití jako cache).

Prostudujte kvalitní REDIS dokumentaci (https://redis.io/documentation, především https://redis.io/topics/data-types-intro) a zaměřte se na příkazy pro manipulaci s elementárními hodnotami (set, exists, del, inc), seznamy (rpush, lpush, lpop, rpop, lrange, llen) a na automatickou expiraci hodnot (expire, ttl).

ÚKOLY

1. *Jak se liší operace RPOP a BRPOP?*
2. *Nainstalujte si pythonský modul pro REDIS (https://pypi.org/project/redis/)*
3. *Vytvořte program, který rekurzivně prochází stránky staroslověnské wikipedie (https://cu.wikipedia.org) a hledá počet interních odkazů, na jednotlivé stránky a resp. interních odkazů na jednotlivé stránky mířících (je zvolena wikipedie jen s několika málo stovkami stránek). Výsledky interpretujte (znalost azbuky výhodou).*
4. *Použijte REDIS pro representaci synchronizované fronty zpráv mezi dvěma procesy (= dvěma spuštěnými pythonskými skripty, z nichž jeden do fronty zapisuje a druhý čte).*

DOKUMENTOVĚ ORIENTOVANÉ DATABÁZE

MONGO DB (KLASICKÁ DOKUMENTOVĚ ORIENTOVANÁ DATABÁZE)

základní ukládanou entitou je JSON dokument (s rozšířenou syntaxí a sémantikou) podporující:

- číselné a řetězcové atributy
- representace časových okamžiků (obdoba timestamp)
- jedinečné globální identifikátory (obdoba GUID)

ÚKOLY

Pokud neznáte formát JSON prostudujte si ho na https://en.wikipedia.org/wiki/JSON

základní kameny Mongo databáze

- dokument (přibližně odpovídá řádku)
  - klíč (atribut) = položka (identifikovaná jménem)
  - dokument je jednoznačně identifikován povinným klíčem _id
- kolekce (tabulka s dynamickým schématem)
- databáze (seskupení kolekcí ≈ systém práv, umístění)

ÚKOLY

1. *Porovnejte koncept dokumentu s řádkem relační databáze a kolekci s relační tabulkou, Diskutujte především možnost narušení 1.NF (výhody a nevýhody) a neexistenci schématu.*
2. *Prostudujte si MongoDB manuál a to sekce Introduction až MongoDB crud operations.*
3. *Vytvořte s pomocí MongoShell jednoduchou kolekci obsahující tyto údaje o knihách:*
   - *název*
   - *autoři (může jich být více)*
   - *nakladatel (vnořená dokument obsahující jméno nakladatele a jeho původ)*
   - *počet stran*
4. *Vyplňte databázi alespoň deseti položkami a vyzkoušejte dotazy, typu knihy obsahující v názvu, knihy s alespoň dvěma autory, knihy počtem stránek v určitém intervalu.*
5. *Nainstalujte MongoDB modul pro Python.*
6. *Úkol: Vytvořte v Pythonu program, který uloží RSS kanál Idnes.cz (https://servis.idnes.cz/rss.aspx?c=zpravodaj) do MongoDB (s podporou vícenásobného spuštění, kdy jsou vloženy jen nové články). Nad databází proveďte dotazy typu, všechny zahraniční zprávy, všechny zprávy určitého dne, apod.*

GRAFOVÉ DATABÁZE

Grafové databáze se svým principem podstatně liší od SQL a jiných NoSQL databází. Jsou vhodné pro data, která lze representovat jako grafy s různými druhy hran a ohodnocením. Dotazování se převádí na hledání cest v těchto grafech.

NEO4J

Neo4j is a native graph database, built from the ground up to leverage not only data but also data relationships. Neo4j connects data as it's stored, enabling queries never before imagined, at speeds never thought possible. (reklamní popis ze stránek https://neo4j.com/)

ÚKOLY

1. *Nastudujte e-book T*he Definitive Guide to Graph Databases *(zdarma ke stažení na https://neo4j.com/whitepapers/rdbms-developers-graph-databases-ebook, jen 30 stran) a prostudujte a vyzkoušejte příklady v ní uvedení (je nutné si nainstalovat grafické prostředí pro interakci s databází)*
2. *Nainstalujte si Python driver (https://neo4j.com/docs/driver-manual/4.0/)*
3. *Zkuste naplnit Neo4j databázi informacemi z výstupu linuxovského příkazu sudo lshw -json (pokud Linux nemáte, pak použijte live distribuci nebo výstup v podobě JSON souboru*

získejte od šťastlivce, který Linux má). Zaměřte se na atributy id, product a capabilities. Výsledek zobrazte v desktopu Neo4j a zkuste dotazy nad grafy (typu: co je na základní desce, co má capabilitu USB).