Универзитет у Нишу
Електронски факултет

Cluster решење Neo4J базе података

Семинарски рад
Системи за управљање базама података

Предметни наставник:
Александар Станимировић

Студент:
Ивана Алексић, бр. индекса 1590

Садржај

1.Кластеризација

Кластеризација (Clustering) је техника у рачунарству и информационим технологијама која омогућава груписање више рачунарских ресурса (сервери, рачунари или чворови) тако да раде заједно као један систем. Ова техника се користи за побољшање перформанси, скалабилности и толеранције система на грешке.

Основни концепти кластеризације су чворови и клартери. Чвор је појединачна јединица у кластеру, обично сервер или рачунар. Чворови раде заједно како би пружили побољшане перформансе и доступност. Кластер је скуп чворова који раде заједно и међусобно комуницирају како би извршавали задатке. Чворови у кластеру деле оптерећење и подржавају једни друге у случају отказа. Ако један чвор у кластеру откаже, други чворови преузимају његове задатке, осигуравајући да систем остане оперативан. Кластери омогућавају хоризонтално скалирање, што значи да се капацитет система може повећати додавањем више чворова, што омогућава систему да издржи веће оптерећење и више захтева.

Кластеризација се може користити за:

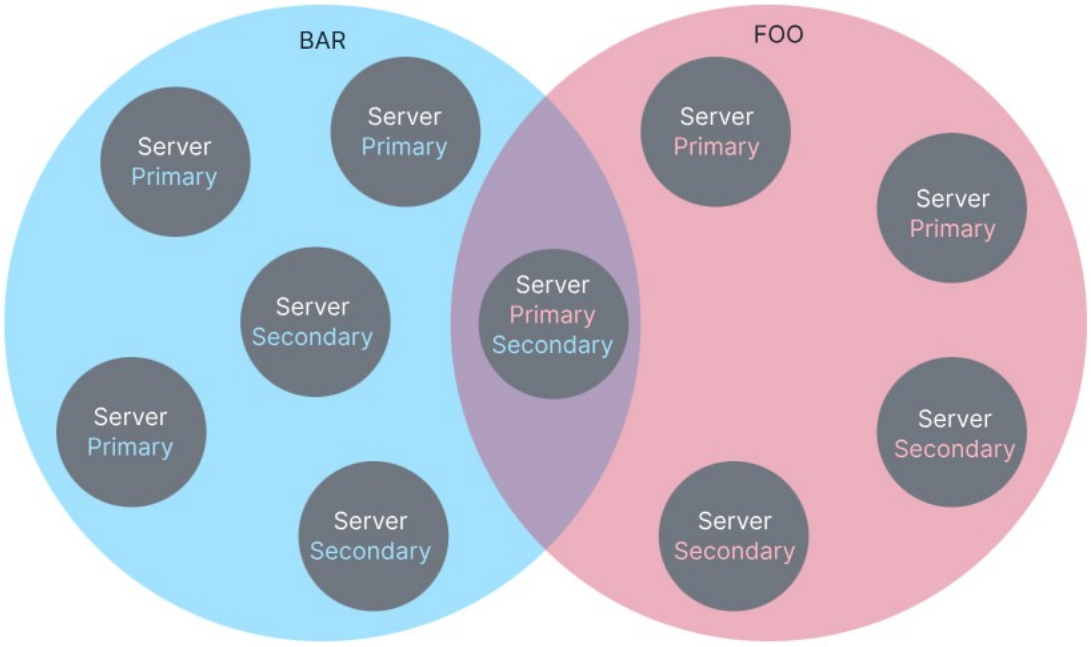
- Балансирање оптерећења на све чворове у кластеру,
- Континуирани рад система тако што нови чворови преузимају задатке отказаних чворова,
- Захтевне рачунарске задатке, где више чворова ради паралелно на истом задатку како би убрзали обраду.
- Обезбеђују дистрибуирано складиштење података, где подаци могу бити реплицирани и доступни са више чворова, чиме се повећава доступност и брзина приступа подацима.

2. Кластеризација и Neo4j

Neo4J кластеринг омогућава да сервери који хостују базе података у примарном режиму обезбеђују платформу отпорну на грешке за обраду трансакција која остаје доступна док већина примарних сервера функционише. Сервери који хостују базе података у секундарном режиму обезбеђују скалабилну платформу за упите ка графу у дистрибуираној топологији. Када се позове, клијентска апликација гарантује да ће прочитати бар своје сопствене записе. Управљање базом података је одвојено од управљања сервером.

Заједно, ово омогућава систему крајњег корисника да буде потпуно функционалан и да чита и уписује у базу података у случају вишеструких кварова на хардверу и мрежи. Поред тога, администрација кластера није компликована, укључујући скалирање величине кластера и дистрибуцију и балансирање доступних ресурса. Са оперативне тачке гледишта, корисно је посматрати кластер као хомогени скуп сервера који покрећу бројне базе података. Сервери имају две различите могућности хостовања базе података, које се називају примарни и секундарни режим. Сервер може истовремено да делује као примарни хост за једну или више база података и као секундарни хост за друге базе података. Слично, могуће је да база података буде смештена на само једном серверу, чак и када је тај сервер део кластера. У таквим случајевима, сервер увек хостује ту базу података у примарном режиму.

Ова два режима су основна у било којој продукцији.



Слика 1: Архитектура кластера

2.1 Примарни режим сервера

Сервер који хостује базу података у примарном режиму омогућава операције читања и писања. База података може бити хостована од стране једног или више примарних хостова. Да би се постигла висока доступност, треба креирати базу података са више примарних хостова. Ако висока доступност није потребна, онда се база података може креирати са једним примарним сервером за минимално кашњење при упису.

Примарне базе података постижу високу доступност реплицирањем свих трансакција користећи *Raft* протокол. *Raft* осигурава да су подаци трајно безбедни тако што чека да већина примарних сервера у бази података (N/2+1) потврди трансакцију, пре него што потврди да је изведена апликацији крајњег корисника. У пракси, само један од више примарних сервера извршава трансакције уписа од клијената. Сервер који уписује се бира аутоматски између примарних сервера базе података и може се променити током времена. Сервер који уписује синхроно реплицира записе осталим примарним серверима. Секундарни сервери базе података асинхроно реплицирају записе са ажурираних чланова кластера.

Синхрона репликација има утицај на кашњење трансакције уписа. Имплицитно, трансакције уписа се потврђују најбржом већином, али како број примарних сервера расте, тако расте и величина већине потребне за потврду уписа.

Толеранција грешке за базу података се израчунава са формулом M = 2Φ + 1, где је M број примарних сервера који су потребни да толеришу Φ грешака. Примери -

- Да би се толерисала два пала примарна сервера, потребна је топологија од пет сервера који хостују базу података у примарном режиму.
- Кластер који је најмање отпоран на грешке, кластер који може толерисати само једну грешку, мора имати три примарне базе података.
- Могуће направити кластер који се састоји од само два примарна сервера. Међутим, тај кластер није отпоран на грешке. Ако један од два сервера поквари, преостали сервер постаје само за читање.
- База података са једним примарним сервером не може толерисати никакве грешке. Због тога је препоручљиво имати три или више примарних сервера за постизање високе доступности.

Ако база података претрпи довољно примарних грешака, више не може да обрађује уписе и постаје само за читање да би се сачувала безбедност.

2.2 Секундарни режим сервера

Секундарни сервери базе података се асинхроно реплицирају са примарних путем слања евиденције трансакција. Они периодично испитују сервер који је виши по хијерархији за нове трансакције које им они шаљу их. Многи секундарни сервери могу добијати податке из релативно малог броја примарних сервера, што омогућава да се велики посао одради из упита.

Базе података обично имају релативно велики број секундарних сервера. Губитак секундарног не утиче на доступност базе података, осим губитка њеног дела пропусности упита графа. То не утиче на толеранцију грешке базе података.

Главна одговорност секундарних база података је да скалирају радна оптерећења читања. Секундарне јединице се понашају као кеш меморије за податке графа и у потпуности су способне да извршавају произвољне упите и процедуре (само за читање).

Због своје асинхроне природе, секундарни сервери можда неће обезбедити све трансакције извршене на примарном серверу(серверима).

2.3 Узрочна конзистентност

Узрочна конзистентност је један од бројних модела доследности који се користе у дистрибуираном рачунарству. Обезбеђује да узрочно повезане операције види свака инстанца у систему истим редоследом. Сходно томе, клијентске апликације гарантовано читају сопствене записе, без обзира на то са којом инстанцом комуницирају. Ово поједностављује интеракцију са великим кластерима, омогућавајући клијентима да их третирају као један (логички) сервер.

Узрочна конзистентност омогућава упис у базе података хостоване на серверима у примарном режиму и читање тих записа из база података хостованих на серверима у секундарном режиму (где су операције графа смањене). На пример, узрочна конзистентност гарантује да је запис који је креирао кориснички налог присутан када тај исти корисник накнадно покуша да се пријави.

Приликом извршавања трансакције, клијент може затражити обележивач који затим представља као параметар наредним трансакцијама. Користећи тај обележивач, кластер може да обезбеди да ће само сервери који су обрадили клијентову обележену трансакцију покренути следећу трансакцију. Ово обезбеђује узрочни ланац који омогућава исправну семантику читања после писања са становишта клијента.

Осим обележивача, свим осталим управља кластер. Управљачки програми базе података раде са менаџером топологије кластера како би одабрали најприкладније сервере за усмеравање упита.

2.4 Рафт алгоритам

Рафт је консензус алгоритам који се користи за управљање реплицираним системима, осигуравајући да сви чворови у кластеру буду усклађени и да коректно примењују промене. Консензус је основни проблем у дистрибуираним системима отпорним на грешке. Консензус подразумева да се више сервера усагласи око вредности.

Сваки сервер има машину стања и лог. Машина стања је компонента чију отпорност на грешке жељени циљ. Клијентима ће се чинити да комуницирају са једном поузданом машином стања, чак и ако мањина сервера у кластеру откаже. Свака машина стања као улаз узима команде из свог лога. Лог би укључивао команде попут "постави x на 3".

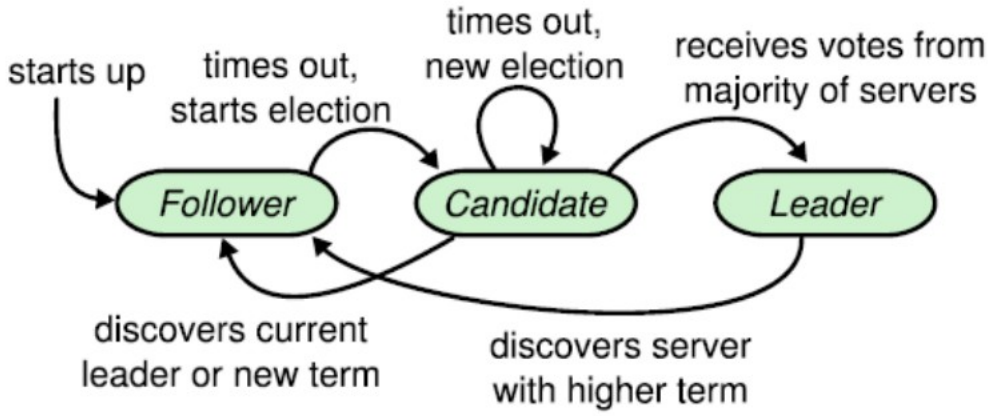
Консензус алгоритам се користи за сагласност око команди у логовима сервера. Консензус алгоритам мора осигурати да ако било која машина стања примени команду "постави x на 3" као n-ту команду, свим машинама стања ће n-та команда бити иста. Као резултат, свака машина стања обрађује исти низ команди и на тај начин производи исти низ резултата и долази до истог низа стања.

Један чвор у кластеру је изабран за лидера. Лидер је одговоран за управљање свим операцијама уписа и координисање репликације података на следбенике, прима уписне захтеве од клијената и додаје их у свој лог, а затим шаље уносе лога следбеницима и чека да већина следбеника потврди пријем уноса пре него што сматра да је унос потврђен. Када је унос потврђен, лидер примењује унос на своје податке и информише клијента да је операција успешна. Сви остали чворови су следбеници, примају команде од лидера и реплицирају податке. Ако следбеник има унос који није у складу са лидерским логом, лидер ће преписати неконзистентне уносе следбеника својим уносима. Лидер осигурава да се уноси примењују у истом редоследу на свим чворовима.

Време је подељено на термине. Сваки термин почиње изборима новог лидера. Термини су нумерисани целим бројевима и користе се за идентификацију текућег стања кластера.

Када следбеник не добије поруку од лидера у одређеном временском периоду, он постаје кандидат и иницира изборе. Кандидат шаље захтеве за гласовима свим осталим чворовима и сваки чвор може гласати за једног кандидата по термину. Кандидат који добије већину гласова постаје лидер за тај термин. Различити чворови имају различите тајмауте за детекцију неактивности лидера како би се смањила вероватноћа покретања избора у исто време. Лидер

периодично шаље "heartbeats" како би одржао своју лидерску позицију и спречио следбенике да постану кандидати.



Слика 2: Рафт алгоритам

3. Постављање кластера

Први корак у подешавању инфраструктуре кластера је конфигурација више сервера који ће формирати кластер на којем се могу хостовати базе података.

Најбитнија подешавања сервера:

- **server.default_advertised_address** - Адреса коју ће друге машине користити за повезивање. У типичном случају, ово треба да буде потпуно квалификовано доменско име или IP адреса овог сервера.

- **server.default_listen_address -** Адреса или мрежни интерфејс који ова машина користи за слушање долазних порука. Подешавање ове вредности на 0.0.0.0 омогућава Neo4j-у да се повеже на све доступне мрежне интерфејсе.
- **dbms.cluster.discovery.endpoints -** Адреса за откривање мреже за све чланове кластера, укључујући и овај сервер. Подешавање мора бити исто на свим члановима кластера. Понашање овог подешавања може се модификовати конфигурацијом поставке **dbms.cluster.discovery.resolver_type**.
- **initial.dbms.default_primaries_count -** Број иницијалних хостовања базе података у примарном режиму. Ако није специфицирано, подразумевано је једно хостовање у примарном режиму.
- **initial.dbms.default_secondaries_count -** Број иницијалних хостовања базе података у секундарном режиму. Ако није специфицирано, подразумевано је нула хостовања у секундарном режиму.

3.1 Креирање Neo4J кластера у Docker-у

Како би се креирао Neo4J кластер, потребно је преузети Neo4J слику, командом:

docker pull neo4j. Први корак је креирање docker мреже која омогућава контејнерима да међусобно комуницирају као и да комуницирају са спољним светом. Приказана наредба креира мрежу под називом ***neo4j-cluster1***

```
D:\SISTEMI3>docker network create neo4j-cluster1
652886e50ad8e44e0664bfa3ba0a79455424e378e0a22fdddabcfe085197cd8f
```

Слика 3: Креирање мреже у

Након креирања мреже, креирају се чворови кластера и сваки чвор се засебно конфигурише. Приказана команда покреће docker контејнер са Neo4j базом података, конфигурисаним за кластеризацију. Параметри напредбе:

- **--name neo4j1** – додељује контејнеру име ***neo4j1***
- **--net=neo4j-cluster1 -** Повезује контејнер са раније креираном docker мрежом ***neo4j-cluster1***
- **-p 7477:7477 -p 7688:7688** – Мапира портове на хосту и портове у контејнеру. Порт **7477** на хосту је маприан на порт **7477** у контејнеру, а порт **7688** на хосту на порт **7688** у контејнеру, што омогућава приступ Neo4J сервису изван контејнера.
- **-e NEO4J_server.cluster.role=CORE -** Поставља улогу чвора на "CORE", што значи да је то главни чвор који може да прима операције

- **-e NEO4J_server.default__advertised__address=neo4j1** – Поставља адресу коју ће овај чвор оглашавати другим чворовима у кластеру
- **-e NEO4J_server.causal__clustering__initial__discovery__members=neo4j1:5000,neo4j2:5000,neo4j3:5000** – Поставља листу иницијалних чланова за откривање у кластеру. Листа садржи имена и портове других чворова у кластеру.
- **-e NEO4J_server.connector.bolt.address=neo4j1:7688** – Поставља адресу за Bolt на **neo4j1:7688**
- **-e NEO4J_server.connector_http_advertised__address=neo4j1:7477** - Поставља адресу за HTTP на **neo4j1:7477**
- **-e NEO4J_server.config.strict_validation.enable=false** – Онемогућава строгу валидацију конфигурације
- **-v neo4j1_data:/data** - Монтира Docker волумен neo4j1_data на директоријум /data унутар контејнера, где се чувају подаци базе.
- **-v neo4j1_logs:/logs** - Монтира Docker волумен neo4j1_logs на директоријум /logs унутар контејнера, где се чувају логови базе.
- **-v neo4j1_import:/var/lib/neo4j/import** - Монтира Docker волумен neo4j1_import на директоријум /var/lib/neo4j/import унутар контејнера, који се користи за увоз података.
- **-v neo4j1_plugins:/plugins** - Монтира Docker волумен neo4j1_plugins на директоријум /plugins унутар контејнера, где се чувају додаци за Neo4j.
- **neo4j:latest** – Користи најновију верзију Neo4j Docker слике

```
D:\SISTEMI3>docker run -d --name neo4j1 --net=neo4j-cluster1 -p 7477:7477 -p 7688:7688 -e NEO4J_server.cluster.role=CORE -e NEO4J_server.default__adve
rtised__address=neo4j1 -e NEO4J_server.causal__clustering__initial__discovery__members=neo4j1:5000,neo4j2:5000,neo4j3:5000 -e NEO4J_server.connector.b
olt.address=neo4j1:7688 -e NEO4J_server.connector_http_advertised__address=neo4j1:7477 -e NEO4J_server.config.strict_validation.enable=false -v neo4j1
_data:/data -v neo4j1_logs:/logs -v neo4j1_import:/var/lib/neo4j/import -v neo4j1_plugins:/plugins neo4j:latest
c720acf5ac5e229fb8101bca3d6a1ae1ebe61aec36ccc20378b9da919d5eef79
```

Слика 4: Покретање и конфигурисање првог чвора

```
D:\SISTEMI3>docker run -d --name neo4j2 --net=neo4j-cluster1 -p 7478:7478 -p 7689:7688 -e NEO4J_server.cluster.role=CORE -e NEO4J_server.default__adve
rtised__address=neo4j2 -e NEO4J_server.causal__clustering__initial__discovery__members=neo4j1:5000,neo4j2:5000,neo4j3:5000 -e NEO4J_server.connector.b
olt.address=neo4j2:7689 -e NEO4J_server.connector_http_advertised__address=neo4j2:7478 -e NEO4J_server.config.strict_validation.enable=false -v neo4j2
_data:/data -v neo4j2_logs:/logs -v neo4j2_import:/var/lib/neo4j/import -v neo4j2_plugins:/plugins neo4j:latest
1a230a4da497b65923a31fdfe91956513028a9a1fbbd67be05b2871c34c2ac41
```

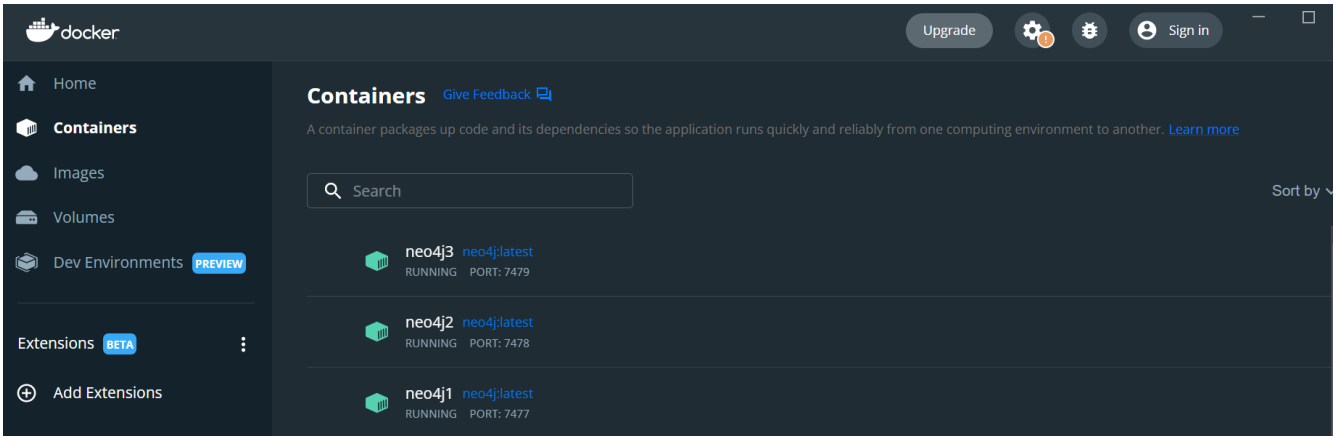
Слика 5: Покретање и конфигурисање другог чвора

```
D:\SISTEMI3>docker run -d --name neo4j3 --net=neo4j-cluster1 -p 7479:7479 -p 7690:7688 -e NEO4J_server.cluster.role=CORE -e NEO4J_server.default__adve
rtised__address=neo4j3 -e NEO4J_server.causal__clustering__initial__discovery__members=neo4j1:5000,neo4j2:5000,neo4j3:5000 -e NEO4J_server.connector.b
olt.address=neo4j3:7690 -e NEO4J_server.connector_http_advertised__address=neo4j3:7479 -e NEO4J_server.config.strict_validation.enable=false -v neo4j3
_data:/data -v neo4j3_logs:/logs -v neo4j3_import:/var/lib/neo4j/import -v neo4j3_plugins:/plugins neo4j:latest
3f367319ff25b3563fa96b132836dff786afa2b2d3907ee076df2534156251a4
```

Слика 6: Покретање и конфигурисање трећег чвора

Ознаке **-e, -v ,-p** из претходних наредби означавају, редом, за постављање променљивих окружења унутар docker контејнера, за монтирање docker волумена у контејнеру, за маприање портова између хост машине и контејнера.

Након покретања чворова, у Docker десктоп апликацији се појављују контејнери:



Слика 7: Контејнери у Docker-у

Провером мреже, може се видети да она саржи три креирана чвора:

```
PS D:\SISTEMI3> docker network inspect neo4j-cluster1
[
    {
        "Name": "neo4j-cluster1",
        "Id": "652886e50ad8e44e0664bfa3ba0a79455424e378e0a22fdddabcfe085197cd8f",
        "Created": "2024-06-25T16:31:40.751079591Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "172.21.0.0/16",
                    "Gateway": "172.21.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "1a230a4da497b65923a31fdfe91956513028a9a1fbbd67be05b2871c34c2ac41": {
                "Name": "neo4j2",
                "EndpointID": "fc30785eb0e79bfe89e112d0cddb5aba87b31afdb731ff1da1cbf361e9bacdb5",
                "MacAddress": "02:42:ac:15:00:03",
                "IPv4Address": "172.21.0.3/16",
                "IPv6Address": ""
            },
            "3f367319ff25b3563fa96b132836dff786afa2b2d3907ee076df2534156251a4": {
                "Name": "neo4j3",
                "EndpointID": "18077967b248ef0c4937f1ec68484add16027bfc18769734876774723c46eeb3",
                "MacAddress": "02:42:ac:15:00:04",
                "IPv4Address": "172.21.0.4/16",
                "IPv6Address": ""
            },
            "c720acf5ac5e229fb8101bca3d6a1ae1ebe61aec36ccc20378b9da919d5eef79": {
                "Name": "neo4j1",
                "EndpointID": "c409c70c83c0f2c2d7636701ba9451a933164dfe06b67749fc7f65de36da5273",
                "MacAddress": "02:42:ac:15:00:02",
                "IPv4Address": "172.21.0.2/16",
                "IPv6Address": ""
            }
        },
        "Options": {},
        "Labels": {}
    }
]
```

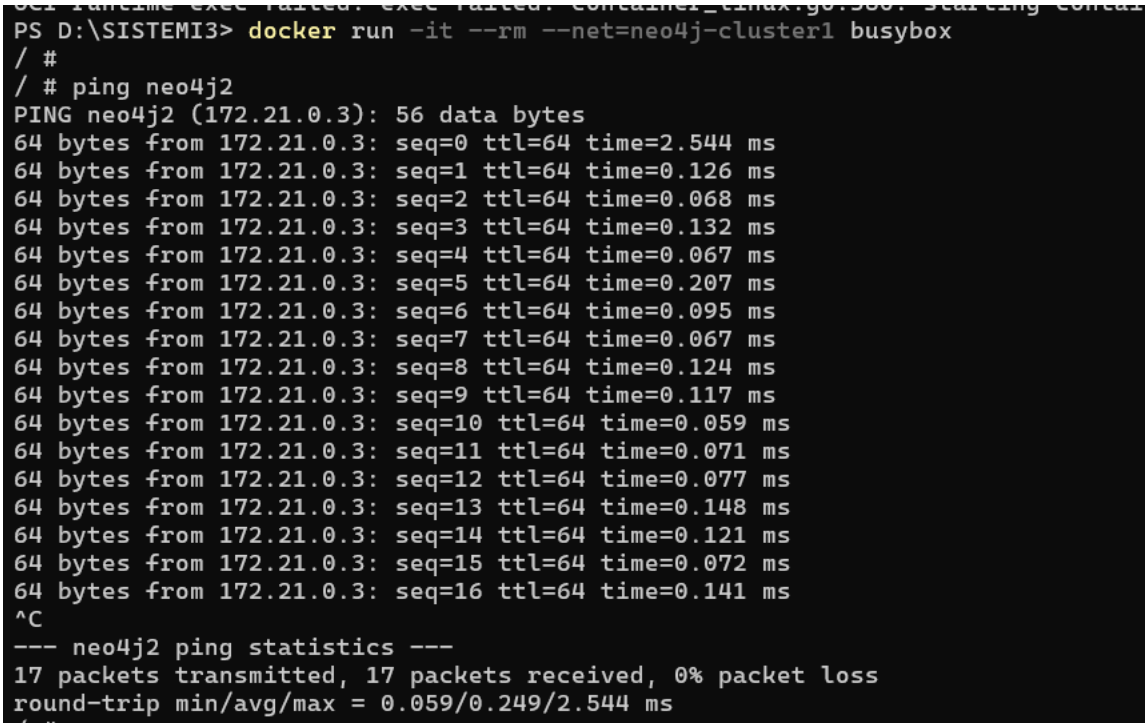
Слика 8: Приказ мреже

Провера могућности комуникације између контејнера ће бити обављена на два начина.

Коришћењем ***curl*** алата за теситрање мрежног саобраћаја:

Слика 9: Тестирање комуникације између контејнера

Покретањем ***busybox*** контејнера на истој мрежи:



Слика 10: Теситрање комуникације између контејнера

Покретање контејнера унутар докера:

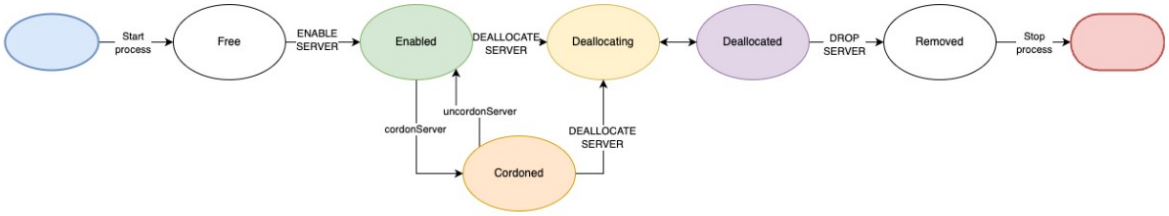
```
PS D:\SISTEMI3> docker exec -it neo4j1 cypher-shell -u neo4j
password:
Password change required
new password:
confirm password:
Connected to Neo4j using Bolt protocol version 5.4 at neo4j://localhost:7687 as user neo4j.
Type :help for a list of available commands or :exit to exit the shell.
Note that Cypher queries must end with a semicolon.
neo4j@neo4j>
```

Слика 11: Покретање контејнера у докеру

4.Управљање сервера у кластеру

Сервер у кластеру може бити у 5 различитих стања:

- Слободан – када је сервер откривен од стране откривачког сервиса, креира се у стању ,,Слободан“. Сервер у овом стању има јединствени, аутоматски генерисани ИД али нису конфигурисани
- Омогућен - Сервер у стању ,,Слободан“ мора експлицитно да буде омогућен како би се сматрао активним чланом кластера.
- Деалокација - Када сервер више није потребан, не може се уклонити из кластера док хостује било које базе података. У овом стању се све његове хостоване базе преусмеравају на друге сервере у кластеру
- Деалоциран – У овом стању сервер не хостује никакве базе података осим системских и може се уклонити из кластера
- Изолован – Сервери у овом стању неће бити коришћени за хостовање база. За разлику од стања деалокоције, сервери у овом стању не губе већ постојеће басе
- Избачен – Након што се база нађе у стању деалокације, и хостује само системску базу, безбедно је да се избаци из кластера



Слика 12: Стања сервера у кластеру

Команда **SHOW SERVERS** приказује сервере у кластеру:

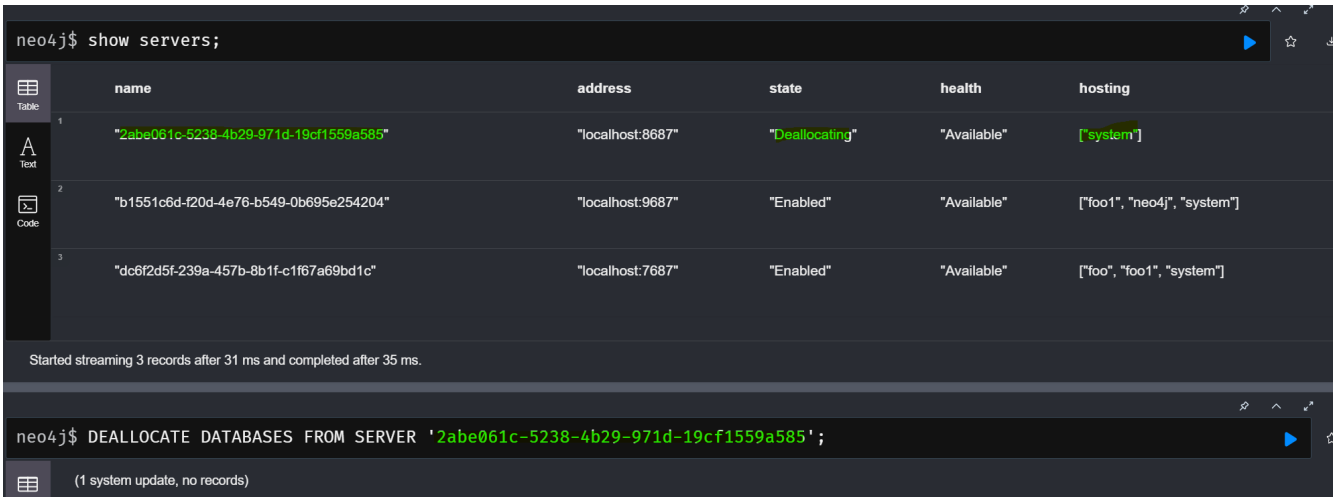
```
PS D:\SISTEMI3 PROBA> docker exec -it server3 cypher-shell
Failed to connect to neo4j://localhost:7687, fallback to bolt://localhost:7687
Connected to Neo4j using Bolt protocol version 5.4 at bolt://localhost:7687.
Type :help for a list of available commands or :exit to exit the shell.
Note that Cypher queries must end with a semicolon.
@neo4j> SHOW SERVERS;
+-----------------------------------------------------------------------------------------------------------+
| name                                 | address          | state     | health      | hosting             |
+-----------------------------------------------------------------------------------------------------------+
| "2abe061c-5238-4b29-971d-19cf1559a585" | "localhost:8687" | "Enabled" | "Available" | ["system"]          |
| "b1551c6d-f20d-4e76-b549-0b695e254204" | "localhost:9687" | "Enabled" | "Available" | ["neo4j", "system"] |
| "dc6f2d5f-239a-457b-8b1f-c1f67a69bd1c" | "localhost:7687" | "Enabled" | "Available" | ["system"]          |
+-----------------------------------------------------------------------------------------------------------+

3 rows
ready to start consuming query after 534 ms, results consumed after another 42 ms
```

Слика 13:Приказ сервера у кластеру

Могуће је додавање новог севера у кластер, кластер се креира на стандардни начин, а када се кластер креира, преко креираног ИД се додаје у кластер наредбом:

neo4j@neo4j> **ENABLE SERVER** 'ID SERVERA';

Уклањање сервера из кластера се ради у два корака, деалокација и избацивање. Деалоцирање сервра се врши командом **DEALLOCATE DATABASES** чији је параметар име сервера:



Слика 14: Деалокација база сервера

Када се базе података деалоцирају са сервера, важно је водити рачуна о топологији за сваку базу података како би се осигурало да у кластеру остане довољно сервера који могу задовољити топологије сваке базе података. Покушај да се деалоцира база са сервера што би резултирало мањим бројем доступних сервера него што је потребно, не успева и резултира грешком, а никакве промене се не извршавају.

Након што се база деалоцира, могуће је избацити сервер. Избацивање сервера се врши комадом

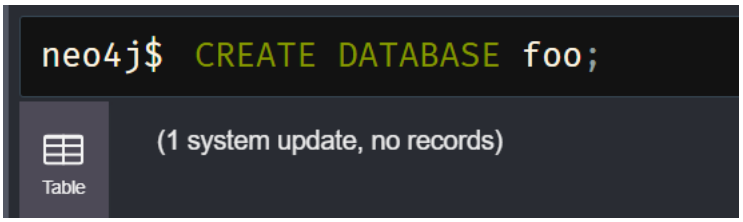
DROP SERVER чији је аргумент име сервера.

Могућа је и контрола метаподатака сервера коришћењем команде **ALTER SERVER 'name' SET OPTIONS { option: value }**

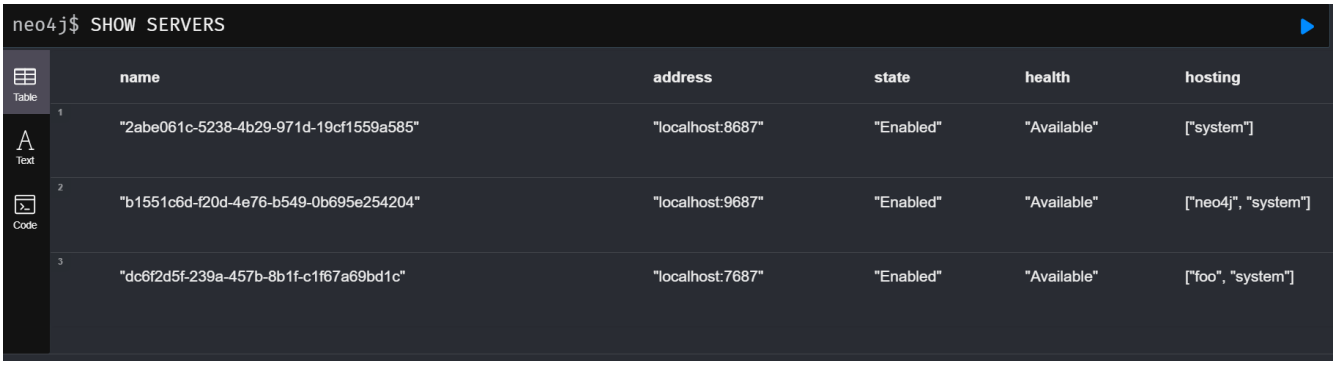
Неки од могућих метаподатака за манипулаицју су режим рада базе, дозвољене и забрањене базе као и тагови сервера.

5.Управљање базама у кластеру

Додавање базе у сервер командом **CREATE DATABASE foo;**

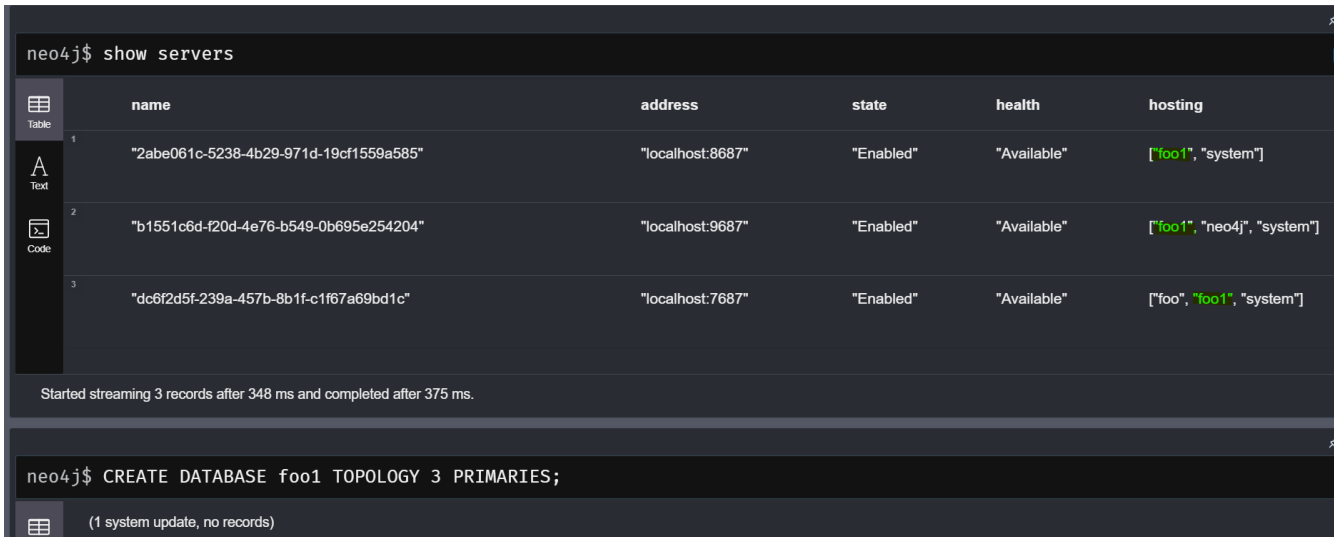


Слика 15: Креирање базе на једном серверу



Слика 16: Приказ сервиса након команде

Додавање базе у кластер се врши комадом **CREATE DATABASE foo1 TOPOLOGY 3 PRIMARIES;** уз специфицирање топологије. Команда може бити успешно извршена само ако сервери кластера задовољавају специфицирану топологију. Ако не, команда резултира грешком. На пример, ако постоје само два сервера, команда неће успети и резултираће грешком.



Слика 17: Приказ сервиса након команде

Могуће је изменити топологију или приступ базе командом **ALTER DATABASE.** Као и команда **CREATE DATABASE**, ова команда резултира грешком ако кластер не садржи довољно сервера да задовољи захтевану топологију. Поред тога, команда ALTER DATABASE је опционо идемпотентна и такође резултира грешком ако база података не постоји. Могуће је додати команду са IF EXISTS како би се осигурало да се не врати грешка ако база података не постоји. Када постоји више од једне могуће пермутације специфициране топологије, Neo4j користи алокатор да одлучи како да распореди базу података у кластеру. Ако команда ALTER DATABASE смањује број алокација базе података, алокације на изолованим серверима се прво уклањају.

Слика 18: Приказ сервиса након команде

6.Literatura

[1] - https://neo4j.com/docs/operations-manual/current/clustering/

[2] - https://hub.docker.com/_/neo4j

[3] - https://neo4j.com/dcs/operations-manual/current/docker/