Raport

Instytutu Automatyki i Informatyki Stosowanej

Politechniki Warszawskiej

Symulacyjna ocena efektywności i skalowalności energooszędnego systemu komputerowego dla centrum przetwarzania danych.

Antonina Krajewska

E-mail: antonina.krajewska@nask.pl

Raport wykonany w ramach projektu badawczego NCN, numer 2015/17/B/ST6/01885

Warszawa, wrzesień 2018

1 Wstęp

Niniejszy raport zawiera wyniki prac dotyczących symulacyjnej oceny efektywności i skalowalności systemu bazodanowego. Przeprowadzone prace dotyczyły analizy wydajności klastra, pełniącego rolę magazynu danych w systemie wczesnego ostrzegania przed zagrożeniami teleinformatycznymi w sieciach IT. Ze względu na rolę opisanego systemu, szczegóły dotyczące jego architektury oraz profilu obciążenia nie zostały opisane w raporcie. Jako główny wskaźnik wydajności klastra przyjęto średni czas oczekiwania na odpowiedź na zapytanie bazodanowe. Krótszy czas oczekwiania na odpowiedź na zapytania wpływa na obniżenie kosztów energetycznych działania centrum obliczeniowego.

1.1 Cel i zakres raportu

Pierwszy etap badań stanowiły testy wydajnościowe umożliwiające porównanie dwóch strategii rozkładania obciążenia i szeregowania zadań obliczeniowych w klastrze MongoDB. W tym celu zaprojektowano i wdrożono dwa klastry MongoDB, które zamiennie pełniły rolę magazynu danych w analizowanym systemie. Na podstawie wyników testów wydajnościowych wybrano architekturę klastra NoSQL.

W drugim etapie prac przeprowadzono symulacyjne testy odporności na awarię klastra zaprojektowanego zgodnie z wybranym algorytmem alokacji zasobów oraz szeregownia zadań obliczeniowych.

1.2 Układ raportu

Raport składa się z 8 rozdziałów. Pierwszy rozdział zawiera wprowadzenie dotyczące przeprowadzonych badań. Rozdział 2. zawiera opis konfiguracji środowiska testowego oraz charakterystykę danych testowych. Rozdział 3. dotyczy wykorzystanych narzędzi diagnostycznych i testujących, a rozdział 4. opisuje analizowane metryki. Rozdziały 5., 6. i 7. zawierają kolejno scenariusz, wyniki i wnioski z przeprowadzonych testów wydajnościowych. W rozdziale 8. opisano symulacyjne testy odporności na awarię klastra zaprojektowanego zgodnie z wybraną strategią rozkładania obciężenia.

2 Konfiguracja środowiska testowego

Testy przeprowadzono w środowisku testowym opisanego systemu wczesnego ostrzegania o cyberzagrożeniach. W ramch prac utworzono dwa klastry MongoDB:

- klaster typu *replica set*
- klaster typu *sharded*

W trakcie testów każdy z klastrów był zintegrowany z aplikacją, pełniąc rolę magazynu danych systemu.

2.1 Klaster typu *replica set*

Klaster typu *replica set* był skonfigurowany jako zbiór trzech replik MongoDB. Składał się z trzech instancji *mongod* rozłożonych na trzech maszynach wirtualnych: *mongodb1*, *mongodb2* oraz

mongodb3. Każda z maszyn wirtualnych była uruchomiona na oddzielnej maszynie fizycznej. Proces *mongod* na serwerze *mongodb1* pełnił rolę nadrzędną *(węzła primary*), a instancje uruchomione na maszynach *mongodb2* i *mongodb3* pełniły rolę podrzędną *(węzłów secondary)*.

Maszyny *mongodb1*, *mongodb2*, *mongodb3* miały następujące parametry:

- CPU: 16 rdzeni
- RAM 128 GB
- 1,9TB partycja na pliki bazy danych w pamięci SSD
- System operacyjny: Linux Debian Stretch 9

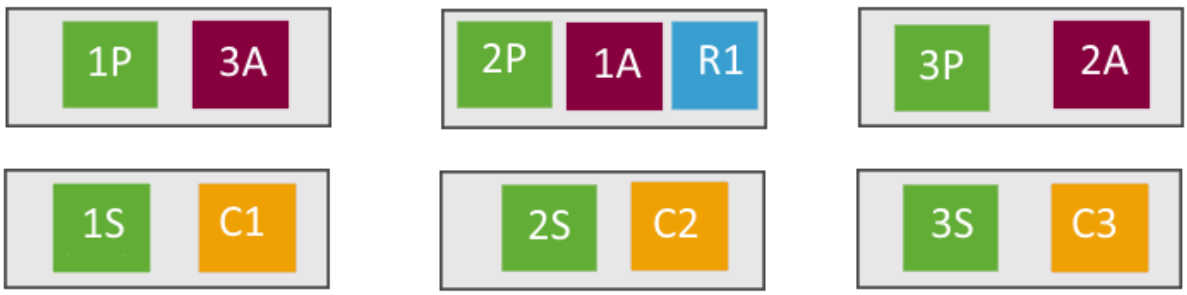
Na każdej z maszyn zainstalowano oprogramowanie:

- mongodb-org-server 3.6.5
- mongodb-org-shell 3.6.5
- mongodb-org-tools 3.6.5

Serwisy mongod działały na ustawieniach domyślnych. Wyjątkiem była wielkość pamięci podrecznej silnika WiredTiger. Z uwagi na różnice w wolumenie wygenerowanych danych, został on zmniejszony do 1GB.

2.2 Klaster typu *sharded*

Klaster typu *sharded* został wdrożony na sześciu maszynach maszynach wirtualnych *mongodb4* – *mongodb9*. Każda z maszyn wirtualnych znajdowała się na oddzielnej maszynie fizycznej. Klaster składał się z trzech shardów wdrożonych jako zbiór replik. Każdy zbiór składał się z dwóch replik oraz jednego procesu *mongod* o roli arbitra. Serwer konfiguracyjny został wdrożony jako zbiór replik mongodb składający się z jednego węzła podrzędnego oraz dwóch węzłów podrzędnych. Poniższy schemat przedstawia rozmieszczenie elementów partycjonowanego klastra na maszynach wirtualnych.



Rozmieszczenie elementów klastra typu sharded na maszynach wirtualnych.

Wszystkie maszyny miały następujące parametry:

- CPU: 16 rdzeni
- RAM: 128 GB

- 1,1TB partycja na pliki bazy danych w pamięci SSD
- System operacyjny: Linux Debian Stretch 9

Na każdej z maszyn zainstalowano oprogramowanie:

- mongodb-org-server 3.6.5
- mongodb-org-shell 3.6.5
- mongodb-org-tools 3.6.5
- mongodb-org-mongos 3.6.5

Wszystkie procesy mongod działały na ustawieniach domyślnych . Wyjątkiem była wielkość pamięci podręcznej silnika WiredTiger, która została zmniejszona do 1GB.

2.3 Charakterystyka danych testowych

W celu przeprowadzenia testów utworzono testową bazę danych zgodną z modelem danych w opisanym systemem wczesnego ostrzegania o cyberzagrożeniach. Testowe dane zostały wegenerowane przez dedykowane narzędzia generujące alarmy w aplikacji.

Poniższa tabela przedstawia charakterystykę danych w klastrze typu *replica set*

Przestrzeń nazw	**Liczba dokumentów**	**Wielkość danych (GB)**
Baza danych	146832551	78.00
Kolekcja_A	54235855	29.59
Kolekcja_B	36313357	17.41

Poniższa tabela przedstawia charakterystykę danych w klastrze typu *sharded*

Przestrzeń nazw	**Shard 1**		**Shard 2**		**Shard3**	
	Liczba dokumentów	**Wielkość danych (GB)**	**Liczba dokumentów**	**Wielkość danych (GB)**	**Liczba dokumentów**	**Wielkość danych (GB)**
Baza danych	81978746	47.84	32799281	15.71	33064977	15.22
Kolekcja_A	18119590	9.88	18165275	9.91	18164952	9.90
Kolekcja_B	12382271	6.55	12205054	4.64	11671796	5.00

W obu klastrach indeksy założone na poszczególnych kolekcjach były takie same

3 Narzędzia diagnostyczne i testujące

Testy wydajności poszczególnych konfiguracji baz zostały przeprowadzone za pomocą dedykowanych narzędzi wytworzonych przez zespół programistów NASK. Do wytworzenia ruchu sieciowego powodujące obciążenie sytemu zaimplementowano generatory alarmów systemu. Generatory wykorzystywały narzędzie *hping3*, a ich diałanie generatorów opierało się na wysyłaniu pakietów zgodnch z regułami IDS skonfigurowanymi w systemie.

Głównym wskaźnikiem wydajności obu klastrów był średni czas oczekiwania na wynik wybranych zapytań AQL. Pomiar powyższego wskaźnika był przeprowadzony za pomocą dedykowanego narzędzia *AQL Benchmark Tool* – symulującego pracę użytkowników systemu. AQL Benchmark

Tool wysyła żądania HTTP zawierające zapytania AQL komponentu dostępowego aplikacji oraz mierzy czas oczekiwania na odpowiedź dla każdego zapytania.

Ponadto wykorzystano narzędzie konsolowe *mongostat*[1] do obserwacji przepustowości bazy MongoDB.

4 Metryki

Podstawowym wskaźnikiem wydajności klastra MongoDB był średni czas oczekiwania na odpowiedź na wybrane zapytania AQL.

Ponadto monitorowano statystyki związane z przepustowością klastra MongoDB:

- liczbę operacji typu insert na sekundę
- liczbę operacji typu update na sekundę
- liczbę operacji typu query na sekundę

5 Scenariusz testów wydajnościowych

Testy wydajnościowe opierały się na pomiarze czasu oczekiwania na odpowiedź wybranych zapytań AQL adresowanych do klastra obciążonego przez operacje bazodanowe generowane przez system w trakcie pracy generatorów alarmów.

W celu przeprowadzenia testów utworzono testowe zapytania AQL. Ze względu na złożoność zostały one podzielone na dwie grupy: zapytania proste oraz zapytania złożone typu. Szczegółowy opis testowych zapytań znajduje się podrozdziale 7.3. Dla każdej grupy testowych zapytań przeprowadzono osobny test według poniższego scenariusza.

5.1 Warunki początkowe:

Dla każdej grupy zapytań AQL wykonywano następujące kroki:

1) w celu usunięcia danych z pamięci podręcznej silnika WiredTiger wywoływano poniższe polecenie na każdej z maszyn wchodzących w skład klastra

```
sysctl -w vm.drop_caches=1
```

2) uruchamiano dedykowane generatory alarmów,

3) pamięć podręczna silnika WiredTiger osiągała domyślny próg zajętości – 80%,

4) uruchamiono generator zapytań AQL Benchmark Tool.

5.2 Testowe zapytania

W celu porównania wydajności obu klastrów wygenerowano testowe zapytania AQL. Każde z zapytań AQL odpowiadoło jednemu zapytaniu MongoDB. Zapytania zostały podzielone na dwie grupy:

A) Zapytania proste

B) Zapytania złożone

A) Zapytania proste

1 https://docs.mongodb.com/manual/reference/program/mongostat/

Na potrzeby badań wygenerowano trzy typy zapytań prostych. Ze względu na model uprawnień systemu wczesnego ostrzegania o cyberzagrożeniach, dla każdego typu generowano dwa zapytania MongoDB odpowiadające zapytaniom generowanym przez użytkowników o roli *administrator* lub *użytkownik*. Poniżej przedstawiono testowe zapytania MongoDB wygenerowane dla użytkownika o roli *administrator*.

```
1. db.kolekcja_A.aggregate([{'$match': {'time_start', '$gte': TIMESTAMP},
{'source_ip_num' : '0017105511'}, {'$limit': 100}])

2. db.kolekcja_A.aggregate([{'$match': {'time_start', '$gte': TIMESTAMP},
{'source_port' :  SOURCE_PORT}, {'$limit': 100}])

3. db.kolekcja_B.aggregate([{'$match': {'timestamp' : {'$gte': TIMESTAMP}},
{'event_type':'ids_detected'}, {'$group': {'count': {'$sum': 1}, '_id':
{u'source_ip': '$source_ip'}}}, {'$limit': 1001}])
```

Poniżej przedstawiono testowe zapytania MongoDB wygenerowane dla użytkownika o roli *użytkownik*.

```
1.db.kolekcja_A.aggregate([{'$match': {'time_start', '$gte': TIMESTAMP},
{'source_ip_num' : '0017105511'}, {'$limit': 100}, {'institution_ids', {'$in':
INSTITUTIONS_LIST})])

2.db.kolekcja_A.aggregate([{'$match': {'time_start', '$gte': TIMESTAMP},
{'source_port' : SOURCE_PORT}, {'$limit': 100}, {'institution_ids', {'$in':
INSTITUTIONS_LIST})])

3. db.kolekcja_B.aggregate([{'$match': {'timestamp' : {'$gte': TIMESTAMP}},
{'event_type':'ids_detected'}, {'institution_ids': {'$in': INSTITUTIONS_LIST}},
{'$group': {'count': {'$sum': 1}, '_id': {u'source_ip': '$source_ip'}}},
{'$limit': 1001}])
```

Zmienna TIMESTAMP miała rozkład $T_0 + N(120, 60)$. Gdzie T_0 to ustalona wartość, a $N(120, 60)$ oznacza rozkład normalny o średniej 120 i odchyleniu standardowym 60.

Zmienna SOURCE_PORT miała rozkład jednostajny w zbiorze (1736, 1737, 1738, 1739, 2048, 2047, 2046).

Na potrzeby testów wygenerowano po 100 zapytań każdego typu.

B) Zapytania złożone typu

Na potrzeby testów wygenerowano cztery rodzaje zapytań złożonych typu. Dla każdego typu generowano dwa zapytania MongoDB odpowiadające zapytaniom generowanym przez użytkowników o roli *administrator* lub *użytkownik*. Poniżej przedstawiono testowe zapytania MongoDB wygenerowane dla użytkownika o roli *administrator*.

```
1.  db.kolekcja_B.aggregate([{'$match': {'timestamp' : {'$gte': TIMESTAMP}},
{'event_type':'ids_detected'}, {'$group': {u'count': {'$sum': 1}, '_id':
{u'source_ip': u'$source_ip'}}}, {'$limit': 1001}, {'$project': {u'count': 1,
```

```
'_id': 0, u'source_ip': u'$_id.source_ip'}}])

2. db.kolekcja_A.aggregate([{'$match':{'time_start' : {'$gte': TIMESTAMP}},
{'$group': {'_id': {'cc': '$cc', 'source_ip': '$source_ip'}}}, {'$project':
{'cc': '$_id.cc', 'source_ip': '$_id.source_ip'}}, {'$group': {'count':
{'$sum': 1}, '_id': {'cc': '$cc'}}}, {'$project': {'cc': '$_id.cc', 'count':
1}}, {'$sort':{'count': -1}}, {'$limit': 10}])


3. db.kolekcja_B.aggregate([{'$match': {'timestamp':{'$gte': TIMESTAMP}},
{'event_type':'ids_detected'}}, {'$group': {'_id': {'msg': '$msg'}, 'threat':
{'$sum': 1}}}, {'$project': {u'msg': u'$_id.msg', u'threat': 1}}, {'$sort':
{'threat': -1}, {'$limit': 10}])
```

Poniżej przedstawiono testowe zapytania MongoDB wygenerowane dla użytkownika o roli *użytkownik.*

```
1. db.kolekcja_B.aggregate([{'$match': {'timestamp' : {'$gte': TIMESTAMP}},
{'event_type':'ids_detected'}, {'institution_ids':{'$in': INSTITUTIONS_LIST}},
{'$group': {u'count': {'$sum': 1}, '_id':{u'source_ip': u'$source_ip'}}},
{'$limit': 1001}, {'$project': {u'count': 1, '_id': 0, u'source_ip':
u'$_id.source_ip'}}])

2. db.kolekcja_A.aggregate([{'$match':{'time_start' : {'$gte': TIMESTAMP},
{'institution_ids':{'$in': INSTITUTIONS_LIST}}, {'$group': {'_id': {'cc':
'$cc', 'source_ip': '$source_ip'}}}, {'$project': {'cc': '$_id.cc',
'source_ip': '$_id.source_ip'}}, {'$group': {'count': {'$sum': 1}, '_id':
{'cc': '$cc'}}}, {'$project': {'cc': '$_id.cc', 'count': 1}}, {'$sort':
{'count': -1}}, {'$limit': 10}])


3. db.kolekcja_B.aggregate([{'$match': {'timestamp':{'$gte': TIMESTAMP}},
{'event_type':'ids_detected'}, {'institution_ids':{'$in': INSTITUTIONS_LIST}},
{'$group': {'_id': {'msg': '$msg'}, 'threat': {'$sum': 1}}}, {'$project':
{u'msg': u'$_id.msg', u'threat': 1}}, {'$sort': {'threat': -1}, {'$limit':
10}])
```

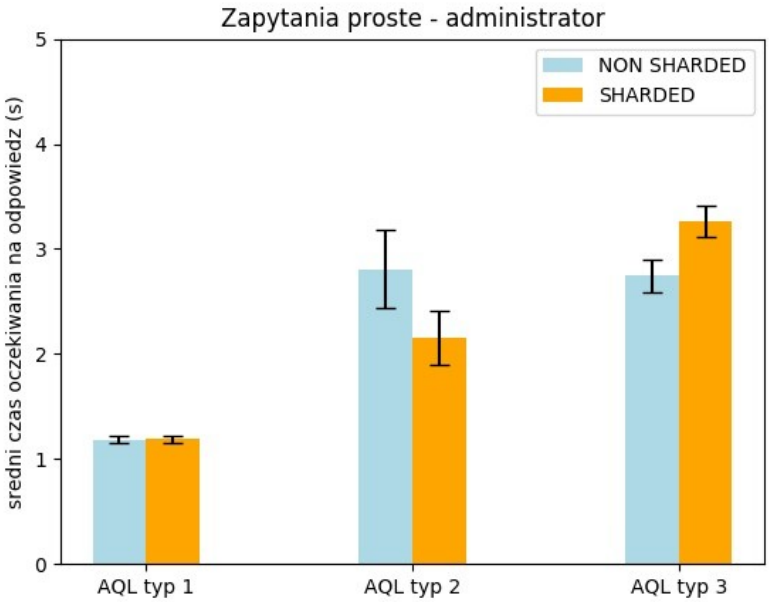
Zmienna TIMESTAMP miała rozkład $T_0 + N(120, 60)$, gdzie T_0 ma ustaloną wartość, a $N(0, 60)$ oznacza rozkład normalny o średniej 300 i odchyleniu standardowym 60.

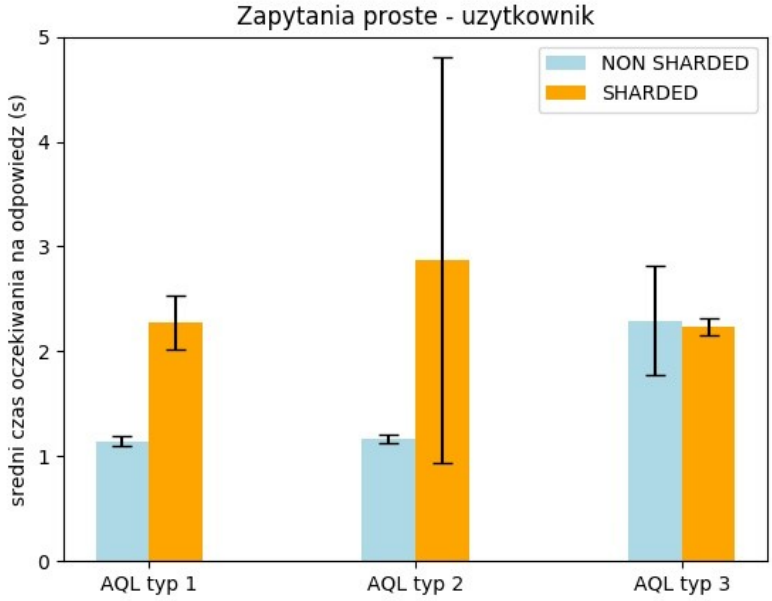
Liczebność każdej grupy wynosiła 100 zapytań.

6 Wyniki testów wydajnościowych

6.1 Wyniki testów wydajnościowych dla zapytań prostych

Poniżej przedstawiono wyniki testów wydajnościowych dla zapytań prostych w zależności od roli użytkownika (*administrator* vs. *użytkownik*)



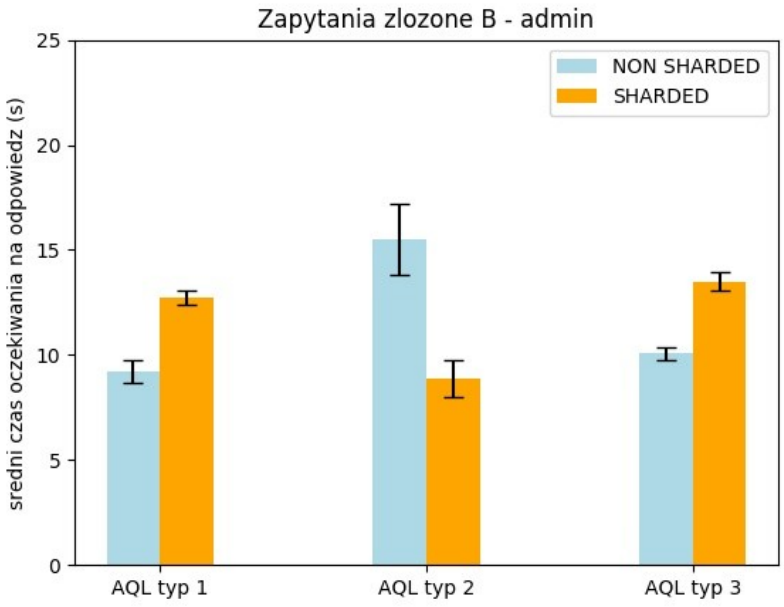


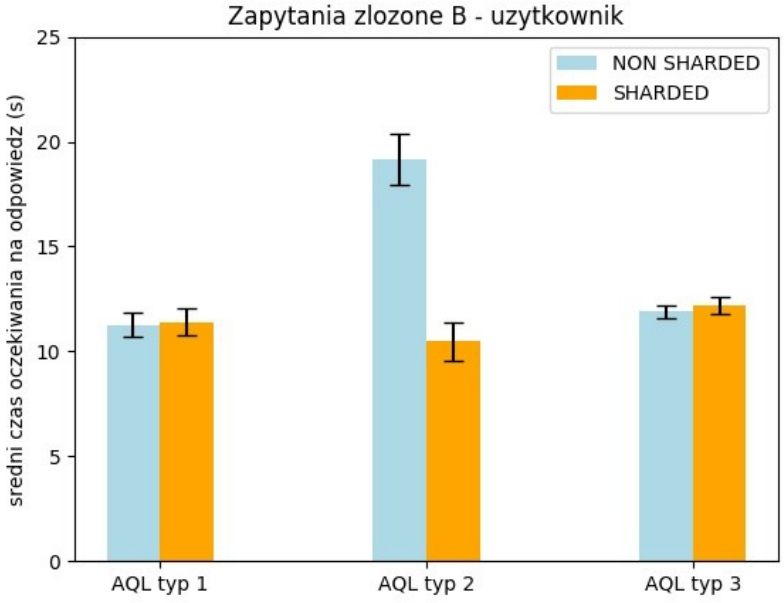
Typ zapytania	ADMINISTRATOR				UŻYTKOWNIK			
	NON SHARDED		SHARDED		NON SHARDED		SHARDED	
	μ(s)	σ(s)	μ(s)	σ(s)	μ(s)	σ(s)	μ(s)	σ(s)
AQL typ 1	1.19	0.04	1.19	0.04	1.15	0.05	2.27	0.26
AQL typ 2	2.81	0.37	2.15	0.26	1.16	0.04	2.87	1.93
AQL typ 3	2.74	0.16	3.26	0.15	2.29	0.52	2.23	0.08

Dla wszystkich trzech typów zapytań generowanych dla roli *administrator*, średni czas oczekiwania na odpowiedź był podobny dla obu klastrów.

W przypadku zapytań generowanych dla roli *użytkownik* średni czas oczekiwania na odpowiedź były wyższy dla klastra typu *sharded* dla zapytań typu 1 oraz typu 2.

6.1 Zapytania złożone





Typ zapytania	ADMINISTRATOR				UŻYTKOWNIK			
	NON SHARDED		SHARDED		NON SHARDED		SHARDED	
	μ(s)	σ(s)	μ(s)	σ(s)	μ(s)	σ(s)	μ(s)	σ(s)
AQL typ 1	9.19	0.54	12.71	0.33	11.25	0.57	11.40	0.64
AQL typ 2	15.49	1.67	8.89	0.87	19.67	1.24	10.47	0.93

AQL typ 3	10.08	0.31	13.50	0.46	11.89	0.28	12.19	0.40

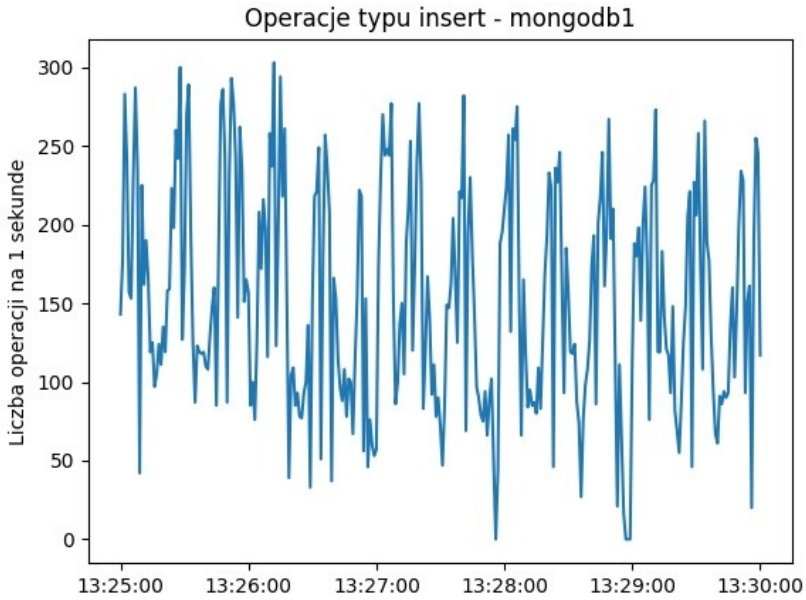
Średni czas oczekiwania na odpowiedź na zapytania typu 2 generowanych dla obu ról był znacznie niższy w klastrze typu *sharded*. W przypadku zapytań typu 1 i 3 średni czas oczekiwania na odpowiedź był podobny dla roli użytkownik oraz niższy w klastrze typu *replica set*.

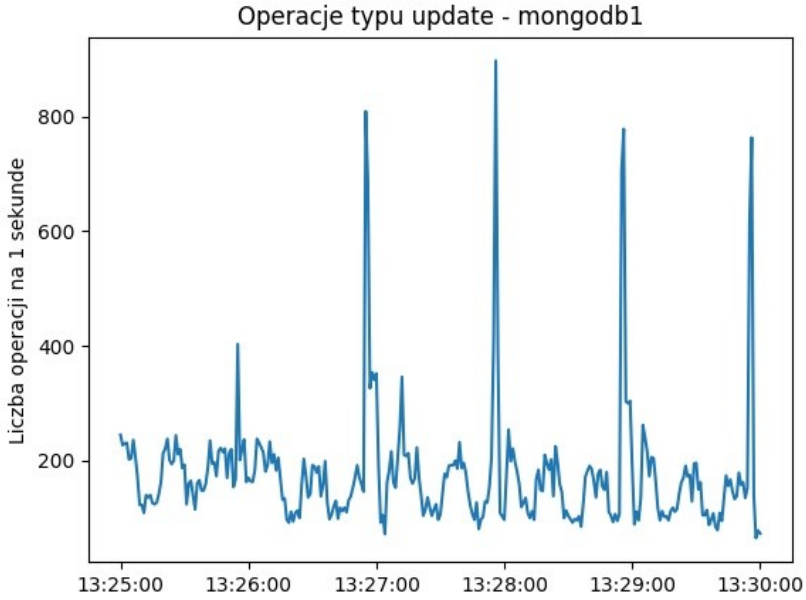
6.2 Zapytania generowane przez system wczesnego ostrzegania o cyber zagrożeniach

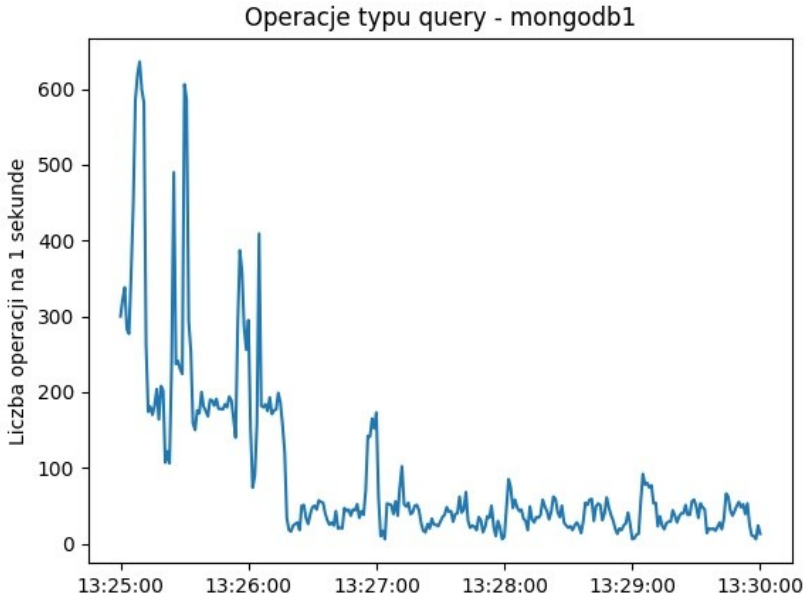
Poniżej przedstawiony wyniki pomiaru metryk związanych z przepustowością obu klastrów w trakcie pracy generatorów alarmów systemu:

- liczby operacji typu insert na sekundę
- liczby operacji typu update na sekundę
- liczby operacji typu query na sekundę

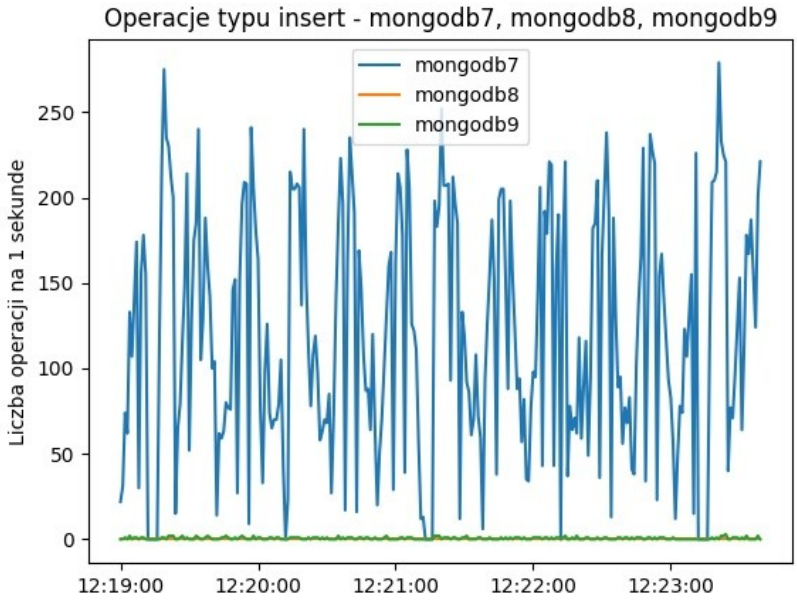
Poniższe wykresy przedstawiają wyniki pomiaru na serwerze mongodb1 pełniącym rolę *primary* w klastrze typu *replica set.*

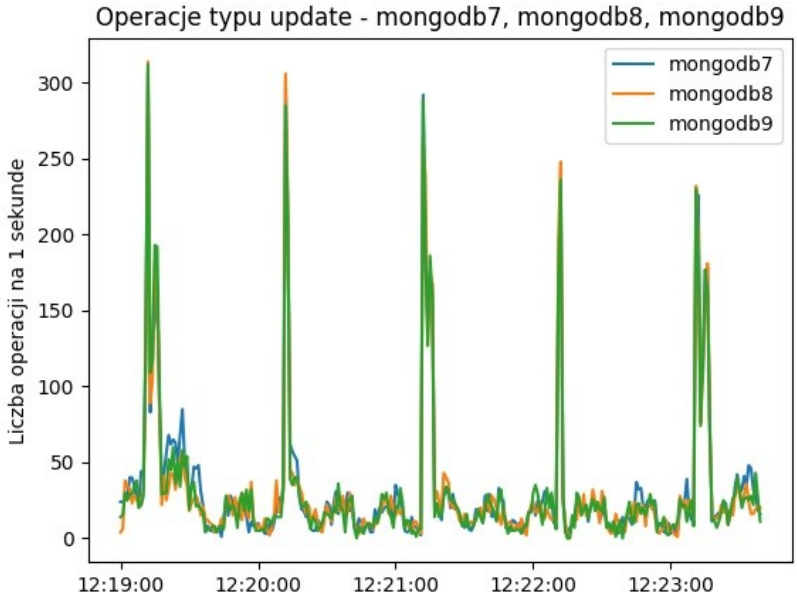


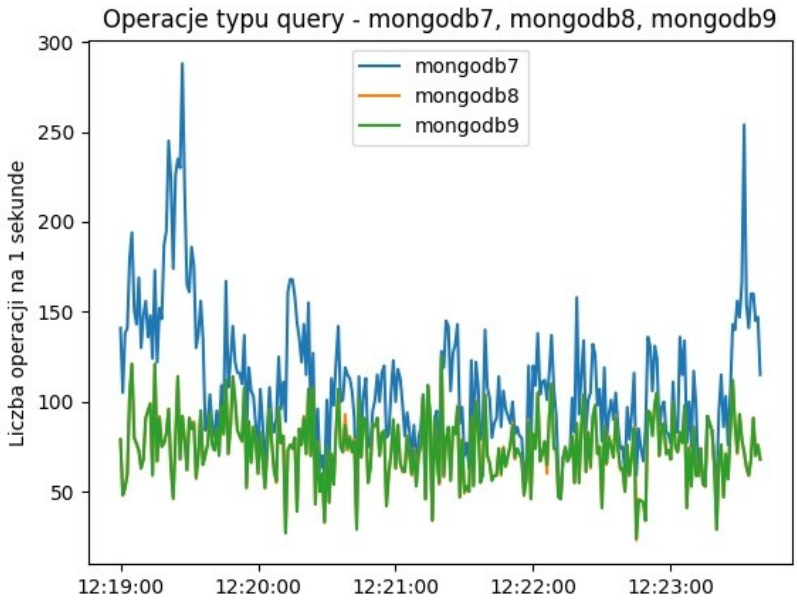




Poniższe wykresy przedstawiają wyniki pomiaru na serwerach *mongodb7*, *mongodb8*, *mongodb*9 pełniących rolę *primary* w klastrze typu *sharded*.


Operacje typu insert - mongodb7, mongodb8, mongodb9
mongodb7
mongodb8
mongodb9
Liczba operacji na 1 sekunde
0
50
100
150
200
250
12:19:00
12:20:00
12:21:00
12:22:00
12:23:00


Operacje typu update - mongodb7, mongodb8, mongodb9
mongodb7
mongodb8
mongodb9
Liczba operacji na 1 sekunde
0
50
100
150
200
250
300
12:19:00
12:20:00
12:21:00
12:22:00
12:23:00



7 Wnioski i rekomendacje

Głównym wskaźnikiem wydajności obu klastrów był średni czas oczekiwania na odpowiedź na wybrane zapytania AQL. Wyniki testów dla zapytań prostych wykazują, że czas oczekiwania na odpowiedź był niższy dla klastra typu *replica set*. Dla złożonych zapytań czas oczekiwania na odpowiedź był istotnie niższy dla jednego rodzaju zapytania dla klastra typu sharded. Dla pozostałych typów zapytań średni czas oczekiwania był porównywalny dla obu klastrów.

Wybór klucza oraz strategii shardingu jest ściśle związany z profilem obciążenia w bazie danych MongoDB. Zły dobór klucza oraz strategii partycjonowania może negatywnie wpłynąć na wydajność bazy. Dane wygenerowane na potrzeby testów wydajnościowych były zgodne z formatem systemu wczesnego ostrzegania o cyber zagrożeniach, jednak ich charakterystyka może się różnić od danych produkcyjnych. Ponadto wielkość wolumenu wygenerowanych danych była istotnie mniejsza od zbioru danych produkcyjnych. Z tego względu wybór klastra typu sharded na tym etapie byłby przedwczesną optymalizacją.

8 Testy odporności na awarie

Poniżej przedstawiono wyniki testów odporności na awarie dla rekomendowanego klastra typu *replica set.*

8.1 Warunki początkowe dla testów odpornościowych

Klaster bazy danych jest poprawnie skonfigurowany, węzły klastra są zsynchronizowane. Klaster zawiera dane w formacie zgodnym z systemem wczesnego ostrzegania o cyber zagrożeniach.

8.2 Scenariusz testów odpornościowych

Testy odporności na awarię przeprowadzono symulując możliwe awarie klastra bazy danych.

Scenariusz 1.

Usługa *mongod* na maszynie *mongodb2* pełniącą rolę *secondary* została zatrzymana.

Zbiór replik MongoDB traktuje ten węzeł jako niedostępny *(not reachable/healthy).* Klaster kontynuuje swoje działanie, ale przestaje być odporny na awarię jednego z węzłów. Po ponownym włączeniu, węzeł *mongodb2* powinien ponownie dołączyć się do klastra i zsynchronizować dane.

Scenariusz 2.

Usługa *mongod* na maszynie *mongodb1* pełniącej rolę *primary* została zatrzymana. Zbiór replik MongoDB traktuje ten węzeł jako niedostępny *(not reachable/healthy).* Jeden z węzłów *mongodb2, mongodb3* pełniących rolę *secondary* zaczyna pełnić rolę *primary*. Po ponownym włączeniu, węzeł *mongodb1* powinien ponownie dołączyć się do klastra i zsynchronizować dane.

Scenariusz 3.

Usługa *mongod* została zatrzymana na dwóch z trzech węzłów klastra. Żaden węzeł nie pełni roli *primary,* próba wykonania operacji zapisu kończy się błędem. Po ponownym uruchomieniu węzłów klaster działa poprawnie.

Scenariusz 4.

Usługa *mongod* na została zatrzymana na wszystkich węzłach klastra. Klaster został wyłączony. Po ponownym uruchomieniu maszyn, klaster można odtworzyć, bez utraty spójności danych.

8.3 Rezultaty testów testów odpornościowych

Klaster MongoDB typu *replica set* przeszedł pom wszystkie symulowane scenariusze awarii.

Bibliografia

1: Pramod J. Sadalage, Martin Fowler, NoSQL. Kompendium wiedzy, 2011
2: Kristina Chodorow, MongoDB: The Definitive Guide, 2nd EditionPowerful and Scalable Data Storage, 2013