UNIVERZITA HRADEC KRÁLOVÉ  
Fakulta informatiky a managementu

*Elektrická energie v Česku v roce 2023*

*Daniel Paleček, 1. ročník*

V Hradci Králové dne 10. 5. 2025

Obsah

[Úvod 3](#_Toc197790849)

[1 Architektura 4](#_Toc197790850)

[1.1 Schéma a popis architektury 4](#_Toc197790851)

[1.2 Specifika konfigurace 4](#_Toc197790852)

[1.2.1 CAP teorém 5](#_Toc197790853)

[1.2.2 Cluster 5](#_Toc197790854)

[1.2.3 Uzly 5](#_Toc197790855)

[1.2.4 Sharding 5](#_Toc197790856)

[1.2.5 Replikace 5](#_Toc197790857)

[1.2.6 Perzistence dat 5](#_Toc197790858)

[1.2.7 Distribuce dat 6](#_Toc197790859)

[1.2.8 Zabezpečení 6](#_Toc197790860)

[2 Funkční řešení 7](#_Toc197790861)

[2.1 Struktura 7](#_Toc197790862)

[2.2 Instalace 7](#_Toc197790863)

[3 Případy užití a případové studie 8](#_Toc197790864)

[4 Výhody a nevýhody 9](#_Toc197790865)

[5 Další specifika 10](#_Toc197790866)

[6 Data 11](#_Toc197790867)

[7 Dotazy 12](#_Toc197790868)

[Závěr 13](#_Toc197790869)

[Zdroje 14](#_Toc197790870)

[Přílohy 15](#_Toc197790871)

Úvod

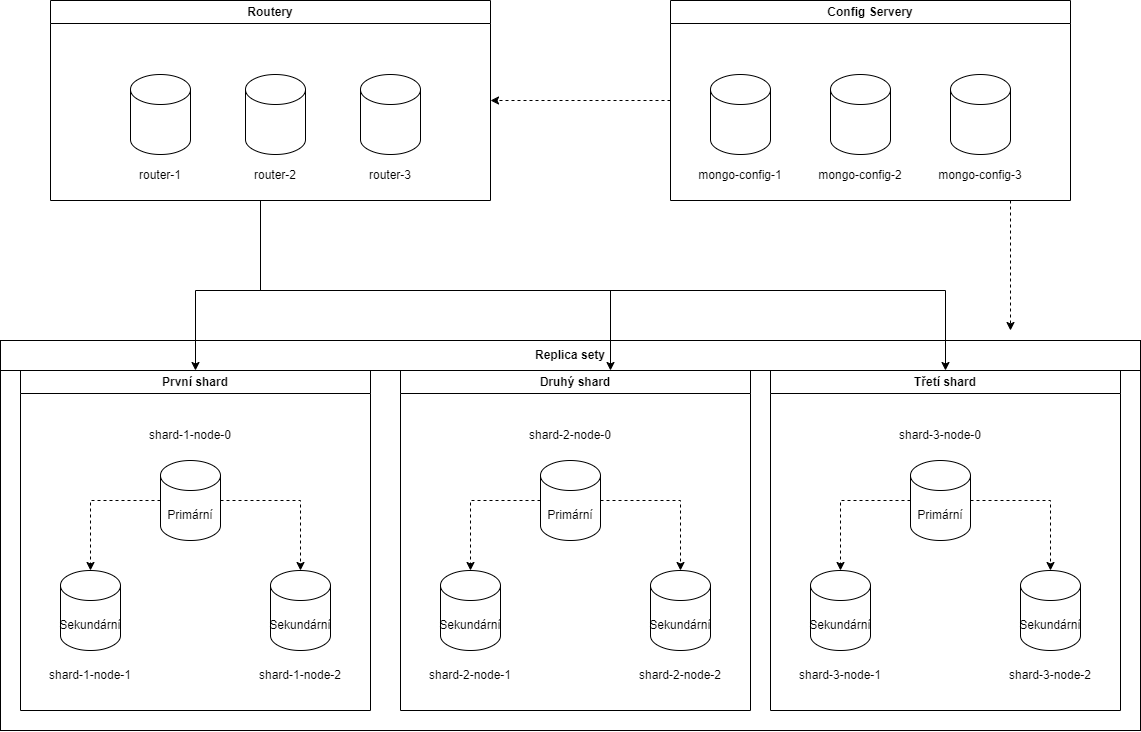
Tato seminární práce se zabývá uložením a zpracováním dat z ENTSO-E Transparency Platform. Data jsou uložená v MongoDB a analýza dat je provedena v pythonu. Zpracovávaná data se týkají produkce, konzumace a ceny elektřiny v české republice v roce 2023. Přesná struktura a provedení projektu a zpracování dat bude popsáno v následujících kapitolách. Semestrální projekt se zabývá pouze daty jedné země a jako možné rozšíření my například mohlo být porovnání mezi zeměmi, které Transparency Platform nabízí.

Co se týče technické stránky projektu, byla použita databáze mongo verze 8.0.9 se třemi routery. Databáze je shardovaná a replikovaná a její spuštění je automatizované viz. read me.

# Architektura

Databáze MongoDB je nasazena v prostředí docker pomocí clusterového řešení. Autentizace mezi jednotlivými uzly clusteru je zajištěna prostřednictvím keyfile.

## Schéma a popis architektury



Architektura vychází z oficiálně doporučeného clusterového nasazení databáze MongoDB, které je navrženo s důrazem na škálovatelnost, konzistentnost a odolnost. Toto řešení umožňuje horizontální škálování při rostoucím objemu dat a zajišťuje redundancy dat prostřednictvím replikačních setů (replica sets). Bezpečnost komunikace mezi jednotlivými uzly clusteru je zajištěna pomocí keyfile. Cluster je nasazený v docker kontejnerech.

MongoDB cluster se skládá ze tří hlavních komponent: konfiguračních serverů, routerů a shardů. Konfigurační servery uchovávají metadata o struktuře clusteru, včetně informací o tom, jak jsou data rozdělena mezi jednotlivé shardy. Routery (mongos) slouží jako vstupní bod pro klientské požadavky a směrují je na příslušné shardy podle rozložení dat. Každý shard je tvořen replikačním setem (replica set) se třemi uzly což zajišťuje konzistenci, replikaci a odolnost vůči výpadkům.

## Specifika konfigurace

Každý komponent systému (konfigurační servery, routery a shardy) byl nasazen jako samostatná docker kontejner, což umožňuje snadné řízení, restartování i škálování jednotlivých částí infrastruktury. Replikační sety shardů byly nastaveny s třemi uzly, kdy jeden je primární a dva sekundární. Konfigurace obsahuje 3 konfigurační servery, 3 routery a 3 shardy.

### CAP teorém

Mongo cluster odpovídá z pohledu CAP teorému typu CP. Upřednostňuje konzistenci a odolnost před dostupností. V případě výpadku části clusteru může být dočasně omezen zápis, aby byla zachována konzistence dat. CP je pro použitá data vhodná garance, protože dostupnost těchto dat není tak důležitá jako jejich konzistence.

Consistency: MongoDB zajišťuje konzistenci tím, že zápisy se provádějí pouze na primárním uzlu. Sekundární uzly přijímají kopie dat a udržují je v konzistentním stavu. To znamená, že v případě, že dojde k přepnutí primárního uzlu, nový primární uzel je zvolen a data jsou stále konzistentní mezi všemi uzly, i když mohou být dočasně nedostupné pro zápisy, dokud není nový primární uzel nastaven.

Partition tolerance: MongoDB je navrženo tak, aby zvládalo síťové rozdělení a pokračovalo v provozu i v případě selhání jednotlivých částí. Dokáže tolerovat síťové výpadky a stále udržovat své operace. Automaticky detekuje rozdělení sítě a používá mechanismus volby nového primárního uzlu, pokud dojde k výpadku aktuálního primárního uzlu. Tento proces se nazývá failover.

### Cluster

Celý cluster je navržen tak, aby umožňoval škálování dle potřeby. Přidáním nových shardů lze rozšiřovat kapacitu clusteru. MongoDB zároveň zajišťuje automatické vyvažování dat mezi shardy (balancing), což minimalizuje riziko nerovnoměrného zatížení jednotlivých uzlů.

### Uzly

Shardy se skládají ze tří uzlů. Data se replikují z primárního uzlu na dva sekundární. V případě výpadku primárního uzlu je zvolen jeden ze sekundárních jako nový primární.

### Sharding

Data jsou rozdělena do tří shardu podle složených klíčů (Area, časový interval). Díky je možné cluster jednoduše horizontálně škálovat. V rámci semestrálního projektu jsou vzhledem k malému množství dat tři shardy více než dostačující.

### Replikace

V semestrální práci je využito tří replikačních sad. Toto řešení splňuje minimální doporučovanou konfiguraci a z povahy tohoto projektu je dostačující. Data se tedy zapisují pouze na primární uzel a na dva sekundárních se pouze udržuje kopie dat z primárního uzlu.

### Perzistence dat

Perzistence je dosaženo tím že, jsou trvale uchovávána na disku tedy v sekundární paměti. Primární tedy operační paměť je v mongu používána hlavně jako cache pro často přistupovaná data. Kvůli rychlosti se mongo snaží držet aktivní data v operační paměti. Pokud je paměť nedostatečná, systém automaticky uvolňuje méně používané části z RAM a znovu je načítá z disku, když jsou potřeba. Vnitřně MongoDB ukládá dokumenty ve formátu BSON.

### Distribuce dat

Data jsou rozdělena na tři shardy, na každém z nich je replikační set se třemi uzli. Zápis do MongoDB je směrován na příslušný shard podle hodnoty shard key. Po zápisu proběhne replikace na sekundární uzly v rámci replikační sady. Při výběru shardu pro zápis MongoDB provádí hashování shard key. Čtení dat může být směrováno jak na primární, tak na sekundární uzly shardu. Tento přístup zvyšuje škálovatelnost čtení, protože aplikace může číst data z více uzlů současně. Tento proces je řízen nastavením readPreference, které určuje, zda se má číst pouze z primárního uzlu, nebo i ze sekundárních.

Nastavení readPreference:

db.getMongo().setReadPref('nearest')

Nastavení shardování:

sh.enableSharding('Transparency');

Nastavení jednoho ze shard keys:

db.adminCommand({ shardCollection: 'Transparency.Load', key: { 'MTU (CET/CEST)': 'hashed' } });

Rozdělení na uzlech:

rs-shard-1 - 67 932

rs-shard-2 - 65 826

rs-shard-3 - 67 712

### Zabezpečení

Zabezpečení databáze je zajištěno vytvořením uživatele s přihlašovacími údaji:

db.createUser({user: "user", pwd: "user", roles:[{role: "root", db: "admin"}]});

Díky tomu je přístup k databázi povolen pouze vytvořeným uživatelům.

K zabezpečení komunikace v rámci clusteru MongoDB používá keyfile. Využití keyfile není nutné, ale je doporučované.

# Funkční řešení

Xxxxxx

## Struktura

.

├── README.md

├── docker-compose.yml

├── import

│   ├── Transparency.Load.json

│   ├── Transparency.Prices.json

│   └── Transparency.Production.json

├── keyfile

├── python

│   ├── analyze.py

│   └── import.py

└── scripts

├── auth.js

├── delete.sh

├── init-configserver.js

├── init-router.js

├── init-shard1.js

├── init-shard2.js

├── init-shard3.js

├── init.sh

└── test-data.js

Projekt se skládá z docker-compose.yml, složky s daty k importu, složky s keyfilem, složky s python scripty k analýze a importu dat a složky se skripty pro instalaci clusteru. Instalace databáze je provedena spuštěním scriptu init.sh. Před spouštěním docker‑compose init script vytvoří keyfile, který se uloží do složky keyfile. Následně init script provolá docker-compose a po dokončení i ostatní příkazy nutné k nastavení databáze.  
Samotná data se nahrávají pomocí python scriptu import.py a jejich analýza je provedena v analyze.py.

### docker-compose.yml

## Router

router1:

image: mongo:8.0.9

container\_name: router-1

command: mongos --port 27017 --configdb rs-config-server/configsvr1:27017,configsvr2:27017,configsvr3:27017 --bind\_ip\_all --keyFile /keyfile/auth/mongodb-keyfile

ports:

- 27117:27017

restart: always

volumes:

- ./scripts:/scripts

- ./data/mongodb\_cluster\_router1\_db:/data/db

- ./keyfile/auth/mongodb-keyfile:/keyfile/auth/mongodb-keyfile:ro

- ./data/mongodb\_cluster\_router1\_config:/data/configdb

router2:

image: mongo:8.0.9

container\_name: router-2

command: mongos --port 27017 --configdb rs-config-server/configsvr1:27017,configsvr2:27017,configsvr3:27017 --bind\_ip\_all --keyFile /keyfile/auth/mongodb-keyfile

volumes:

- ./scripts:/scripts

- ./data/mongodb\_cluster\_router2\_db:/data/db

- ./keyfile/auth/mongodb-keyfile:/keyfile/auth/mongodb-keyfile:ro

- ./data/mongodb\_cluster\_router2\_config:/data/configdb

ports:

- 27118:27017

restart: always

links:

- router1

router3:

image: mongo:8.0.9

container\_name: router-3

command: mongos --port 27017 --configdb rs-config-server/configsvr1:27017,configsvr2:27017,configsvr3:27017 --bind\_ip\_all --keyFile /keyfile/auth/mongodb-keyfile

ports:

- 27131:27017

restart: always

volumes:

- ./scripts:/scripts

- ./data/mongodb\_cluster\_router3\_db:/data/db

- ./keyfile/auth/mongodb-keyfile:/keyfile/auth/mongodb-keyfile:ro

- ./data/mongodb\_cluster\_router3\_config:/data/configdb

links:

- router1

Vytvoření tří routerů s mongos. Mongos je routovací service, který směruje požadavky na shardy, které má k dispozici. Dále se zde nastavuje umístění keyfile, určují se konfigurační servery a mapuje se port 27017 uvnitř kontejneru na port 27117 hostitele. V rámci volumes se kopírují soubory z projektu do kontejneru.

## Config Servers

configsvr1:

image: mongo:8.0.9

container\_name: mongo-config-1

command: mongod --port 27017 --configsvr --replSet rs-config-server --keyFile /keyfile/mongodb-keyfile

volumes:

- ./scripts:/scripts

- ./data/mongodb\_cluster\_configsvr1\_db:/data/db

- ./keyfile/mongodb-keyfile:/keyfile/mongodb-keyfile:ro

- ./data/mongodb\_cluster\_configsvr1\_config:/data/configdb

ports:

- 27119:27017

restart: always

links:

- shard1-0

- shard2-0

- shard3-0

- configsvr2

- configsvr3

configsvr2:

image: mongo:8.0.9

container\_name: mongo-config-2

command: mongod --port 27017 --configsvr --replSet rs-config-server --keyFile /keyfile/mongodb-keyfile

volumes:

- ./scripts:/scripts

- ./data/mongodb\_cluster\_configsvr2\_db:/data/db

- ./keyfile/mongodb-keyfile:/keyfile/mongodb-keyfile:ro

- ./data/mongodb\_cluster\_configsvr2\_config:/data/configdb

ports:

- 27120:27017

restart: always

configsvr3:

image: mongo:8.0.9

container\_name: mongo-config-3

command: mongod --port 27017 --configsvr --replSet rs-config-server --keyFile /keyfile/mongodb-keyfile

volumes:

- ./scripts:/scripts

- ./data/mongodb\_cluster\_configsvr3\_db:/data/db

- ./keyfile/mongodb-keyfile:/keyfile/mongodb-keyfile:ro

- ./data/mongodb\_cluster\_configsvr3\_config:/data/configdb

ports:

- 27121:27017

restart: always

Vytvoření tří konfiguračních serverů.   
**mongod --configsvr:** nastavuje instanci jako konfigurační server

## Shards

shard1-0:

image: mongo:8.0.9

container\_name: shard-1-node-0

command: mongod --port 27017 --shardsvr --replSet rs-shard-1 --keyFile /keyfile/mongodb-keyfile

volumes:

- ./scripts:/scripts

- ./data/mongodb\_cluster\_shard1\_0\_db:/data/db

- ./keyfile/mongodb-keyfile:/keyfile/mongodb-keyfile:ro

- ./data/mongodb\_cluster\_shard1\_0\_config:/data/configdb

ports:

- 27122:27017

restart: always

links:

- shard1-1

- shard1-2

shard1-1:

image: mongo:8.0.9

container\_name: shard-1-node-1

command: mongod --port 27017 --shardsvr --replSet rs-shard-1 --keyFile /keyfile/mongodb-keyfile

volumes:

- ./scripts:/scripts

- ./data/mongodb\_cluster\_shard1\_1\_db:/data/db

- ./keyfile/mongodb-keyfile:/keyfile/mongodb-keyfile:ro

- ./data/mongodb\_cluster\_shard1\_1\_config:/data/configdb

ports:

- 27123:27017

restart: always

shard1-2:

image: mongo:8.0.9

container\_name: shard-1-node-2

command: mongod --port 27017 --shardsvr --replSet rs-shard-1 --keyFile /keyfile/mongodb-keyfile

volumes:

- ./scripts:/scripts

- ./data/mongodb\_cluster\_shard1\_2\_db:/data/db

- ./keyfile/mongodb-keyfile:/keyfile/mongodb-keyfile:ro

- ./data/mongodb\_cluster\_shard1\_2\_config:/data/configdb

ports:

- 27124:27017

restart: always

Ukázka vytvoření shardu (v projektu jsou tři shary). Primární shard (shard1-0) má link na shard1-1 a shard1-2.

**mongod –shardsvr:** nastavuje instanci jako shard

**--replSet rs-shard-1:** přiřazuje shard do replika setu

volumes:

mongodb\_cluster\_router1\_db:

mongodb\_cluster\_router1\_config:

mongodb\_cluster\_router2\_db:

mongodb\_cluster\_router2\_config:

mongodb\_cluster\_router3\_db:

mongodb\_cluster\_router3\_config:

mongodb\_cluster\_configsvr1\_db:

mongodb\_cluster\_configsvr1\_config:

mongodb\_cluster\_configsvr2\_db:

mongodb\_cluster\_configsvr2\_config:

mongodb\_cluster\_configsvr3\_db:

mongodb\_cluster\_configsvr3\_config:

mongodb\_cluster\_shard1\_0\_db:

mongodb\_cluster\_shard1\_0\_config:

mongodb\_cluster\_shard1\_1\_db:

mongodb\_cluster\_shard1\_1\_config:

mongodb\_cluster\_shard1\_2\_db:

mongodb\_cluster\_shard1\_2\_config:

mongodb\_cluster\_shard2\_0\_db:

mongodb\_cluster\_shard2\_0\_config:

mongodb\_cluster\_shard2\_1\_db:

mongodb\_cluster\_shard2\_1\_config:

mongodb\_cluster\_shard2\_2\_db:

mongodb\_cluster\_shard2\_2\_config:

mongodb\_cluster\_shard3\_0\_db:

mongodb\_cluster\_shard3\_0\_config:

mongodb\_cluster\_shard3\_1\_db:

mongodb\_cluster\_shard3\_1\_config:

mongodb\_cluster\_shard3\_2\_db:

mongodb\_cluster\_shard3\_2\_config:

Každý kontejner má vlastní svazek, to slouží k perzistenci dat na hostitelském počítači.

## Instalace

Instalace probíhá automaticky spuštěním init.sh. Script nejdříve vytvoří keyfile a následně spustí docker-compose a cluster nastaví. Na import dat je připraven import.py script.

# Případy užití a případové studie

MongoDB je dokumentová databáze. Díky své flexibilní struktuře bez pevného schématu, podpoře horizontálního škálování a schopnosti pracovat s velkými objemy dat v reálném čase je vhodná pro systémy, kde se očekává rychlý růst nebo vysoká variabilita dat. Typickými případy užití jsou moderní webové a mobilní aplikace, e-commerce platformy, systémy pro sběr a analýzu telemetrických dat, analytické nástroje či personalizační systémy.

Vzhledem k typu dat, kterými se tato semestrální práce zabývá, je MongoDB vhodné, protože se jedná o data bez pevné dat, která definují časové úseky, proto by využití SQL databáze nebylo vhodné. Databáze dovoluje s daty snadno pracovat a propojovat je napříč kolekcemi. Projekt je tedy případem užití MongoDB k ukládání a práci s telemetrickými daty.

Důvodem využití MongoDB místo jiné NoSQL databáze je povaha a struktura využitých dat.

## Případové studie

### Forbes

Forbes je skvělým příkladem toho, jak je MongoDB velmi škálovatelné pro amatérské použití jednotlivými vývojáři až po podniky, které mají více než 100 milionů unikátních uživatelů přistupujících z různých zařízení. MongoDB má horizontální škálovatelnost, která umožňuje rychlé rozšíření bez nutnosti úplného nebo částečného vypnutí serverů. Pro rostoucí podnik je to velmi výhodné a zvyšuje to rychlost vývoje. Jedním z nejdůležitějších pilířů každé webové stránky je rychlost a doba odezvy. MongoDB je v průměru 7krát rychlejší než SQL. To poskytuje uživatelům na webu plynulejší zážitek, protože rychleji reaguje na požadavky na přístup a načítání. Kromě toho má Forbes mnoho složitých dat, která je třeba uložit. Složitá data se týkají rozmanitosti vstupních dat. Integrace těchto dat pomocí tradičních databázových přístupů byla náročným úkolem. MongoDB nabízí jednodušší způsob ukládání nestrukturovaných dat v polo-strukturované podobě. Jedním z nejdůležitějších pilířů vývoje je spolupráce mezi vývojáři. MongoDB podporuje snadné nástroje pro spolupráci a následuje agilní paradigma. Kromě toho je kód MongoDB přátelský k jiným jazykům, což usnadňuje jeho integraci než psaní dotazů v SQL. Navzdory tomu, že CTO Forbesu jasně uvedl, že používají Google Cloud, výhody používání MongoDB stále otevírají dveře pro hybridní cloudový systém, protože je snadno škálovatelný. Navíc pokud má Forbes nějaká citlivá data, která je třeba uchovávat on-premises, bude je možné snadno komunikovat s Google Cloud, protože hybridní přístup není v rozporu s vývojem MongoDB.

Zdroj:[1]

### Square Enix

Společnost Square Enix, známá po celém světě svými herními tituly, čelila problémům se škálovatelností a spolehlivostí své infrastruktury, která byla založena na tradičních relačních databázích. Tyto systémy nedokázaly dostatečně pružně reagovat na rostoucí nároky online herních služeb, které vyžadují nepřetržitou dostupnost, rychlou odezvu a schopnost pracovat s různorodými daty v reálném čase. Aby tyto výzvy překonali, rozhodli se inženýři Square Enix přejít na databázovou platformu MongoDB Atlas. Tato moderní dokumentová databáze jim umožnila konsolidovat různé databázové instance do jednoho efektivního systému, který je snadno škálovatelný a lépe odpovídá dynamickým požadavkům herního prostředí. Díky flexibilnímu datovému modelu MongoDB mohli vývojáři rychleji iterovat a přizpůsobovat aplikace měnícím se požadavkům hráčů. Nasazením MongoDB se Square Enix podařilo zvýšit výkon svých backendových služeb, zlepšit dostupnost po celém světě a současně zefektivnit správu dat. Výsledkem bylo spolehlivé řešení, které podporuje miliony hráčů v reálném čase a umožňuje firmě pružně inovovat bez obav z technologických omezení.

Zdroj:[2]

### Citizens Bank

Citizens Bank, jedna z největších a nejdéle fungujících bank ve Spojených státech, se v posledních letech potýkala s rostoucím tlakem na zajištění bezpečnosti svých digitálních služeb. Tradiční systémy pro detekci podvodů, založené převážně na dávkovém zpracování dat, nebyly schopny dostatečně rychle reagovat na nově se objevující hrozby, zejména v oblasti šekových a vkladových transakcí, které po pandemii COVID-19 prudce vzrostly. V roce 2023 se banka rozhodla pro zásadní modernizaci své platformy pro prevenci podvodů. Cílem bylo konsolidovat tři oddělené aplikace (aplikaci pro monitorování transakcí, správu upozornění a řešení reklamací) do jednoho moderního systému, schopného fungovat v reálném čase. Klíčovým rozhodnutím bylo zvolení databázové platformy MongoDB Atlas hostované na Amazon Web Services (AWS), která umožnila nejen vysoký výkon, ale také snadnou škálovatelnost a flexibilitu datového modelu. Nově vytvořená platforma přinesla bance hned několik zásadních výhod. Dokáže provádět hodnocení rizika transakcí v reálném čase, což výrazně zrychlilo reakční dobu na podezřelé aktivity. MongoDB Atlas zajistil 99,99% dostupnost a průměrnou dobu odezvy pod 30 milisekund u většiny požadavků. Díky automatizaci a lepší správě zdrojů se bance zároveň podařilo snížit provozní náklady o přibližně 1,5 milionu dolarů ročně. Tato technologická transformace nejen zefektivnila boj proti podvodům, ale také posílila důvěru klientů v digitální služby Citizens Bank. Případ Citizens Bank je ukázkou toho, jak mohou tradiční finanční instituce s pomocí moderních cloudových databází jako MongoDB výrazně zlepšit svou výkonnost, bezpečnost a zákaznický servis.

Zdroj:[3]

# Výhody a nevýhody

## Výhody

1. Flexibilní schéma (schema-less model)

MongoDB ukládá data ve formátu BSON (binární JSON), což umožňuje dynamické a nestandardizované dokumenty. Uživatel tak není vázán pevným schématem jako u relačních databází. Tedy každý dokument může mít jinou strukturu podle potřeby. To se hodí zejména při vývoji aplikací, kde se datový model často mění.

2. Horizontální škálování (sharding)

Na rozdíl od mnoha tradičních databází MongoDB podporuje nativní horizontální škálování. Pomocí shardingu lze rozdělit databázi na více shardů, čímž se zvyšuje výkon a umožňuje zpracování obrovského množství dat.

3. Vestavěná podpora replikace

MongoDB umožňuje snadno konfigurovat replikační sady, které zajišťují vysokou dostupnost a odolnost vůči výpadkům. V případě selhání primárního uzlu je automaticky zvolen nový primární uzel.

4. Agregační framework

MongoDB nabízí pokročilý agregační framework, který umožňuje efektivně provádět analýzu dat (např. filtrování, seskupování, počítání, výpočty apod.) přímo v databázi, bez nutnosti dalšího zpracování na straně aplikace.

5. Výkon a rychlost

Díky internímu indexování, podpoře in-memory operací a optimalizovanému zpracování zápisů a čtení dosahuje MongoDB velmi dobrého výkonu i při vysokém zatížení.

## Nevýhody

1. Neefektivní u složitých relačních struktur

MongoDB není ideální pro aplikace, kde je třeba udržovat silně propojená data s mnoha vztahy. Chybí podpora pro klasické relace a složité JOIN operace známé z relačních databází. Nahrazování těchto vztahů často vede k redundanci nebo složitému agregování.

2. Vyšší nároky na paměť a úložiště

Díky absenci normalizace a uložení kompletních dokumentů (včetně opakujících se struktur) může MongoDB spotřebovávat více paměti a diskového prostoru než relační databáze. Dokumenty jsou často větší kvůli duplicitním údajům.

3. Komplexnější agregace může být výkonnostně náročná

Ačkoliv má MongoDB pokročilý agregační framework, některé složitější dotazy (např. vícestupňové výpočty nebo transformace nad velkými daty) mohou vést k výraznému zpomalení, pokud nejsou vhodně optimalizovány nebo indexovány.

4. Transakce jsou omezené a novější

Plná podpora ACID transakcí přišla až ve verzi 4.0. Přestože jsou dnes dostupné, jejich použití napříč více kolekcemi nebo shardovanými databázemi je složitější a méně výkonné než u klasických SQL databází.

## Řešení v této práci

Z hlediska řešení popisovaného v této semestrální práci, bylo využito následujících výhod MongoDB:   
Flexibilní schéma – některá pole v datasetech nemají daný datový typ

Vestavěná podpora replikace – replikace na 3 shardech

Agregační framework – k vytváření dotazů nad daty

Výkon a rychlost – pro zpracování velkých kolekcí

Řešení ale i komplikovaly nevýhody mongoDB jako:

Neefektivní u složitých relačních struktur – propojování napříč datasety je složitější a výpočetně náročnější

Vyšší nároky na paměť a úložiště – řešení není použitelné na slabých zařízeních

Komplexnější agregace může být výkonnostně náročná – některé agregace trvají delší dobu, než vrátí výsledek

# Další specifika

Toto řešení splňuje daná specifika MongoDB clusteru. Splňuje minimální počadavky na replikaci i shading. A k zabezpečení používá keyfile generovaný pomocí openssl.

# Data

Xxxxxx

# Dotazy

Xxxxxx

Závěr

Xxxxxx

Zdroje

[1] (PDF) MongoDB Case Study on Forbes. *ResearchGate* [online]. [vid. 2025-05-10]. Dostupné z: doi:10.13140/RG.2.2.32766.46408

[2] LYNN, Michael. *Scaling the Gaming Industry with Gaspard Petit of Square Enix | MongoDB* [online]. [vid. 2025-05-10]. Dostupné z: https://www.mongodb.com/developer/products/mongodb/scaling-gaming-mongodb-square-enix-gaspard-petit/

[3] Citizens Bank Partners with MongoDB and AWS to Reduce Costs by $1.5M | Citizens Bank & MongoDB Case Study | AWS. *Amazon Web Services, Inc.* [online]. [vid. 2025-05-10]. Dostupné z: https://aws.amazon.com/partners/success/citizens-bank-mongodb/

Přílohy

Xxxxxx