

# Apache Cassandra 2.0

## Przewodnik instalacji i konfiguracji w systemie Debian Wheezy

Jan Baranowski, Michał Kaik  
Politechnika Poznańska

18 czerwca 2014

### 1 Wstęp

Niniejszy dokument ma za zadanie przedstawić proces instalacji i konfiguracji serwera baz danych NoSQL *Apache Cassandra* w środowisku rozproszonym. Na potrzeby demonstracji zakłada się że środowisko to będzie składać się z kilku węzłów połączonych siecią lokalną.

*Apache Cassandra* jest serwerem baz danych NoSQL, początkowo rozwijanym przez *Facebooka* na potrzeby umożliwienia efektywnego wyszukiwania wiadomości w skrzynce odbiorczej. Obecnie (od marca 2009 roku) *Cassandra* jest rozwijana przez *Apache Foundation* (jako jeden z projektów top-level) i stanowi podstawę dla zestawu narzędzi *DataStax*.

*Cassandra* powstała jako narzędzie mające w założeniu cechować się:

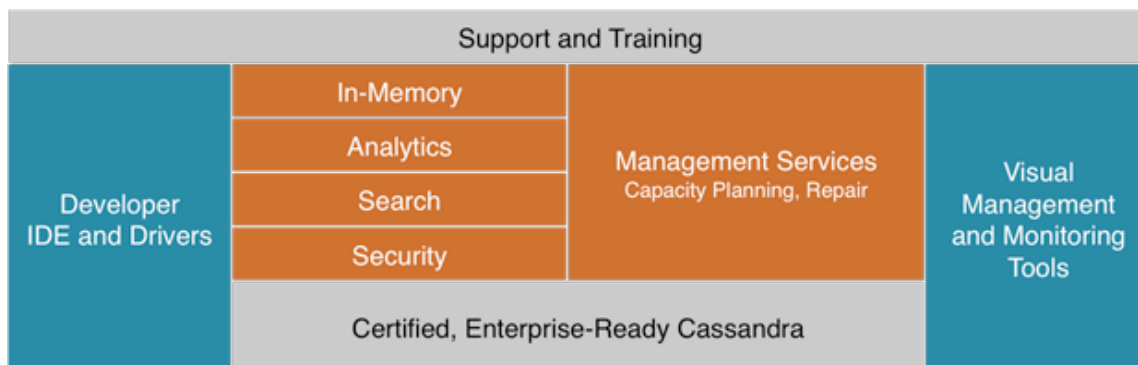
- wysoką dostępnością (*Cassandra* jest określana jako zawsze zapisywalna baza danych)
- niskim opóźnieniem wykonywanych operacji (*ang.* latency)
- odpornością na awarie (możliwością replikacji danych, brakiem komponentów, których awaria może zdestabilizować system (*ang.* single points of failure))
- możliwością regulacji kompromisu pomiędzy szybkością działania a odpornością na awarie i spójnością replik
- relatywnie prostym modelem danych

Zarówno opisanie kolumnowego modelu danych wykorzystywanego w *Cassandrze*, jak i mechanizmów, dzięki którym spełnia ona założenia projektowe nie jest celem tego dokumentu. Autorzy mogą jedynie podać propozycje publikacji, które opisują wspomniane zagadnienia. Zarówno dla modelu danych, jak i budowy wewnętrznej *Cassandry* będzie to przede wszystkim dokumentacja udostępniana przez firmę *DataStax* [4]. Kolumnowy model danych, przedstawiony z perspektywy osób pracujących z bazami relacyjnymi został doskonale (choć może zbyt obszernie) opisany w *Cassandra — The Definitive Guide* [3]. Z kolei architektura *Cassandry* w dużym skrócie (choć zapewne w sposób wystarczający by służyć jako wstęp do tego dokumentu) została przedstawiona w prezentacji [2].

Procedurę przygotowania klastra przedstawiono na przykładzie systemu operacyjnego Debian GNU/Linux 6, ponieważ uchodzi on za jedno z lepszych rozwiązań dla serwerów.

## 2 Wybór dystrybucji

Na samym początku nowi użytkownicy *Cassandra* stają przed wyborem dystrybucji tego oprogramowania. Możliwości są dwie: instalacja standardowej wersji dostarczonej przez *Apache Foundation* oraz instalacja platformy *DataStax* — zestawu narzędzi obudowujących *Cassandra* i dostarczających m.in. funkcje pozwalające na uproszczenie zarządzania klastrem, wizualne monitorowanie, analizę obciążeń węzłów, itp. (por. rysunek 1).



Rysunek 1: architektura *DataStax* w odniesieniu do *Cassandra*, źródło: [15]

Wybór jest ważny o tyle, że pomimo wspólnego fundamentu, jakim jest *Cassandra*, procesy instalacji obu dystrybucji nie mają ze sobą nic wspólnego, tj. (według wiedzy autorów) nie da się doinstalować do dystrybucji *Apache* platformy *DataStax*.

W tym miejscu należy wspomnieć o twórcach platformy — firmie o zaskakującej nazwie *DataStax*, zajmującej się dostosowaniem *Cassandra* do potrzeb przedsiębiorstw poprzez m.in. rozwój i testowanie bazy danych, dostarczanie narzędzi ułatwiających administrację systemem, organizację szkoleń, certyfikację personelu technicznego, itd. Bodaj najbardziej znaczącym wkładem *DataStax* w rozwój projektu jest opracowanie szeregu driverów/konektorów dla różnych języków programowania, dzięki którym programiści mogą korzystać z *Cassandra* w sposób analogiczny do rozwiązań relacyjnych (np. tak jak w przypadku *MySQL Connectors*, por. [11]) i rozbudowa dokumentacji technicznej platformy, włączając w to dokumentację plików konfiguracyjnych, architektury *Cassandra* i języka *CQL* (odpowiednika *SQL* dla kolumnowych baz danych).

Na potrzeby niniejszego dokumentu założono, że właściwym wyborem będzie dystrybucja *Apache Cassandra*, ze względu na to, że dokument ma pełnić rolę wprowadzenia do technologii, nie zaś pozwalać na błyskawiczne wdrożenie systemu (*ang.* rapid deployment). Poza tym, przewodniki instalacji i konfiguracji *DataStaxa* dostępne są np. na stronie [12].

## 3 Instalacja

*Cassandra* zostanie zainstalowana przy użyciu narzędzia *APT* z repozytorium *Apache Software Foundation*. Wymagane będzie dodanie odpowiedniego repozytorium do listy źródeł *APT*a. Dodatkowo ze względu na to, że *Cassandra* napisana jest w Javie, pokazany zostanie proces instalacji *Oracle JRE* w systemie *Debian* (ta maszyna wirtualna jest zalecana przez twórców *Cassandra*).

### 3.1 Wybór maszyny wirtualnej Java

*Apache Software Foundation* dostarcza pakiety zawierające *Cassandra* w formacie *\*.deb*, a także repozytorium dla *APT*a. Jedną z zależności pakietu *cassandra* jest pakiet *openjdk-7-jre*. Jest to w pewnym sensie sprzeczne z zaleceniami twórców bazy danych, ponieważ ci rekomendują maszynę wirtualną *Oracle* jako środowisko uruchomieniowe (nawet w testowanej wersji bazy 2.0 odpowiedni

komunikat wyświetlany jest w logu). Nie zmienia to faktu, że testy (funkcjonalne, lecz nie wydajnościowe) przeprowadzone na potrzeby tego artykułu udowodniły, że *Cassandra* działa bez zarzutu także w środowisku *OpenJDK*.

Należy zatem wybrać pomiędzy stosowaniem się do zaleceń a prostotą instalacji.

## 3.2 Instalacja i konfiguracja Oracle Java

Pakiet z *OpenJDK* jest instalowany jako pakiet zależny podczas instalacji *Cassandry*. Jeżeli jednak administrator zdecyduje się użyć *Oracle JVM*, poniżej przedstawiona jest skrótowa procedura instalacji tego oprogramowania w systemie *Debian*. Stanowi ona kompilację najprostszych rozwiązań i tym różni się od większości przewodników dostępnych w internecie, że poza konsolą systemu nie wymaga żadnych dodatkowych narzędzi (np. mechanizmu transferu plików z maszyny-terminala do maszyny-serwera).

Pierwszym krokiem jest pobranie pakietu oprogramowania (JRE, nie JDK) ze strony *Oracle*. Standardowo by pobrać plik należy zaakceptować umowę licencyjną, jednak przy odpowiedniej konfiguracji możliwe jest ominięcie tego kroku [9]:

Listing 1: pobieranie *Oracle JRE*

```
$ wget --no-cookies \  
> --no-check-certificate \  
> --header "Cookie: oraclelicense=accept-securebackup-cookie" \  
> "http://download.oracle.com/otn-pub/java/jdk/7u60-b19/jre-7u60-linux-i586.tar.gz" \  
> -O /tmp/jre-7u60-linux-i586.tar.gz
```

Popularnym sposobem instalacji pobranego oprogramowania jest wypakowanie archiwum (zazwyczaj do katalogu */opt*) i ręczna konfiguracja ścieżki systemowej (por. [9]). *Debian* dostarcza jednak narzędzie pozwalające przekonwertować archiwum do pakietu *DEB* [10].

Należy je zainstalować, a potem użyć:

Listing 2: budowa pakietu *DEB* z *Oracle JRE*

```
$ apt-get install java-package  
  
$ fakeroot make-jpkg /tmp/jre-7u60-linux-i586.tar.gz
```

Powstały w ten sposób pakiet *DEB* należy zainstalować. Po instalacji należy ustawić *Oracle JVM* jako domyślną maszynę wirtualną w systemie.

Listing 3: instalacja i konfiguracja *Oracle JRE*

```
$ dpkg -i /tmp/oracle-j2re1.7_1.7.0+update60_i386.deb  
  
$ update-alternatives --config java  
There are 2 choices for the alternative java (providing /usr/bin/java).  
  
  Selection Path Priority Status  
-----  
  0 /usr/lib/jvm/java-7-openjdk-i386/jre/bin/java 1051 auto mode  
* 1 /usr/lib/jvm/j2re1.7-oracle/bin/java 316 manual mode  
  2 /usr/lib/jvm/java-7-openjdk-i386/jre/bin/java 1051 manual mode  
  
Press enter to keep the current choice[*], or type selection number:  
  
$ java -version  
java version "1.7.0_60"  
Java(TM) SE Runtime Environment (build 1.7.0_60-b19)  
Java HotSpot(TM) Client VM (build 24.60-b09, mixed mode)
```

Według wiedzy autorów Cassandra nie wymaga ustawiania zmiennej `JAVA_HOME` dla żadnego z użytkowników (ani roota, ani użytkownika *cassandra*, właściciela demona). Jeżeli jednak zajdzie taka potrzeba, zainstalowane maszyny wirtualne można znaleźć w katalogu `/usr/lib/jvm`.

Listing 4: ustawianie `JAVA_HOME`

```
$ echo "export JAVA_HOME=/usr/lib/jvm/j2re1.7-oracle/" >> /home/<user>/.bashrc
```

### 3.3 Konfiguracja repozytorium

By móc zainstalować *Cassandrę*, należy dodać odpowiednie repozytorium *Apache Software Foundation* do źródeł programu *APT*. Zgodnie z konwencją każda większa wersja *Cassandry* znajduje się w osobnym repozytorium. Na potrzeby tego dokumentu w systemie testowym zostanie zainstalowana *Cassandra 2.0.8*.

Do pliku `/etc/apt/sources.list` należy dopisać:

Listing 5: nowe źródła pakietów dla *APT*a

```
deb http://www.apache.org/dist/cassandra/debian 20x main
deb-src http://www.apache.org/dist/cassandra/debian 20x main
```

Próba pobrania spisu zawartości repozytorium zakończy się niepowodzeniem, ponieważ *APT* nie zna kluczy publicznych dla repozytorium *Apache Software Foundation*. Te należy dodać w następujący sposób:

Listing 6: pobieranie kluczy publicznych repozytorium *ASF*

```
$ gpg --keyserver pgp.mit.edu --recv-keys F758CE318D77295D
$ gpg --export --armor F758CE318D77295D | sudo apt-key add -

$ gpg --keyserver pgp.mit.edu --recv-keys 2B5C1B00
$ gpg --export --armor 2B5C1B00 | sudo apt-key add -
```

Po zakończeniu wszystkich operacji należy pobrać spis zawartości repozytoriów:

Listing 7: odświeżanie list pakietów

```
$ apt-get update
```

### 3.4 Instalacja *Cassandry*

Ostatnim krokiem procesu instalacji jest zainstalowanie *Cassandry* z użyciem *APT*a:

Listing 8: instalacja *Cassandry*

```
$ apt-get install cassandra
```

W efekcie *Cassandra* powinna zostać zainstalowana i uruchomiona.

## 4 Konfiguracja węzła

Ta część dokumentu opisuje czynności związane z konfiguracją węzła *Cassandry* tak, by był on zdolny dołączyć do klastra. *Cassandra* jest systemem P2P, więc o utworzeniu klastra węzły decydują wspólnie w oparciu o taką samą nazwę klastra zdefiniowaną w pliku konfiguracyjnym i włączony kanał komunikacji poprzez plotkowanie. Kanał ten jest domyślnie wyłączony, zatem każdy węzeł *Cassandry* w sieci będzie z początku tworzył osobny klaster.

## 4.1 Stan systemu po instalacji

Zaraz po instalacji w systemie pojawia się nowa usługa: **cassandra**. Jest ona domyślnie uruchomiona. Jej konfiguracja zapobiega możliwości dołączenia węzła do jakiegokolwiek klastra ze względu na zablokowanie mechanizmu plotkowania (dokładnie: agent implementujący algorytm plotkujący nasłuchuje wyłącznie na adresie IP 127.0.0.1).

Pliki konfiguracyjne można znaleźć w katalogu `/etc/cassandra`. Są wśród nich:

- `/etc/cassandra/cassandra-env.sh` — skrypt definiujący zmienne środowiskowe w formie argumentów dla maszyny wirtualnej Java (np. wielkość stosu, ale także m.in. włączenie/wyłączenie mechanizmu *JMX*).
- `/etc/cassandra/cassandra-rackdc.properties` — informacja o tym, w którym fizycznie racku i centrum danych znajduje się obecny węzeł.
- `/etc/cassandra/cassandra-topology.properties` — przybliżone informacje o tym, w których rackach i centrach danych znajdują się inne węzły klastra. Plik ten jest wykorzystywany przez jedną z kilku wersji algorytmu rozmieszczania replik (*snitcha*).
- `/etc/cassandra/cassandra-topology.yaml` — plik analogiczny do poprzedniego, wykorzystywany jednak przez inną wersję algorytmu.
- `/etc/cassandra/cassandra.yaml` — główny plik konfiguracyjny *Cassandra*.

Pliki danych (w tym: commit logi) znaleźć można w katalogu `/var/lib/cassandra`. Czyszcząc zawartość trzech znajdujących się w nim podkatalogów można zresetować stan klastra. W celu wykonania tej czynności autorzy nie zalecają jednak ich całkowitego usuwania (podczas testowania mogą one zostać utworzone na nowo z prawem zapisu tylko dla roota, co uniemożliwi normalny start usługi).

Plik logu znaleźć można w `/var/log/cassandra/system.log`.

## 4.2 Jak debugować konfigurację?

Przed rozpoczęciem wprowadzania zmian do plików konfiguracyjnych zaleca się wyłączenie automatycznego uruchamiania usługi. Błędnie skonfigurowany węzeł może dołączyć do nieodpowiedniego klastra, a jego usunięcie w takim przypadku nie jest zadaniem trywialnym. Efekt można osiągnąć wykonując polecenie:

Listing 9: wyłączenie uruchamiania *Cassandra* na jej domyślnych runlevelach

```
$ insserv -r cassandra
```

Najprostszym sposobem sprawdzenia poprawności konfiguracji jest uruchomienie węzła w trybie jawnym (w przeciwieństwie do demona). Służy do tego podane poniżej polecenie, które wypisze log działania węzła na standardowe wyjście.

Listing 10: testowe uruchamianie *Cassandra*

```
$ cassandra -f # -f for foreground
```

Dodatkowo można zarchiwizować taki log poleceniem `tee plik.log`, które skopiuje standardowe wejście na standardowe wyjście i do podanego pliku:

Listing 11: testowe uruchamianie *Cassandra* z archiwizacją logu

```
$ cassandra -f | tee /tmp/cassandra_testrun.log
```

**UWAGA!** Jeżeli *Cassandra* zostanie uruchomiona w trybie jawnym podczas gdy równolegle usługa działa w tle, zamiast informacji o błędzie w logu tej pierwszej pojawi się informacja o wyjątku `NullPointerException`.

**UWAGA!** Jeżeli pierwsze uruchomienie *Cassandry* odbędzie się w trybie jawnym z poziomu użytkownika innego niż `cassandra`, usługa nie będzie miała prawa zapisu do nowo utworzonych katalogów danych, nie będzie więc w stanie się uruchomić.

**UWAGA!** Jeżeli zmienna `JAVA_HOME` dla użytkownika uruchamiającego polecenie `cassandra -f` jest ustawiona, ale prowadzi do niepoprawnie zainstalowanej maszyny wirtualnej, *Cassandra* nie uruchomi się, ale ani skrypt startowy ani log nie poinformują o błędzie.

### 4.3 Pliki konfiguracyjne

Role podstawowych plików konfiguracyjnych zostały przedstawione w punkcie 4.1. W tabeli 1 zostaną opisane parametry konfiguracyjne (z `/etc/cassandra/cassandra.yaml`), których wartości powinny zostać świadomie dobrane przed uruchomieniem pojedynczego węzła.

Konfigurację węzła ułatwia sam plik konfiguracyjny *Cassandry*, zawierający obszerne komentarze.

### 4.4 Weryfikacja stanu węzła

Szeroko pojęty stan węzła po konfiguracji można sprawdzić na kilka sposobów. Przede wszystkim po uruchomieniu usługi należy sprawdzić czy ta faktycznie działa i gdy tak nie jest, przejrzeć log.

W następnej kolejności można sprawdzić czy *Cassandra* nasłuchuje na portach zdefiniowanych w pliku konfiguracyjnym:

Listing 12: sprawdzanie na których portach nasłuchuje *Cassandra*

```
$ netstat -ln4p #listening, numeric, IPv4 only, with owner process information
```

Kolejnym sposobem jest próba połączenia się z węzłem poprzez Thrift API.

Listing 13: dostęp do *Cassandry* przez *Thrift API*.

```
$ cqlsh localhost 9160 #use "quit;" to exit CQL shell
```

Wreszcie można wyświetlić stan klastra, do którego należy węzeł:

Listing 14: sprawdzanie stanu klastra

```
$ nodetool --host localhost -p 7199 status # 7199 is the JMX port
Datacenter: datacenter1
=====
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
-- Address Load Tokens Owns (effective) Host ID Rack
UN 127.0.0.1 98.76 KB 256 100.0% d85efd66-29fb-46d7-b824-a4cc3fc4e75b rack1
```

## 5 Konfiguracja klastra

### 5.1 Łączenie węzłów w klaster

Zakładając, że odpowiednie węzły spełniają następujące warunki:

- nazwa klastra w plikach konfiguracyjnych wszystkich węzłów jest taka sama
- agent algorytmu plotkującego nasłuchuje na adresie IP interfejsu osiągalnego z sieci lokalnej (parametr `listen_address`)
- początkowe punkty kontaktowe są zdefiniowane tak by każdy węzeł mógł ostatecznie dowiedzieć się o wszystkich pozostałych

by połączyć je w klaster wystarczy jedynie uruchomić na nich usługę `cassandra`.

### 5.2 Weryfikacja konfiguracji klastra

Połączenia między węzłami klastra mają charakter dwukierunkowy. Oznacza to, że jeżeli węzeł nie jest widoczny w sieci, to (o ile działa) sam także nie widzi pozostałych węzłów.

Stąd najprostszym sposobem na sprawdzenie czy wszystkie węzły zostały odpowiednio połączone jest sprawdzenie jaki stan klastra widzi dowolny z jego węzłów. Informację tę może podać narzędzie `nodetool`.

Listing 15: sprawdzanie stanu klastra po dołączeniu węzłów

```
$ nodetool -host localhost -p 7199 status
Note: Ownership information does not include topology; for complete information,
specify a keyspace
Datacenter: DC1
=====
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
-- Address Load Tokens Owns Host ID Rack
UN 10.0.0.101 63.38 KB 256 35.3% c6b6661a-c501-45ad-acc1-3e17fea6d241 NODE1AND2
UN 10.0.0.100 63.48 KB 256 31.0% c45900c6-6866-4b5a-b32b-cc0158bca524 NODE0
UN 10.0.0.102 63.5 KB 256 33.7% 4d598617-7f3c-480c-9310-f0eae90ef96b NODE1AND2
```

Jeżeli jeden z węzłów nie dołączył do klastra lub znajduje się w stanie `DOWN`, należy sprawdzić czy uruchomił się prawidłowo, czy log nie zawiera informacji o błędzie sieci lub agenta algorytmu plotkującego (w pewnych sytuacjach taki błąd nie powstrzymuje usługi przed uruchomieniem), wreszcie czy adres IP dla agenta i początkowe punkty kontaktowe zostały poprawnie zdefiniowane w pliku konfiguracyjnym.

**UWAGA!** Z zawartości pliku konfiguracyjnego `/etc/cassandra/cassandra.yaml` można wnioskować, że zawsze jednym z punktów kontaktowych powinien być `localhost`. Jako adres IP punktu nie powinien być jednak podawany adres `127.0.0.1`, lecz adres interfejsu na którym nasłuchuje agent (parametr `listen_address`).

### 5.3 Działania dodatkowe

W obecnym stanie klaster powinien już działać prawidłowo. Każdy węzeł powinien przyjmować od użytkowników zapytania i odpowiadać na nie w sposób adekwatny do stanu klastra.

Jednak by zapewnić stabilne działanie klastra przez dłuższy okres wymagane są dodatkowe czynności administracyjne.

### 5.3.1 Synchronizacja zegarów

Dokumentacja *Cassandry* udostępniona przez *Apache Software Foundation* wspomina o tym, że do oceny czy propagowane do węzłów zmiany schematu (np. dodanie nowej rodziny kolumn) są przedawnione (zatem powinny być zignorowane) wykorzystywany jest lokalny zegar czasu rzeczywistego (*RTC*).

By zapewnić poprawność propagacji takich zmian należy okresowo synchronizować zegary węzłów np. poprzez protokół *NTP*.

W systemie *Debian* funkcję synchronizacji czasu systemowego z serwerem *NTP* dostarcza pakiet *ntp*. Pakiet ten jest rekomendowany do instalacji przez pakiet *cassandra*. Po instalacji *ntp* automatycznie zostaje uruchomiona usługa *ntp* bazująca na puli serwerów dostarczonych razem z pakietem. Należy sprawdzić czy usługa ta jest zainstalowana, uruchomiona i czy serwery *NTP* są dostępne:

Listing 16: weryfikacja poprawności działania klienta *NTP*

```
$ dpkg -l | grep ntp
ii ntp 1:4.2.6.p5+dfsg-2 i386 Network Time Protocol daemon and utility programs
$
$ service ntp status
[ ok ] NTP server is running.
$
$ ntpq -p # lists the NTP servers being queried
      remote refid st t when poll reach delay offset jitter
=====
*ntp.tktelekom.p 194.29.130.252 2 u 146 256 377 7.188 9.360 0.889
+afrodyta.comple 194.29.130.252 2 u 89 256 377 11.952 10.228 1.769
xpscolka.of.pl 80.50.231.226 2 u 35 256 377 13.027 190.872 26.614
+96-7.cpe.smnt.p 213.199.225.30 3 u 55 256 377 7.046 9.511 1.096
$
```

### 5.3.2 Czyszczenie

*Cassandra* używa trzech mechanizmów w celu zapewnienia ostatecznej spójności replik. Są to kolejno:

- 1) mechanizm *read-repair* pozwalający uaktualnić te repliki, które zostaną wyznaczone jako nieaktualne (a dzieje się to w momencie gdy pojedynczy węzeł zbierze dostatecznie dużo danych z różnych replik, tj. przy odczycie z dostatecznie wysokim poziomem spójności).
- 2) mechanizm *HintedHandoff* uprawniający punkt kontaktowy w klastrze do przechowania uaktualnienia do momentu gdy jego cel (o ile nie jest nim sam punkt kontaktowy) będzie osiągalny (tj. sam zostanie naprawiony lub połączenie z nim zacznie działać)
- 3) mechanizm *AntiEntropy* czyli wymuszona przez administratora synchronizacja koordynatora z wszystkimi zależnymi replikami

Spośród nich tylko trzeci mechanizm wymaga interwencji administratora.

*Cassandra* pozwala na usuwanie danych z bazy w sposób pod kątem zachowania spójności podobny do przeprowadzania zapisów. Jest to o tyle kłopotliwe, że podczas synchronizacji dane, które zostały usunięte mogą być potraktowane jako brakujące uaktualnienia. W efekcie dane raz usunięte mogą pojawić się ponownie nawet na tym samym węźle.

By temu zapobiec *Cassandra* zamiast faktycznie usuwać dane, oznacza je jako usunięte za pomocą specjalnej wartości, tzw. *tombstone*. To z kolei powoduje problem przyrastania danych nawet podczas ich usuwania.



By zapobiec i temu, *tombstony* w *Cassandra* są przechowywane przez określony czas (definiowany podczas tworzenia przestrzeni kluczy, poprzez parametr `gc_grace_seconds`, domyślnie 10 dni [6]).

Stąd zaleca się wymuszenie synchronizacji replik (czyli uruchomienie mechanizmu *AntiEntropy*) częściej niż wynosi najmniejsza wartość `gc_grace_seconds` dla przestrzeni kluczy przechowywanych w węźle.

**UWAGA!** Problem nie będzie występował jeżeli aplikacja korzystająca z *Cassandra* wcale nie zleca usunąć wierszy.

**UWAGA!** Z drugiej strony pełna synchronizacja węzła z replikami jest kosztowna. Konstrukcja struktury *Merkle Tree*, wykorzystywanej do obliczenia różnic pomiędzy replikami wymaga odczytu wszystkich danych przechowywanych w węźle (w tym — tych zrzucanych na dysk). Należy więc ustalić kompromis pomiędzy obciążeniem systemu *tombstonami* a obciążeniem go procedurą synchronizacji.

Mechanizm *AntiEntropy* uruchamia się wymuszając pełną naprawę węzła narzędziem `nodetool`:

Listing 17: naprawa węzła poprzez uruchomienie mechanizmu *AntiEntropy*

```
$ nodetool -host localhost repair
[2014-06-17 21:47:55,488] Nothing to repair for keyspace 'system'
[2014-06-17 21:47:55,546] Starting repair command 2, repairing 474
ranges for keyspace system_traces
[2014-06-17 22:01:37,033] Repair session 0a4463e0-f669-11e3-8ad4-251897e18719 for
range (-2577486857218927215,-2576033184180305603] finished
[2014-06-17 22:01:37,038] Repair session 0b595650-f669-11e3-8ad4-251897e18719 for
range (-1096394548677781591,-1069256055695448190] finished
[...]

[2014-06-17 22:01:38,931] Repair session efaa5290-f66a-11e3-8ad4-251897e18719 for
range (4183239765725363167,4233658233454255946] finished
[2014-06-17 22:01:38,932] Repair session f0bc5ed0-f66a-11e3-8ad4-251897e18719 for
range (-767591973226673249,-759782651038439858] finished
[2014-06-17 22:01:38,932] Repair session f1bc42a0-f66a-11e3-8ad4-251897e18719 for
range (1226781783220621999,1238805897799045860] finished
[2014-06-17 22:01:38,947] Repair session f2bee590-f66a-11e3-8ad4-251897e18719 for
range (7669322402280617470,7674157004615966047] finished
[2014-06-17 22:01:38,947] Repair command 2 finished
```

Dla zastosowań, w których wymagany jest ciągły nadzór administracyjny nad klastrem jest oczywiście możliwe wywoływanie polecenia `nodetool repair` ręcznie. Na potrzeby tego dokumentu przyjęto, że naprawy odbywać się będą tak, by tylko jeden węzeł był jednocześnie obciążany. Do wywoływania polecenia `nodetool repair` użyto `crona` (dla uproszczenia pominięto kwestie związane z uruchamianiem zadania poprzez dedykowanego użytkownika):

Listing 18: konfiguracja okresowego naprawiania klastra

```
$ echo "#!/bin/bash" > /root/repair-cassandra.sh
$ echo "nodetool -host localhost -p 7199 repair --partitioner-range" \
> >> /root/repair-cassandra.sh

$ chmod u+x /root/repair-cassandra.sh

$ crontab -e

GNU nano 2.2.6 File: /tmp/crontab.M3Zyaw/crontab Modified
```

```

# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow command
0 3 * * <node_id> /root/repair-cassandra.sh

^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell

$ ^X

$ crontab -u root -l

# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow command
1 1 * * 1 /root/repair-cassandra.sh # repairing one node per day of week;
                                   # should be enough for 3 nodes in cluster

```

**UWAGA!** Można dodatkowo zredukować obciążenie węzła podając przy naprawie parametr `-partitioner-range`. Spowoduje to naprawę jedynie tych zakresów kluczy dla których węzeł jest *master repliką* (pomijając zakresy ulokowane w węźle przez mechanizm replikacji — standardowe repliki).

**UWAGA!** Synchronizacja replik na żądanie spowoduje automatycznie usunięcie wszystkich *tombstonów* związanych z replikowaną rodziną kolumn, ponieważ po synchronizacji nie są one już potrzebne.

### 5.3.3 Konfiguracja uwierzytelniania

Włączenie uwierzytelniania klienta wymaga w pierwszej kolejności zmiany mechanizmu uwierzytelniania w pliku konfiguracyjnym `/etc/cassandra/cassandra.yaml`. Domyślnie *Cassandra* używa implementacji *AllowAllAuthenticator*, który pozwala każdemu na dostęp do klastra, jednocześnie blokując wykonywania jakichkolwiek operacji na zbiorze użytkowników:

Listing 19: wypisanie listy zdefiniowanych użytkowników przy wyłączonym uwierzytelnianiu

```
$ cqlsh
Connected to OAND1AND2CLUSTER at localhost:9160.
[cqlsh 4.1.1 | Cassandra 2.0.8 | CQL spec 3.1.1 | Thrift protocol 19.39.0]
Use HELP for help.
cqlsh> LIST USERS;
Bad Request: You have to be logged in and not anonymous to perform this request
cqlsh>
```

Zmieniając parametr `authenticator` na *PasswordAuthenticator* wyłącza się możliwość anonimowego logowania do klastra, zyskując w to miejsce możliwość zarządzania użytkownikami. Domyślny użytkownik administracyjny ma przypisany login `cassandra` i hasło `cassandra`. Podając te dane przy logowaniu zyskujemy możliwość zmiany domyślnego hasła i utworzenia standardowego użytkownika:

Listing 20: włączenie uwierzytelniania przy dostępie do klastra przez `cqlsh`

```
$ cqlsh
Connection error: Bad Request: You have not logged in

$ cqlsh -u cassandra -p cassandra
Connected to OAND1AND2CLUSTER at localhost:9160.
[cqlsh 4.1.1 | Cassandra 2.0.8 | CQL spec 3.1.1 | Thrift protocol 19.39.0]
Use HELP for help.
cqlsh> LIST USERS;

name | super
-----+-----
cassandra | True

(1 rows)

cqlsh>
```

Listing 21: konfiguracja kont użytkowników

```
$ cqlsh -u cassandra -p cassandra
Connected to OAND1AND2CLUSTER at localhost:9160.
[cqlsh 4.1.1 | Cassandra 2.0.8 | CQL spec 3.1.1 | Thrift protocol 19.39.0]
Use HELP for help.
cqlsh> ALTER USER cassandra WITH PASSWORD 'balcerowiczmusiodejcs';
cqlsh> CREATE USER scott WITH PASSWORD 'tiger' NOSUPERUSER;
cqlsh> quit;
```

Listing 22: weryfikacja konfiguracji kont użytkowników

```
$ cqlsh -u scott -p tiger;
Connected to 0AND1AND2CLUSTER at localhost:9160.
[cqlsh 4.1.1 | Cassandra 2.0.8 | CQL spec 3.1.1 | Thrift protocol 19.39.0]
Use HELP for help.
cqlsh> LIST USERS;

name | super
-----+-----
      scott | False
      cassandra | True

(2 rows)

cqlsh>
```

Oczywiście włączenie uwierzytelniania ma wpływ na wszystkie kanały komunikacji z klastrem, w tym na drivery *DataStax*.

## 6 Zarządzanie klastrem

### 6.1 Monitoring węzłów i klastra

Początkowo *Cassandra* przekazywała szereg statystyk, głównie dot. wydajności (np. opóźnienie odczytów i zapisów, liczba uruchomień procesów kompaktowania struktur danych, użycie pamięci podręcznej, itd.) do systemu monitorowania klastrów *Ganglia* [1] [7].

Obecnie *Cassandra* do tego celu wykorzystuje technologię *JMX*, która może być traktowana jako odpowiednik *SNMP* dla serwerów pisanych w *Javie*, dodatkowo wyposażony w mechanizm RPC.

Absolutna większość narzędzi do monitorowania klastrów potrafi komunikować się z agentami *JMX* (np. *Nagios*, *Munin*). Istnieją też konwertery/mosty pomiędzy technologiami, pozwalające odpytywać agenta *JMX* przez np. REST (*JMX-to-REST*, <http://code.google.com/p/polarrose-jmx-rest-bridge/>) czy *SNMP* (*jmx2snmp*, <https://github.com/tcurdt/jmx2snmp>).

Istnieje też narzędzie pozwalające dodać do *Cassandry* konsolę WWW wyświetlającą metryki udostępniane przez agenta.

By dodać taką konsolę należy poprać narzędzie *MX4J* (<http://mx4j.sourceforge.net/>), wypakować zawartość pobranego archiwum, a następnie dodać archiwum *mx4j-tools.jar* do ścieżki systemowej *Javy* (*classpath*) definiowanej na potrzeby *Cassandry*. Standardowo by to zrobić, wystarczy skopiować plik *mx4j-tools.jar* do katalogu */usr/share/cassandra/lib/*.

By zdefiniować adres i port na którym ma nasłuchiwać konsola WWW, należy przekazać dodatkowe parametry do maszyny wirtualnej. Stąd do pliku */etc/cassandra/cassandra-env.sh* należy dopisać następujące linijki:

Listing 23: konfiguracja *mx4j* dla *Cassandry*

```
JVM_OPTS="$JVM_OPTS -Dmx4jaddress=0.0.0.0"
JVM_OPTS="$JVM_OPTS -Dmx4jport=8081"
```

By sprawdzić czy narzędzie zostało załadowane, należy zrestartować usługę *cassandra* i przeszukać plik logu pod kątem odpowiedniego wpisu. Oczywiście można też sprawdzić czy *Cassandra* nasłuchuje na odpowiednim porcie.

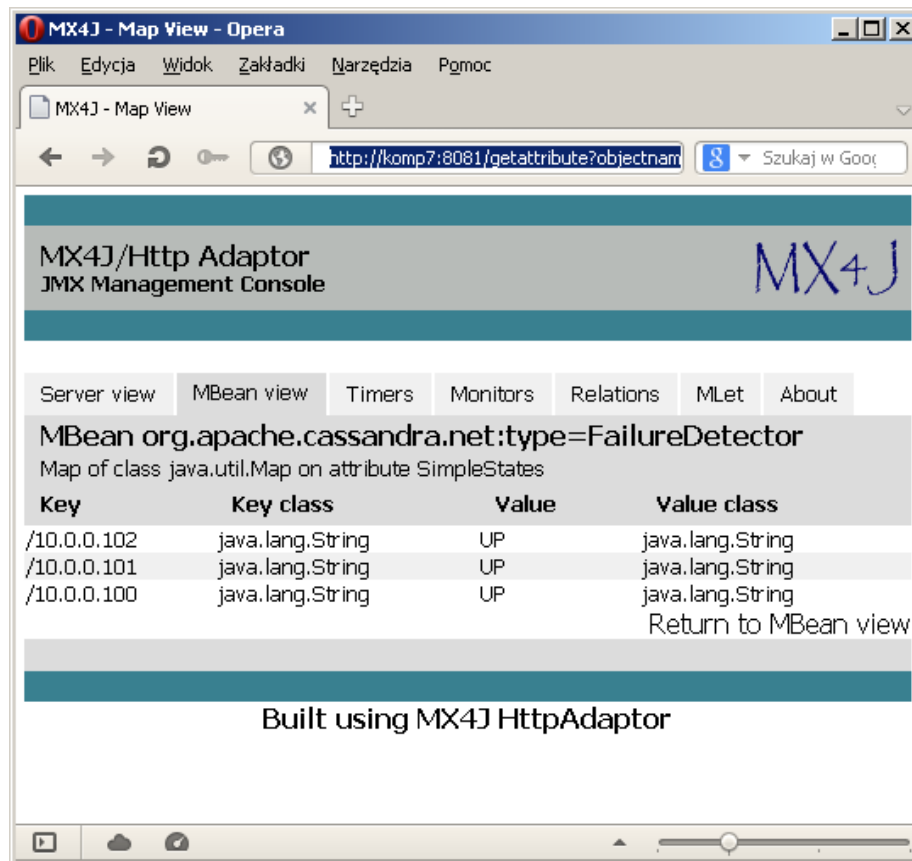
Listing 24: weryfikacja poprawności działania *mx4j*

```
$ cat /var/log/cassandra/system.log | grep mx4j
[...]
```

```
INFO [main] 2014-06-18 00:47:06,506 Mx4jTool.java (line 63) mx4j successfully loaded
```

```
$ netstat -ln4p | grep 8081
tcp 0 0 0.0.0.0:8081 0.0.0.0:* LISTEN 2651/java
```

Pod tak zdefiniowanym adresem można znaleźć konsolę dającą dostęp do metryk udostępnianych za pośrednictwem *JMX*:



Rysunek 2: konsola *MX4J*, podgląd stanu detektora awarii w węźle 10.0.0.100

Oczywiście narzędzie `nodetool` także pozwala wyświetlić parametry pracy węzła:

Listing 25: statystyki dla puli wątków z podziałem na zadania

```
$ nodetool -host localhost tpstats
```

Pool Name	Active	Pending	Completed	Blocked	Alltime	blocked
ReadStage	0	0	2	0		0
RequestResponseStage	0	0	2	0		0
MutationStage	0	0	122	0		0
ReadRepairStage	0	0	0	0		0
ReplicateOnWriteStage	0	0	0	0		0
GossipStage	0	0	4236	0		0
CacheCleanupExecutor	0	0	0	0		0
MigrationStage	0	0	0	0		0
MemoryMeter	0	0	13	0		0
FlushWriter	0	0	8	0		0
ValidationExecutor	0	0	0	0		0
InternalResponseStage	0	0	0	0		0
AntiEntropyStage	0	0	0	0		0
MemtablePostFlusher	0	0	42	0		0
MiscStage	0	0	0	0		0
PendingRangeCalculator	0	0	5	0		0
CompactionExecutor	0	0	43	0		0
commitlog_archiver	0	0	0	0		0
HintedHandoff	0	0	2	0		0

Message type	Dropped
RANGE_SLICE	0
READ_REPAIR	0
PAGED_RANGE	0
BINARY	0
READ	0
MUTATION	0
_TRACE	0
REQUEST_RESPONSE	0
COUNTER_MUTATION	0

Listing 26: statystyki dla przestrzeni kluczy `system_auth`

```
$ nodetool -host localhost cfstats

[...]

Keyspace: system_auth
  Read Count: 2
  Read Latency: 43.764 ms.
  Write Count: 3
  Write Latency: 49.754 ms.
  Pending Tasks: 0

[...]

      Table: users
      SSTable count: 1
      Space used (live), bytes: 4689
      Space used (total), bytes: 4689
      SSTable Compression Ratio: 0.7611940298507462
      Number of keys (estimate): 128
      Memtable cell count: 0
      Memtable data size, bytes: 0
      Memtable switch count: 1
      Local read count: 1
      Local read latency: 0.000 ms
      Local write count: 1
      Local write latency: 0.000 ms
      Pending tasks: 0
      Bloom filter false positives: 0
      Bloom filter false ratio: 0.00000
      Bloom filter space used, bytes: 16
      Compacted partition minimum bytes: 61
      Compacted partition maximum bytes: 72
      Compacted partition mean bytes: 72
      Average live cells per slice (last five minutes): 1.0
      Average tombstones per slice (last five minutes): 0.0

-----
```

## 6.2 Przyłączanie i odłączanie węzłów

By dodać do klastra nowy węzeł w celu odciążenia pozostałych, należy wykonać procedurę analogiczną do opisanej w punkcie 5.1. Istnieje możliwość wyspecyfikowania w pliku konfiguracyjnym za ile tokenów (a nawet za jakie dokładnie) nowy węzeł będzie odpowiedzialny, ale nawet bez takiej informacji nowy węzeł będzie w stanie skutecznie odciążyć klaster, ponieważ dzięki gossipingowi ma wgląd w stan wszystkich innych węzłów.

Powodów by odłączyć węzeł może być kilka:

- klaster podlega aktualnie skalowaniu „w dół” i zbędne węzły są usuwane, choć cały czas działają prawidłowo
- węzeł działa, choć niestabilnie — est spora szansa na to, że w najbliższym czasie ulegnie awarii, powinien więc być zastąpiony nowym
- węzeł już uległ awarii i nie odpowiada na zapytania, jednak inne węzły cały czas próbują się z nim kontaktować

W pierwszym przypadku węzeł można odłączyć przy pomocy procesu którego nazwę autorzy pozwolili sobie przetłumaczyć jako „złomowanie” (*ang.* decommissioning). W jego wyniku tokeny złomowanego węzła zostają usunięte z pierścienia, a dane, które się na nim znajdują są transferowane do nowych odpowiedzialnych za nie węzłów.

W praktyce złomowanie węzła sprowadza się do wykonania jednego polecenia za pośrednictwem narzędzia `nodetool`:

Listing 27: stan klastra z perspektywy węzła w nim pozostającego

```
$ nodetool status
Note: Ownership information does not include topology; for complete information,
specify a keyspace
Datacenter: DC1
=====
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
-- Address Load Tokens Owns Host ID Rack
UN 10.0.0.101 115.51 KB 256 32.8% d6537c65-2a7a-4359-a451-fdb4c6685b0c NODE1AND2
UN 10.0.0.100 154.59 KB 256 35.0% ac12364d-1ed1-4d8b-b667-0817f75dc065 NODE0
UN 10.0.0.102 115.33 KB 256 32.2% ecdf71d2-1a4b-4727-bad6-8e293a0acbe3 NODE1AND2
```

Listing 28: odłączenie węzła NODE2/10.0.0.102

```
$ nodetool decommission
```

Listing 29: stan klastra z perspektywy węzła w nim pozostającego po odłączeniu węzła NODE2

```
$ nodetool status
Note: Ownership information does not include topology; for complete information,
specify a keyspace
Datacenter: DC1
=====
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
-- Address Load Tokens Owns Host ID Rack
UN 10.0.0.101 125.02 KB 256 50.7% d6537c65-2a7a-4359-a451-fdb4c6685b0c NODE1AND2
UN 10.0.0.100 154.59 KB 256 49.3% ac12364d-1ed1-4d8b-b667-0817f75dc065 NODE0
```

**UWAGA!** Żadne dane nie są usuwane ze złomowanego węzła, więc jeżeli ma on być ponownie wykorzystany w innym miejscu pierścienia (z innym tokenem), należy wpierw opróżnić jego katalog danych ręcznie.

W drugim przypadku może być potrzebna naprawa klastra po tymczasowej (*ang.* transient) awarii węzła. Standardowe mechanizmy zapewniania ostatecznej spójności są w stanie rozwiązać problemy pojawiające się w takiej sytuacji, z jednym wyjątkiem. Gdy awaria trwa dłużej niż wynosi okres życia *tombstonów*, uszkodzony węzeł mógł nie zarejestrować pewnych operacji usunięcia wierszy.

Wtedy zaleca się wyczyszczenie węzła (usunięcie zawartości jego katalogu danych), usunięcie związanych z nim tokenów z pierścienia i ponowne dodanie do klastra:



Listing 30: usuwanie węzła z nieaktualnymi danymi z klastra

```
$ nodetool ring

Datacenter: DC1
=====
Address Rack Status State Load Owns Token
                                     4960255647790084904
10.0.0.100 NODE0 Up Normal 35.91 KB 81.40% -8739202312123959326
10.0.0.101 NODE1AND2 Down Normal 35.88 KB 44.33% -5308254802912860692
10.0.0.102 NODE1AND2 Up Normal 35.92 KB 74.26% 4960255647790084904

$ nodetool status
Datacenter: DC1
=====
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
-- Address Load Tokens Owns (effective) Host ID Rack
DN 10.0.0.101 35.88 KB 1 44.3% f329244e-c4ac-4fb0-8343-4476d97a311f NODE1AND2
UN 10.0.0.100 35.91 KB 1 81.4% ae935af3-d6b9-433a-a359-a6ef623340fe NODE0
UN 10.0.0.102 35.92 KB 1 74.3% b5c8225f-aff8-4e7d-9d82-221dc113adad NODE1AND2

$ nodetool removenode f329244e-c4ac-4fb0-8343-4476d97a311f

$ nodetool status
Datacenter: DC1
=====
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
-- Address Load Tokens Owns (effective) Host ID Rack
UN 10.0.0.100 35.91 KB 1 100.0% ae935af3-d6b9-433a-a359-a6ef623340fe NODE0
UN 10.0.0.102 35.92 KB 1 100.0% b5c8225f-aff8-4e7d-9d82-221dc113adad NODE1AND2

$ nodetool ring

Datacenter: DC1
=====
Address Rack Status State Load Owns Token
                                     4960255647790084904
10.0.0.100 NODE0 Up Normal 35.91 KB 100.00% -8739202312123959326
10.0.0.102 NODE1AND2 Up Normal 35.92 KB 100.00% 4960255647790084904
```

**UWAGA!** Różnica pomiędzy operacją `decommission` a `removenode` jest taka, że druga z nich zakłada że usuwany węzeł nie działa. W związku z tym nowe węzły odpowiedzialne za zwalniany zakres tokenów będą pobierały dane z pozostałych w systemie replik, podczas gdy w trakcie złomowania źródłem danych będzie złomowany węzeł.

Wreszcie w przypadku całkowitej awarii węzła znaczącą rolę będzie miało opóźnienie odczytów i zapisów związane z „problemami technicznymi” i/lub niespójność danych.

Gdy każdy z węzłów ma przydzielony tylko jeden token (przydzielany ręcznie), można ten problem rozwiązać następująco:

- 1) należy zastępczy węzeł odpowiednio przygotować poprzez nadanie mu adresu IP innego niż poprzednikowi i tokenu mniejszego o 1 od tokenu poprzednika
- 2) podczas przyłączenia do klastra węzeł nie będzie odpowiadał na żądania odczytu aż do momentu zakończenia tego procesu

- 3) po przyłączeniu zastępcy martwy węzeł będzie odpowiadał tylko za jeden token — należy go usunąć z pierścienia poleceniem `nodetool removetoken`
- 4) na każdym z pozostałych węzłów należy wykonać polecenie `nodetool cleanup`, by uaktualnienia tam zapamiętane w efekcie działania mechanizmu *HintedHandoff* niepotrzebnie nie zajmowały miejsca — nigdy nie dotrą one do celu ponieważ węzeł, do którego były adresowane został trwale usunięty z klastra

W przypadku standardowej konfiguracji (wiele tokenów dla pojedynczego węzła, dodatkowo przydzielanych automatycznie), procedura naprawy polega na podłączeniu zastępcy z takim samym adresem IP i uruchomieniu na nim polecenia `nodetool repair`. Jest to o tyle kłopotliwe, że dla odczytów z żądanym niskim poziomem spójności (np. ONE) węzeł przez okres naprawy (który może być bardzo długi) będzie zwracał klientom informację o braku danych.

## Literatura

- [1] A. Lakshman, P. Malik,  
*Cassandra — A Decentralized Structured Storage System*,  
ACM SIGOPS Operating Systems Review,  
Volume 44 Issue 2, April 2010
- [2] Jan Baranowski, Michał Kaik, Maciej Urbański,  
*Cassandra, Highly available, scalable storage system*,  
Politechnika Poznańska 2014
- [3] E. Hewitt,  
*Cassandra, The Definitive Guide*,  
O'Reilly 2010
- [4] *DataStax*,  
<http://www.datastax.com/>,  
dostęp 17 czerwca 2014 r.
- [5] *DebianPackaging, Cassandra Wiki*,  
<https://wiki.apache.org/cassandra/DebianPackaging>.  
dostęp 16 czerwca 2014 r.
- [6] *DistributedDeletes, Cassandra Wiki*,  
<http://wiki.apache.org/cassandra/DistributedDeletes>  
dostęp 17 czerwca 2014 r.
- [7] *Ganglia Monitoring System*, <http://ganglia.sourceforge.net/>  
dostęp 18 czerwca 2014 r.
- [8] *GettingStarted, Cassandra Wiki*,  
<http://wiki.apache.org/cassandra/GettingStarted>,  
dostęp 16 czerwca 2014 r.
- [9] Daniel Stavrovski,  
*Installing Oracle JAVA 7 on Debian Wheezy*,  
<http://d.stavrovski.net/blog/post/installing-oracle-java-7-on-debian-wheezy>,  
dostęp 16 czerwca 2014 r.
- [10] *Java/Sun, Debian Wiki*,  
<https://wiki.debian.org/Java/Sun>,  
dostęp 16 czerwca 2014 r.
- [11] *MySQL Connectors*,  
<http://www.mysql.com/products/connector/>,  
dostęp 16 czerwca 2014 r.
- [12] *NoSQL Apache Cassandra Documentation*,  
<http://planetcassandra.org/documentation/>,  
dostęp 16 czerwca 2014 r.
- [13] *NTP, Debian Wiki*,  
<https://wiki.debian.org/NTP>,  
dostęp 16 czerwca 2014 r.
- [14] *Operations, Cassandra Wiki*,  
<http://wiki.apache.org/cassandra/Operations>,  
dostęp 16 czerwca 2014 r.

- [15] *Why DataStax?*,  
<http://www.datastax.com/why-datastax>  
dostęp 16 czerwca 2014 r.

Tablica 1: podstawowe parametry konfiguracyjne *Cassandra* (w `/etc/cassandra/cassandra.yaml`).

Parametr	Linia	Komentarz
<code>cluster_name</code>	10	nazwa klastra, którego ten węzeł jest członkiem
<code>authenticator</code>	64	Klasa implementująca mechanizm uwierzytelniania. Do wyboru: <i>AllowAllAuthenticator</i> i <i>PasswordAuthenticator</i> . W drugim przypadku nazwy użytkowników i hasła przechowywane są wewnątrz bazy danych w przestrzeni kluczy <code>system_auth</code> .
<code>authorizer</code>	73	Klasa implementująca mechanizm sprawdzania uprawnień przy dostępie do danych (autoryzację). Do wyboru <i>AllowAllAuthorizer</i> i <i>CassandraAuthorizer</i> . W drugim przypadku uprawnienia przechowywane są wewnątrz bazy danych w przestrzeni kluczy <code>system_auth</code> .
<code>seed_provider/parameters/seeds</code>	192	Węzeł <i>Cassandra</i> nie odkrywa innych węzłów automatycznie, stąd dla algorytmu plotkowania wymagana jest lista początkowych "punktów kontaktowych". Oczywiście punkty powinny być tak dobrane by nie dopuścić do partycjonowania klastra.
<code>listen_address</code>	297	Adres IP kanału komunikacyjnego pomiędzy węzłami. Agent algorytmu plotkującego będzie nasłuchiwał na tym adresie. Jeżeli wartość nie zostanie tutaj podana, <i>Cassandra</i> wybierze adres IP localhosta.
<code>start_native_transport</code>	310	Czy uruchomić binarny protokół transportowy wykorzystywany przez drivery DataStax.
<code>native_transport_port</code>	312	Port dla binarnego protokołu transportowego. Jest on podawany w kodzie klienta gdy ten łączy się z określonym węzłem klastra.
<code>start_rpc</code>	324	Czy uruchomić serwer <i>Thrift RPC</i> . Wyłączenie serwera RPC uniemożliwi dostęp do węzła przez np. <code>cqlsh</code> .
<code>rpc_address</code>	335	Adres IP na którym ma nasłuchiwać serwer <i>Thrift</i> .