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

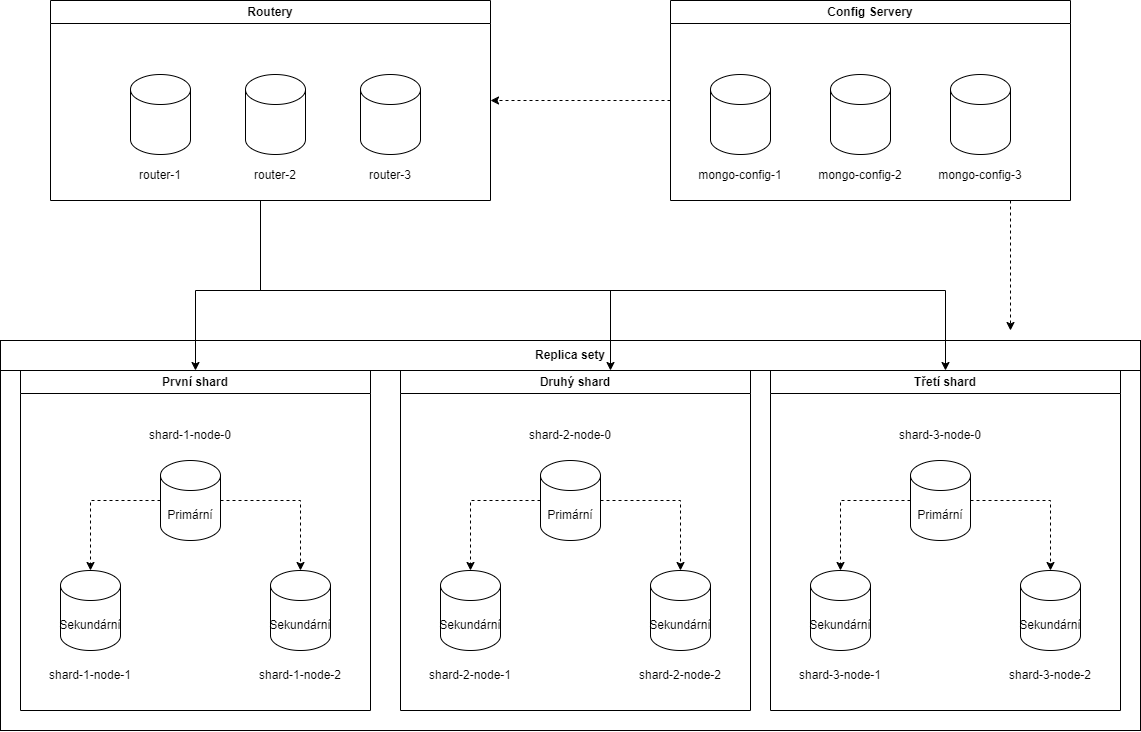
Tato seminární práce se zabývá uložením a zpracováním dat z ENTSO-E Transparency Platform. Data jsou uložená v MongoDB a analýza dat je provedena v pythonu. Zpracovávaná data se týkají produkce, konzumace a ceny elektřiny v české republice v roce 2023. Přesná struktura a provedení projektu a zpracování 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: { 'Area': 1, 'MTU (CET/CEST)': 1 } });

### 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

│   ├── Dockerfile

│   └── auth

├── 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 analíze a importu dat a složky se skripty pro instalaci clusteru. Instalace databáze je provedena spuštěním scriptu init.sh. Před spouštěním docker‑compose se initscript vytvoří keyfile který se uloží do auth

### 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

## Instalace

Xxxxxx

# Případy užití a případové studie

Xxxxxx

# Výhody a nevýhody

Xxxxxx

# Další specifika

Xxxxxx

# Data

Xxxxxx

# Dotazy

Xxxxxx

Závěr

Xxxxxx

Zdroje

Xxxxxx

Přílohy

Xxxxxx