UNIVERZITA PARDUBICE   
Fakulta elektrotechniky a informatiky

*MongoDB Cluster*

*Andrii Streblychenko 3 ročník*

V …..dne….

Obsah

[Úvod 3](#_Toc194133448)

[1 Architektura 4](#_Toc194133449)

[1.1 Schéma a popis architektury 4](#_Toc194133450)

[1.2 Specifika konfigurace 5](#_Toc194133451)

[1.2.1 CAP teorém 5](#_Toc194133452)

[1.2.2 Cluster 5](#_Toc194133453)

[1.2.3 Uzly 5](#_Toc194133454)

[1.2.4 Sharding 5](#_Toc194133455)

[1.2.5 Replikace 6](#_Toc194133456)

[1.2.6 Perzistence dat 6](#_Toc194133457)

[1.2.7 Distribuce dat 6](#_Toc194133458)

[1.2.8 Zabezpečení 6](#_Toc194133459)

[2 Funkční řešení 7](#_Toc194133460)

[2.1 Struktura 7](#_Toc194133461)

[2.1.1 docker-compose.yml 8](#_Toc194133462)

[2.2 Instalace 9](#_Toc194133463)

[3 Případy užití a případové studie 10](#_Toc194133464)

[4 Výhody a nevýhody 11](#_Toc194133465)

[5 Další specifika 12](#_Toc194133466)

[6 Data 13](#_Toc194133467)

[7 Dotazy 14](#_Toc194133468)

[Závěr 15](#_Toc194133469)

[Zdroje 16](#_Toc194133470)

[Přílohy 17](#_Toc194133471)

Úvod

xxxxxxxxx

# Architektura

Cílem této kapitoly je detailně popsat architekturu navrženého řešení pomocí dokumentové NoSQL databáze **MongoDB** ve formě plně shardovaného a replikovaného clusteru. Tento systém je nasazen pomocí **Docker Compose** a využívá více kontejnerů pro různé role v clusteru, jako jsou **config servery, datové shardy, mongos routovací uzly** a **NGINX load balancer**. Řešení je navrženo s důrazem na škálovatelnost, vysokou dostupnost a bezpečnost komunikace mezi jednotlivými komponentami. Architektura reflektuje doporučené praktiky MongoDB

## Schéma a popis architektury

Изображение выглядит как текст, снимок экрана, дисплей, диаграмма

Контент, сгенерированный ИИ, может содержать ошибки.

Celý systém je navržen jako plně škálovatelný MongoDB sharded cluster, nasazený prostřednictvím docker-compose. Architektura zahrnuje tyto komponenty:

NGINX Load Balancer (27080). NGINX slouží jako vstupní bod zvenku. Rozkládá zátěž mezi dvě instance mongos routeru, čímž zajišťuje vysokou dostupnost a vyrovnávání zátěže.

Routers – mongos1, mongos2. Mongos je komponenta MongoDB, která směruje dotazy na správné shardy podle metadata uložených v config servers. Dvě instance mongos zajišťují redundanci a lepší výkon.

Config Servers – configsvr1, configsvr2, configsvr3. Tyto tři uzly tvoří replikový set configReplSet, který uchovává metadata o rozmístění dat mezi shardami. Tato komponenta je nezbytná pro správnou funkčnost sharded clusteru.

Shards – shard1, shard2, shard3. Každý shard je samostatný replikový set o třech uzlech – jeden hlavní (PRIMARY) a dva sekundární (SECONDARY). Tato architektura zajišťuje vysokou dostupnost a replikaci dat pro každý shard.

Docker síť. Všechny komponenty běží v izolované Docker síti, což umožňuje vzájemnou komunikaci, ale omezuje přístup zvenku pouze na port 27080 skrze NGINX.

Odlišnosti od doporučené architektury MongoDB:

Byla přidána NGINX vrstva jako externí přístupový bod s vyrovnáváním zátěže.

## Specifika konfigurace

Architektura tohoto MongoDB řešení je navržena jako plně shardovaný a replikovaný cluster s důrazem na škálovatelnost, vysokou dostupnost, bezpečnost a rozšiřitelnost. Všechny komponenty jsou nasazeny prostřednictvím Docker Compose a komunikují spolu ve vnitřní síti Dockeru.

### CAP teorém

MongoDB ve sharded architektuře se přiklání ke kombinaci CP (Consistency + Partition Tolerance)

Consistency: MongoDB replikové sety zaručují, že zápisy jsou konzistentní a dostupné na PRIMARY uzlu.

Partition Tolerance: systém je odolný vůči výpadkům částí sítě nebo uzlů díky shardingu a replikaci.

### Cluster

Používá se jeden cluster MongoDB, postaveny na «*docker-compose*», který obsahuje všechny komponenty (config servery, mongos, shardy, load balancer). Tento přístup odpovídá standardnímu návrhu distribuované MongoDB infrastruktury.

### Uzly

Celkem je nasazeno 14 uzlů MongoDB:

**3 config servery + 2 mongos routery + (3 shardy × 3 repliky) = 14 uzlů**

Každý replikový set má jednoho PRIMARY a dva SECONDARY uzly.

### Sharding

Použity jsou 3 shardy: *shard1, shard2, shard3* Každý shard obsahuje 3 repliky. Výběr byl učiněn kvůli práci s rozsáhlými daty (např. tisíce záznamů o přestupcích a transakcích), kde je důležité rovnoměrné rozložení zátěže a výkon.

### Replikace

Každý shard je implementován jako replikový set se třemi uzly – *PRIMARY* a dvěma *SECONDARY*. Tim se dava ochranu před výpadkem jednoho nebo dvou uzlů, možnost čtení z replik, pokud je potřeba a automatické přepnutí PRIMARY uzlu v případě výpadku.

### Perzistence dat

Data se ukládají do Docker volume (mapované na /data/dbX) na každém uzlu. MongoDB používá WiredTiger storage engine, který zajišťuje zápis na disk při každé operaci (write concern) a journaling pro zotavení při selhání. Data jsou perzistentní i při restartu kontejnerů díky svázaným svazkům (volumes).

### Distribuce dat

MongoDB rozděluje data na základě shard key, které určujeme při vytváření kolekcí. *Config servery* drží metadata o rozmístění chunků (částí kolekcí). *Mongos* router rozhoduje, který shard má být dotazován. Díky replikaci jsou zápisy zajištěny i v případě výpadku jednotlivého uzlu.

### Zabezpečení

Používá se keyfile-based internal authentication – všechny uzly (config servery, mongos, shard) sdílejí jeden *keyfile*. Dále je nastaven uživatel *admin* s heslem přes .*env* soubor, vyžadovaný pro všechny externí přístupy. Všechny porty kromě *27080* (NGINX) jsou uzavřeny pro vnější přístup, což chrání před neautorizovanými přístupy.

# Funkční řešení

Cílem projektu bylo vytvořit plně automatizovaný a škálovatelný MongoDB cluster, který by fungoval „z krabice“ – tedy bez nutnosti manuálního nastavování jednotlivých komponent. Díky tomu celý cluster lze spustit jediným příkazem přes Docker Compose. Toho bylo dosaženo kombinací konfigurací v Docker Compose souboru (docker-compose.yml), Dockerfile-ů a Bash skriptů (entrypoint skripty, inicializační skripty shardů a config serverů, a automatická tvorba validačních schémat kolekcí). V případě potřeby rozšíření clusteru stačí přidat nové uzly do existujících skriptů a Docker konfigurací.

## Struktura

* **mongodb\_cluster**
  + **config-server**
    - Dockerfile – definuje obraz Docker kontejneru config serveru.
    - configsvr.conf – konfigurační soubor MongoDB config serveru.
    - entrypoint.sh – skript, který inicializuje config server.
    - keyfile – sdílený soubor pro interní autentizaci clusteru.
  + **mongos**
    - Dockerfile – definuje obraz Docker kontejneru pro mongos router.
    - mongos.conf – konfigurační soubor pro mongos router.
    - entrypoint.sh – automaticky registruje shardy v clusteru.
    - schema-collections-init.sh – automatická inicializace validačních schémat kolekcí.
    - keyfile – sdílený autentizační soubor.
  + **shard**
    - Dockerfile – definuje obraz Docker kontejneru shardu s replikační sadou.
    - entrypoint.sh – inicializační skript shardu, který startuje všechny instance shardu.
    - init-shard.sh – nastavuje MongoDB replikační sadu pro shard.
    - mongod{1-3}.conf.template – šablony konfigurací pro jednotlivé instance shardu.
    - supervisord.conf.template – konfigurace Supervisora pro správu instancí shardu.
    - keyfile – sdílený autentizační soubor.
  + **nginx**
    - nginx.conf – konfigurace load balanceru, který směruje požadavky na mongos routery.
  + .env – definuje přístupové údaje a společné proměnné prostředí.
  + docker-compose.yml – centrální soubor konfigurující celý cluster MongoDB.

### docker-compose.yml

Soubor docker-compose.yml definuje strukturu a konfiguraci celého clusteru MongoDB v Docker kontejnerech. Cluster je složen z:

**Config servery** (3 uzly: configsvr1, configsvr2, configsvr3)

* Uložení metadat a konfigurace celého sharded clusteru.
* Nastavují se v režimu replikační sady nazvané configReplSet.
* Používají se sdílené proměnné prostředí (.env) a keyfile pro zabezpečenou interní komunikaci.
* Uzly sdílí volume pro perzistenci konfiguračních dat.

**Shardy** (3 replikované shardy: shard1, shard2, shard3)

* Každý shard sestává ze 3 instancí (hlavní uzel + 2 repliky).
* Každý shard má vlastní replikační sadu (např. shard1).
* Každá instance shardu běží pod kontrolou Supervisoru.
* Data jsou persistována přes oddělené Docker volumes.

**Mongos routery** (2 routery: mongos1, mongos2)

* Přijímají a směrují dotazy aplikací na konkrétní shardy.
* První router automaticky registruje shardy při spuštění (REGISTER\_SHARDS=true).
* Komunikují s config servery a shardy přes interní Docker síť.

**NGINX load balancer**

* Veřejně přístupný jediný bod vstupu (port 27080).
* Směruje požadavky na oba mongos routery pro zajištění vysoké dostupnosti a škálovatelnosti.
* Konfigurační soubor NGINX se připojuje přes volume.

Díky této konfiguraci všechny služby běží izolovaně a bezpečně. Jediný veřejně dostupný port je NGINX na localhost:27080.

## Instalace

Instalace řešení byla vytvořena tak, aby probíhala zcela automaticky, s minimální nutností manuálních zásahů. Celý cluster MongoDB lze spustit jediným příkazem z kořenového adresáře projektu:

**docker-compose up --build**

Parametr --build zajistí automatické sestavení všech kontejnerů a spuštění jejich inicializačních skriptů, čímž nastaví replikační sady, autentizaci a registraci shardů v clusteru.

Cluster bude po úspěšném spuštění dostupný přes load balancer na:

**mongodb://clusterAdmin:YourStrongPassword@localhost:27080**

* Uživatel: clusterAdmin
* Heslo: YourStrongPassword
* Připojení: Port 27080, databáze: main

Databáze main se zakládá automaticky při spuštění clusteru, spolu s kolekcemi a jejich validačními schématy.

Testovací data je možné nahrát následně jednoduchým spuštěním připravených Python skriptů, které jsou umístěné v adresáři noSQL/Data/scripts. Tyto skripty automaticky nahrají CSV datasety do odpovídajících MongoDB kolekcí:

**python upload\_netflix\_films.py**

**python upload\_indian\_traffic.py**

**python upload\_mastercard\_stock.py**

# Případy užití a případové studie

Xxxxxx

# Výhody a nevýhody

Xxxxxx

# Další specifika

Xxxxxx

# Data

Xxxxxx

# Dotazy

Xxxxxx

Závěr

Xxxxxx

Zdroje

Xxxxxx

Přílohy

Xxxxxx