Konfiguracja i diagnostyka

Poziom: średni **Dotyczy: Apache Cassandra 3.x / 4.x** **Czas: ~30 min**

Większość plików konfiguracyjnych Cassandry leży w `/etc/cassandra/conf/` *—* `cassandra.yaml` *opisuje sam węzeł i jego udział w klastrze,* `cassandra-env.sh` *opisuje JVM, na której węzeł chodzi.*

Skróty użyte na tej stronie (2)

Skrót	Co znaczy
GC	Garbage Collector: mechanizm Javy sprzątający nieużywaną pamięć
JVM	Java Virtual Machine: maszyna wirtualna, w której działają programy w Javie

Z życia wzięte

Z ŻYCIA WZIĘTE

Nowy węzeł cass04, dokładany do klastra przed sezonem świątecznym w listopadzie, odmawia dołączenia i w logu systemowym zostawia komunikat o niezgodnej nazwie klastra. Sprawdzamy cassandra.yaml i widzimy literówkę w cluster_name, wpisaną przy kopiowaniu konfiguracji z cass03. Po poprawieniu nazwy węzeł dołącza do klastra i zaczyna streamować swój fragment danych zgodnie z nowym podziałem tokenów, tuż przed zakazem zmian w infrastrukturze obowiązującym od połowy listopada.

cassandra.yaml — najważniejsze parametry

Pojęcie: cassandra.yaml. Główny plik konfiguracyjny węzła, wczytywany raz przy starcie usługi — większość parametrów w nim wymaga restartu `cassandra.service`, żeby zmiana zaczęła obowiązywać.

Parametr	Znaczenie
`cluster_name`	nazwa klastra — węzeł odmawia dołączenia do klastra, jeśli nazwa nie zgadza się z tą zapisaną w jego lokalnym stanie
`seeds`	lista adresów IP węzłów startowych, od których nowy węzeł uczy się o reszcie klastra przez gossip
`listen_address`	adres, na którym węzeł nasłuchuje ruchu wewnątrzklastrowego (gossip, streaming)
`rpc_address`	adres, na którym węzeł przyjmuje połączenia klienckie (`cqlsh`, sterowniki aplikacji)
`endpoint_snitch`	implementacja snitcha opisanego na stronie Architektura Cassandry: pierścień, węzeł, replikacja — w produkcji zwykle `GossipingPropertyFileSnitch`
`data_file_directories`	lista katalogów, w których węzeł trzyma pliki SSTable opisane na stronie Utrzymanie klastra: repair, compaction, czyszczenie; może być więcej niż jeden, na osobnych dyskach
`commitlog_directory`	katalog na commitlog — zalecany na osobnym, szybkim dysku niezależnym od `data_file_directories`, bo oba są zapisywane niemal ciągle
`concurrent_reads` / `concurrent_writes`	liczba wątków obsługujących równolegle operacje odczytu i zapisu; wartości domyślne dobrane pod dyski wirujące, na SSD zwykle podnoszone
`hinted_handoff_enabled`	włącza albo wyłącza mechanizm hinted handoff opisany na stronie Architektura Cassandry: pierścień, węzeł, replikacja

> **Pojęcie: memtable w konfiguracji.** Rozmiar i sposób alokacji pamięci pod memtable, opisaną na stronie Utrzymanie klastra: repair, compaction, czyszczenie, sterują parametry `memtable_heap_space_in_mb` i `memtable_offheap_space_in_mb` — ich suma ogranicza, ile danych świeżo zapisanych może leżeć w pamięci, zanim wymuszony zostanie zrzut do SSTable niezależnie od tego, czy limit rozmiaru pojedynczej memtable został osiągnięty.

Zmiana nazwy klastra w konfiguracji: cassandra.yaml

```
cluster_name: 'NazwaKlastra'
```

Zmiana samej wartości w pliku nie wystarcza do bezpiecznej zmiany nazwy działającego klastra — pełną procedurę, uwzględniającą stan zapisany lokalnie przez każdy węzeł, opisuje strona Zmiana nazwy klastra i datacenter.

cassandra-rackdc.properties — przynależność do racku i DC

```
dc=NazwaDatacenter
rack=NazwaRacku
```

Plik czytany przez `GossipingPropertyFileSnitch` przy starcie węzła, deklarujący, do którego racku i datacenter, opisanych na stronie Architektura Cassandry: pierścień, węzeł, replikacja, węzeł należy — ta informacja jest później propagowana do reszty klastra przez gossip.

cassandra-env.sh i ustawienia JVM

> **Pojęcie: cassandra-env.sh.** Skrypt powłoki wykonywany przy starcie usługi, budujący parametry uruchomieniowe maszyny wirtualnej Javy, na której działa proces `cassandra` — rozmiar sterty (`MAX_HEAP_SIZE`, `HEAP_NEWSIZE`) i wybór oraz strojenie odśmiecacza pamięci (GC). Zbyt mała sterta prowadzi do częstych, krótkich pauz GC; zbyt duża — do rzadszych, ale znacznie dłuższych pauz przy pełnym, generacyjnym GC, bo przejrzenie większej sterty zajmuje więcej czasu niezależnie od algorytmu.

Plik jest też miejscem na tymczasowe dyrektywy `JVM_OPTS` używane w procedurach eksploatacyjnych — na przykład przy zmianie nazwy racku albo datacenter działającego węzła, opisanej na stronie Zmiana nazwy klastra i datacenter, gdzie tymczasowo dopisujemy:

```
JVM_OPTS="$JVM_OPTS -Dcassandra.ignore_rack=true -Dcassandra.ignore_dc=true"
```

Taką dyrektywę traktujemy jako rozwiązanie tymczasowe — zostawiona na stałe ukrywa realne niezgodności topologii, które w normalnych warunkach powinny zatrzymać start węzła jako sygnał błędu konfiguracji, nie coś do trwałego zignorowania.

Logi

Pojęcie: system.log. Główny log procesu `cassandra`, domyślnie w `/var/log/cassandra/system.log` — komunikaty o starcie i zatrzymaniu usługi, dołączeniu i odłączeniu węzłów widziane przez gossip, ostrzeżenia o dużych partycjach i dużej liczbie tombstone'ów, błędy kompaktowania i repair. Pierwsze miejsce, w które zaglądamy przy jakimkolwiek nietypowym zachowaniu węzła.

Pojęcie: debug.log. Log tego samego procesu na poziomie szczegółowości `DEBUG`, domyślnie również w `/var/log/cassandra/` — włączany, gdy `system.log` nie daje wystarczającego kontekstu do zdiagnozowania problemu; z racji objętości nie trzymamy go włączonego permanentnie na produkcji bez rotacji dopasowanej do dostępnego miejsca na dysku.

Pojęcie: gc.log. Log pracy Garbage Collectora JVM, konfigurowany w `cassandra-env.sh` — surowe dane o każdej pauzie GC, jej długości i ilości zwolnionej pamięci, uzupełniające skumulowane statystyki dostępne przez `nodetool gcstats` opisane na stronie nodetool: stan klastra i operacje administracyjne.

Typowe objawy i ich przyczyny

Objaw	Najczęstsza przyczyna	Gdzie sprawdzamy
Węzeł widoczny jako `DN` w `nodetool status`	brak łączności gossipowej mimo działającego procesu	`nodetool info` na podejrzanym węźle, kolumna `Gossip active`
Wysoka liczba pending compactions	tempo zapisu przewyższa tempo kompaktowania, zbyt niski limit przepustowości	`nodetool compactionstats`, `nodetool tpstats`
Długie pauzy GC	zbyt mała albo zbyt duża sterta, nieodpowiedni algorytm GC do rozmiaru danych	`nodetool gcstats`, `gc.log`
Ostrzeżenie o liczbie tombstone'ów w logu	częste `DELETE` bez odpowiednio krótkiego `gc_grace_seconds` albo bez cyklicznego repair	`system.log`, `nodetool tablehistograms`
`OperationTimedOut` w `cqlsh`	przeciążony coordinator albo repliki, zbyt wysoki poziom spójności jak na aktualny stan klastra	`nodetool tpstats`, `nodetool proxyhistograms`

Ostrzeżenie o tombstone'ach pojawia się w `system.log` w formie zbliżonej do `Read X live rows and Y tombstone cells` — mechanizm i koszt tombstone'a opisuje strona Model danych: keyspace, tabela, klucz partycji, klucz klastrujący, próg, po którym Cassandra zaczyna ostrzegać, jest

konfigurowalny (`tombstone_warn_threshold`), a próg, po którym odrzuca zapytanie zamiast je wykonać, jest ustawiony wyżej (`tombstone_failure_threshold`).

`OperationTimedOut` zwracany klientowi przez `cqlsh` albo sterownik aplikacji oznacza, że coordinator nie otrzymał w wymaganym czasie odpowiedzi od tylu replik, ile wymagał ustawiony poziom spójności opisany na stronie Architektura Cassandry: pierścień, węzeł, replikacja — samo w sobie to nie mówi, czy operacja się nie wykonała, czy tylko odpowiedź nie zdążyła wrócić na czas; przy zapisie warto to rozróżnić, zanim ponowimy operację.

Uruchamianie i zatrzymywanie węzła

Zatrzymanie usługi z uprzednim wypięciem węzła

```
sudo nodetool drain
sudo systemctl stop cassandra.service
```

Wykonujemy `drain`, opisany na stronie nodetool: stan klastra i operacje administracyjne, przed zatrzymaniem usługi zawsze, gdy zatrzymanie jest planowane (aktualizacja systemu, restart maszyny) — dzięki temu dane z memtable trafiają na dysk przed zamknięciem procesu, zamiast polegać wyłącznie na commitlogu przy kolejnym starcie.

Uruchomienie usługi

```
sudo systemctl start cassandra.service
```

Po starcie węzeł odtwarza ewentualne niezapisane dane z commitlogu, dołącza do gossipu i zaczyna być widoczny w `nodetool status` u pozostałych węzłów — proces ten trwa zwykle kilkanaście do kilkudziesięciu sekund, zależnie od rozmiaru commitlogu do odtworzenia.

Kolejność przy restarcie całego klastra

Restart wielu węzłów naraz zamiast po kolei zdejmuje z klastra jednocześnie tyle replik, że przy niekorzystnym doborze może przekroczyć replication factor i chwilowo uniemożliwić spełnienie wymaganego poziomu spójności dla części danych. Restart klastra wykonujemy więc węzeł po węźle: zatrzymujemy jeden węzeł (z uprzednim `drain`), czekamy, aż wróci i będzie widoczny jako `UN` u pozostałych węzłów w `nodetool status`, i dopiero wtedy przechodzimy do kolejnego. Przy klastrze wielo-DC zachowujemy tę samą zasadę osobno w obrębie każdego datacenter.

Częste błędy

Zmiana `cluster_name` bez pełnej procedury. Sama edycja pliku i restart usługi kończy się błędem niezgodności nazwy klastra z lokalnym stanem węzła — pełna procedura na stronie Zmiana nazwy klastra i datacenter.

Restart wszystkich węzłów klastra naraz. Zdejmuje jednocześnie zbyt wiele replik z klastra — operacje wymagające poziomu spójności silniejszego niż `ONE` mogą w tym oknie zacząć zwracać błędy, mimo że żaden pojedynczy węzeł formalnie nie uległ awarii.

Ignorowanie ostrzeżeń o tombstone'ach jako „tylko ostrzeżenia". Ostrzeżenie w logu poprzedza zwykle o tygodnie moment, w którym te same zapytania zaczną przekraczać `tombstone_failure_threshold` i zwracać błąd zamiast wyniku — traktujemy je jako sygnał do przeprojektowania wzorca usuwania danych, nie do zignorowania.

□□ Ćwiczenia

> Pytania sprawdzają dobór właściwego narzędzia diagnostycznego do objawu, nie wymagają uruchomionego klastra.

Pytania kontrolne

1. Aplikacja zaczęła dostawać `OperationTimedOut` przy zapisach z poziomem spójności `QUORUM`, mimo że wszystkie węzły są widoczne jako `UN`. Od czego zaczynamy diagnozę?

Odpowiedź

Od `nodetool tpstats` na węzłach pełniących rolę coordinatora dla tych zapisów — sprawdzamy, czy `MutationStage` ma rosnącą kolejkę `Pending`, co oznaczałoby przeciążenie, oraz od `nodetool gcstats`, bo długie pauzy GC na jednej z replik potrafią same w sobie spowodować

przekroczenie czasu oczekiwania na odpowiedź, mimo że węzeł formalnie jest aktywny.

2. W `system.log` pojawia się ostrzeżenie o dużej liczbie tombstone'ów odczytanych dla jednej partycji. Jakie dwie niezależne przyczyny warto sprawdzić?

Odpowiedź

Wzorzec zapisu i usuwania w aplikacji (częste `DELETE` na tej samej partycji, np. kolejka, z której elementy są usuwane po przetworzeniu) oraz to, czy repair wykonuje się cyklicznie w oknie krótszym niż `gc_grace_seconds` — bez repair tombstone'y nie znikają w tempie, w jakim powstają, mimo że mechanizm kompaktowania działa poprawnie.

Powiązane: Architektura Cassandry: pierścień, węzeł, replikacja · Utrzymanie klastra: repair, compaction, czyszczenie · nodetool: stan klastra i operacje administracyjne · Zmiana nazwy klastra i datacenter · Model danych: keyspace, tabela, klucz partycji, klucz klastrujący

Źródła: dokumentacja Apache Cassandra, rozdz. „Configuration" · przykładowe pliki `cassandra.yaml` i `cassandra-env.sh` z dystrybucji

---

Revision #42
Created 2026-09-24 23:34:58 UTC by Tomasz Konieczny
Updated 2026-10-26 14:13:19 UTC by Tomasz Konieczny