



HoGent

Faculteit Bedrijf en Organisatie

NoSQL: Apache Cassandra

Lorenz Verschingel

Scriptie voorgedragen tot het bekomen van de graad van
Bachelor in de toegepaste informatica

Promotor:
Sabine De Vreese
Co-promotor:
Jean-Jacques De Clercq

Instelling: HoGent

Academiejaar: 2015-2016

Tweede examenperiode

Faculteit Bedrijf en Organisatie

NoSQL: Apache Cassandra

Lorenz Verschingel

Scriptie voorgedragen tot het bekomen van de graad van
Bachelor in de toegepaste informatica

Promotor:
Sabine De Vreese
Co-promotor:
Jean-Jacques De Clercq

Instelling: HoGent

Academiejaar: 2015-2016

Tweede examenperiode

Samenvatting

Voorwoord

Inhoudsopgave

1	Inleiding	4
1.1	Probleemstelling en Onderzoeksvragen	4
2	Methodologie	5
3	Data opslag in Cassandra	7
4	Opzetten van de Cassandra cluster	8
4.1	Apache Cassandra	8
4.2	DataStax OpsCenter	8
4.3	Toevoegen van een node	12
5	Datamodellering in Cassandra	13
5.1	Primaire sleutel	13
5.2	Partitie kolom	13
5.3	Clustering kolom	14
5.4	De WHERE clause	14
5.4.1	Restricties opgelegd door de partitie kolommen	14
5.4.2	Restricties opgelegd door de clustering kolommen	15
5.5	Doelen bij datamodellering in Cassandra	15
5.6	Datamodel opstellen voor Cassandra	16
6	Data importeren in Cassandra	18
6.1	Importeren via cqlsh	18
6.2	Importeren via sstableloader	19
6.3	Importeren via cassandra-loader	19
6.4	Een vergelijking van de drie methodes	19
7	Gedrag bij uitvallen van een node	20
8	Nood aan backups in Cassandra	21

9 Conclusie

22

Hoofdstuk 1

Inleiding

De inleiding moet de lezer alle nodige informatie verschaffen om het onderwerp te begrijpen zonder nog externe werken te moeten raadplegen (?). Dit is een doorlopende tekst die gebaseerd is op al wat je over het onderwerp gelezen hebt (literatuuronderzoek).

Je verwijst bij elke bewering die je doet, vakterm die je introduceert, enz. naar je bronnen. In \LaTeX kan dat met het commando `\cite{}` of `\citep{}`. Als argument van het commando geef je de “sleutel” van een “record” in een bibliografische databank in het Bib \TeX -formaat (een tekstbestand). Als je expliciet naar de auteur verwijst in de zin, gebruik je `\cite{}`. Soms wil je de auteur niet expliciet vernoemen, dan gebruik je `\citep{}`. Hieronder een voorbeeld van elk.

? schreef een van de standaardwerken over sorteer- en zoekalgoritmen. Experts zijn het erover eens dat cloud computing een interessante opportuniteit vormen, zowel voor gebruikers als voor dienstverleners op vlak van informatietechnologie (?).

1.1 Probleemstelling en Onderzoeksvragen

Hoofdstuk 2

Methodologie

Om een antwoord te vinden op alle onderzoeksvragen werd deze bachelorproef opgesplitst in twee luiken. Het eerste luik omvat het eerder theoretische deel, waar een literatuurstudie aan te pas kwam. Het tweede luik bevat het praktische deel die nodig was om op een aantal vragen antwoord te krijgen.

In het theoretische luik komt zoals reeds eerder vermeld de literatuurstudie naar voren. Hierin wordt nagegaan hoe data wordt opgeslagen binnen Cassandra, wat er juist bedoeld wordt met het meervoudig opslaan van data, hoe belangrijk back-ups zijn binnen dit systeem, voor welke problemen is Cassandra een oplossing. . .

Om dit alles te kunnen nagaan werd het praktische gedeelte opgezet. Eerst moest er een Cassandra cluster opgezet worden. Dit gebeurde aan de hand van vagrant virtuele machines. Er werd geopteerd voor vagrant omdat dit een snelle manier is om verschillende identieke virtuele machines op te zetten. Ook kon aan de hand van één enkel script de volledige omgeving gecontroleerd worden. Voor de installatie van Cassandra werd eerst gekozen om met de apache versie te werken. Het idee hiervan was om met de meest recente versie te werken. Om praktische reden werd later verkozen om via het OpsCenter Community Edition van Datastax te werken. Deze tool maakte het mogelijk om via een webinterface de databank Cassandra te beheren en te monitoren. Door gebruik te maken van deze opzet konden snel nodes toegevoegd of verwijderd worden binnen de cluster, via deze opzet kon de schaalbaarheid makkelijk getest worden.

Toen deze cluster opgezet was, werd de data die voorzien werd door de Universiteit van Gent ingeladen in deze virtuele cluster. Voor deze data ingeladen kon worden moest eerst stilgestaan worden bij het datamodel van deze data. Deze data werd dus ook gebruikt om uit te leggen hoe je het best een datamodel opstelt binnen Cassandra. Hier werd eveneens kort stil gestaan bij het verschil tussen datamodellering binnen een

relationele databank en Cassandra.

In een laatste deel moest ook nog de betrouwbaarheid van Cassandra getest worden. Hier werd doelbewust een van de virtuele machines, een van de nodes van de databank, uitgeschakeld om te zien hoe Cassandra hierop reageert. Doordat dit in een virtuele omgeving gebeurde is er geen risico op verlies van kritieke data.

Hoofdstuk 3

Data opslag in Cassandra

Hoofdstuk 4

Opzetten van de Cassandra cluster

4.1 Apache Cassandra

Om te beginnen aan het opzetten van de van de clusters werd geopteerd om gebruik te maken van virtuele machines, die geconfigureerd werden met Vagrant.

In een eerste poging om een werkende Cassandra cluster te bekomen werd er op elke Vagrant machine Cassandra 3.3, op moment van schrijven de meest recente versie, geïnstalleerd. Nadat dit was gebeurd dienden nog enkele stappen te voltooid worden om een werkende cluster te bekomen (DataStax, 2016). De configuratie op deze manier gaf echter veel problemen, Cassandra werd telkens na enkele bewerkingen onbruikbaar met de foutboodschap "could not access pidfile for Cassandra". Een eerste oplossing voor dit probleem was om te zorgen dat de user cassandra toegang had tot de pidfile, want namelijk niet het geval was doordat de installatie van Cassandra werd uitgevoerd door de vagrant setup. Maar ook dit leverde weinig resultaat op. Wel dient opgemerkt te worden dat deze installatie van Cassandra zelf geen problemen met zich meebracht. Voor het aanpassen van de configuratie file van Cassandra werkte deze perfect op iedere node.

4.2 DataStax OpsCenter

Uiteindelijk werd er geopteerd om gebruik te maken van OpsCenter omdat dit een gemakkelijke manier is om snel een Cassandra cluster te bekomen en omdat dit ook goede mogelijkheden tot monitoren van de database voorziet. Van het OpsCenter

werd er voor de community edition 5.2.4 gekozen. Hiermee komt Cassandra 2.1.11 geïnstalleerd (Cantoni, 2016).

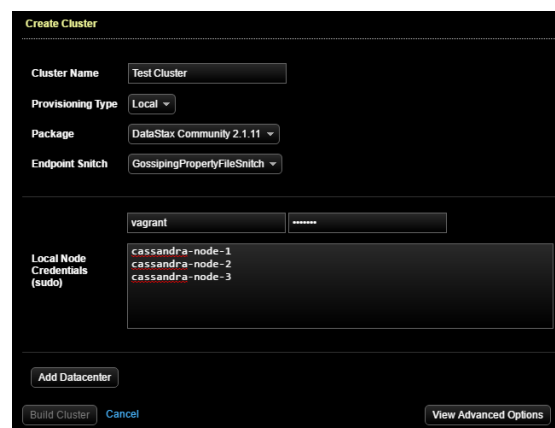
De setup bestaat uit 1 master node waarop het OpsCenter runt en dan 3 slave nodes waar de uiteindelijke Cassandra database op komt te runnen. Na de NAT router van Oracle Virtual Box werd een privaat netwerk opgezet zodanig deze machines met elkaar konden communiceren. Hiervoor moest elke machine een ip-adres krijgen binnen het netwerk en ook de `/etc/hosts` aangepast worden. In bijlage A kunnen de scripts hiervoor teruggevonden worden.

Eenmaal de virtuele machines correct geconfigureerd waren, werd er overgegaan tot de eigenlijke installatie van Cassandra. Zoals eerder vermeld werd hiervoor gebruik gemaakt van het OpsCenter. Hiervoor werd op de master node naar de localhost:8888 gesurft om de installatie te starten. Op de pagina dit te voorschijn komt werd voor de optie 'brand new cluster gekozen'.

In het volgende venster wordt er om verschillende zaken gevraagd. Tabel 4.1 en figuur 4.1 geven weer hoe dit venster ingevuld werd.

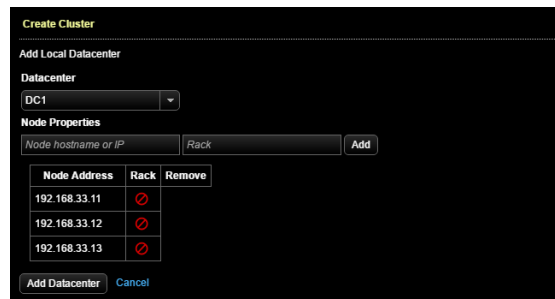
Property Name	Waarde
Cluster Name	BP Cluster
Type	local
Package	datastax community 2.1.11
Endpoint Snitch	GossipingPropertyFileSnitch
Username en password	vagrant/vagrant
Local Node Credentials	cassandra-node-1, cassandra-node-2, cassandra-node-3

Tabel 4.1: Configuratie van de Cassandra Cluster



Figuur 4.1: Cassandra: Instellingen deel 1

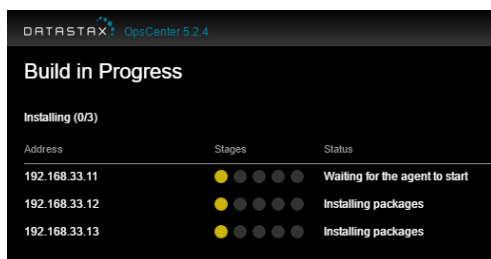
Hier dient een datacenter toegevoegd te worden. Hierbij wordt de naam van het datacenter vrij gekozen en zijn de node properties het ip-adres van de slave nodes (Figuur: 4.2).



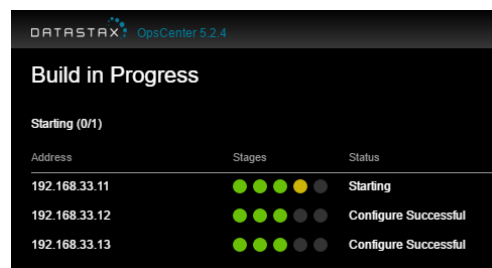
Figuur 4.2: Cassandra: Instellingen deel 2

Eenmaal de datacenters zijn toegevoegd kan men verdergaan. Bij het drukken op de knop 'build cluster' word nog gevraagd om de fingerprints van de nodes te accepteren.

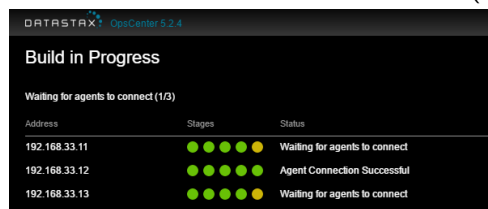
Hierna begint OpsCenter met de installatie van de Cassandra cluster (Figuur: 4.3). Deze installatie neemt enkele ogenblikken in beslag. Als hier fouten voorkomen ligt dit veelal aan het feit dat er onvoldoende werkgeheugen aanwezig is op de slave nodes. In de setup die hier gebruikt werd het minimum aanvaarde geheugen geven aan de slave nodes, nl 2GB. Samen met Cassandra wordt ook de DataStax agent meegeïnstalleerd zodat het OpsCenter kan communiceren met iedere node.



(a) Deel 1



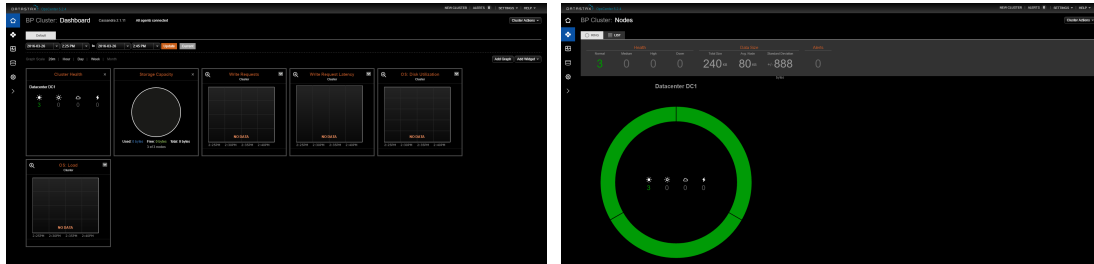
(b) Deel 2



(c) Deel 3

Figuur 4.3: Installatie van Cassandra door OpsCenter

Na de installatie komt men terecht in het Dashboard van OpsCenter (Figuur 4.4a). Hierin worden een aantal zaken weergegeven zoals de gezondheid van de cluster, het aantal write requests, de write request latency, ... Hier kunnen nog meer grafieken aan toegevoegd worden via de knop "add graph". In het tabblad nodes kan gezondheid van de cluster bekeken worden evenals hoe de data verdeeld zit over de cluster. Deze informatie kan in ringvorm zoals op figuur 4.4b weergegeven worden of in een lijst.



(a) Tabblad dashboard

(b) Tabblad nodes: ring

Figuur 4.4: Rondleiding in OpsCenter

Cassandra voorziet zelf ook een tool die deze monitoring voorziet, nodetool. Om te bekijken of de installatie gelukt is kan men via ssh inloggen op een van de nodes van de cluster en hier nodetool laten lopen (Figuur 4.5). Als men met nodetool de status opvraagt dan krijgt men eveneens alle nodes in de cluster te zien, samen het de hoeveelheid data die ze bevatten en hoeveel dit percentueel is van de data die aanwezig is in de cluster.

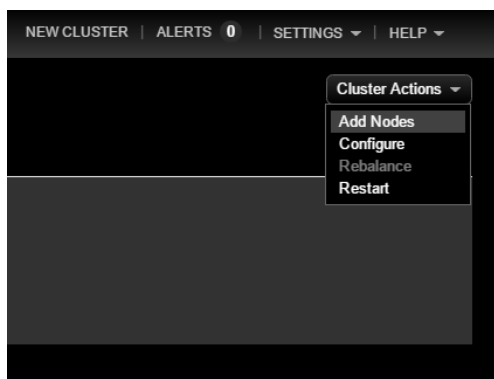
```
vagrant@cassandra-node-1:~$ nodetool status
Datacenter: DC1
Status=Up/Down
// State=Normal/Leaving/Joining/Moving
-- Address      Load      Tokens     Owns    Host ID
--  --
UN 192.168.33.11  80.5 KB   256        ?       74a74814-d29b-4091-b621-89599cb4b1f9
UN 192.168.33.12  80.97 KB  256        ?       0167d105-d0ca-4994-9550-0fc089c6a4db
UN 192.168.33.13  80.38 KB  256        ?       279f3eae-832f-4a1e-9d70-d68512ea39ff

Note: Non-system keyspace don't have the same replication settings, effective ownership information is meaningless
vagrant@cassandra-node-1:~$
```

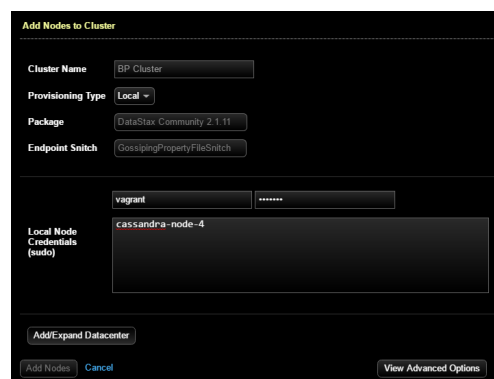
Figuur 4.5: Nodetool

4.3 Toevoegen van een node

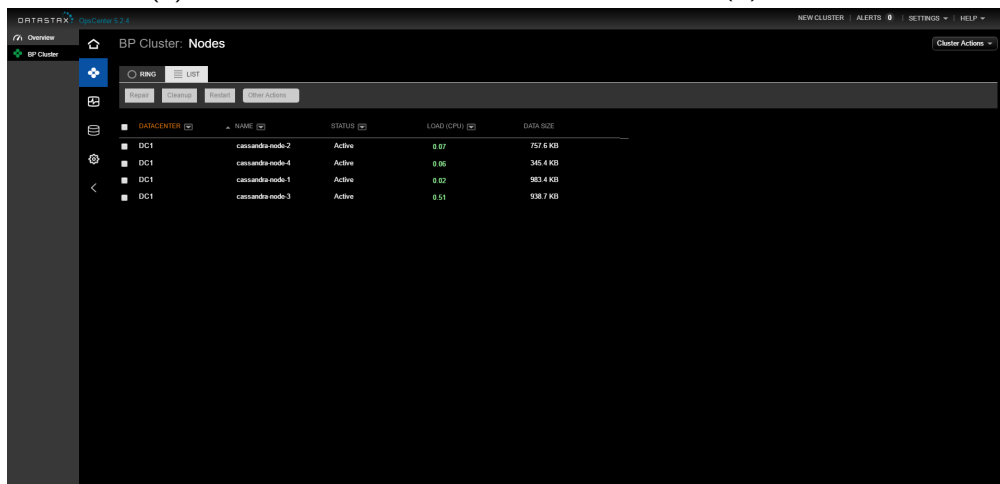
Het proces van het toevoegen van een node is zeer gelijkaardig aan het opzetten van de cluster. Hierbij dient men in het OpsCenter bij het menu "cluster actions" voor de optie "add node" kiezen (Figuur 4.6a). Hier wordt een vierde node toegevoegd aan de cluster. Net zoals bij de installatie van de cluster wordt opnieuw het scherm getoond waarin de aanmeld gegevens voor de gegeven node gevraagd worden (Figuur 4.6b). Via de knop "Add/Expand Datacenter" kan men op dezelfde manier nodes toevoegen aan het bestaande datacenter die tijdens de installatie werd aangemaakt. Bij het bevestigen wordt opnieuw gevraagd om de fingerprint van de node te accepteren. Hierna begint de installatie van de software op de node, de agent en cassandra worden geïnstalleerd en geconfigureerd. Na dit alles kan men in het tabblad nodes, deze maal als een lijst, zien dat de nieuwe node is toegevoegd aan de cluster (Figuur: 4.6c).



(a) Add node



(b) Deel 2



(c) Tabblad node met 4 nodes

Figuur 4.6: Toevoegen van een node via OpsCenter

Hoofdstuk 5

Datamodellering in Cassandra

5.1 Primaire sleutel

Binnen de primaire sleutel in Cassandra worden de partitie (5.2) en clustering (5.2) kolommen vastgelegd voor een tabel. In tegenstelling tot relationele databases dient de primaire sleutel hier niet als unieke sleutel voor de rij maar om snel te weten te komen waar de data zich bevindt (Kan, 2014). Als een tabel slechts één kolom heeft als primaire sleutel dan is deze kolom meteen ook de partitie kolom en zijn er geen clustering kolommen.

Door het feit dat de primaire sleutel in Cassandra niet uniek hoeft te zijn gedragen het INSERT commando en het UPDATE commando zich op een identieke wijze. Een nadeel hierbij is dat er geen waarschuwing wordt gegeven als er rij overschreven wordt. Omdat dit veelal ongewenst gedrag is binnen applicaties komt men al snel terecht bij een samengestelde primaire sleutel, een primaire sleutel uit verschillende kolommen.

5.2 Partitie kolom

Het eerste deel van de primaire sleutel bestaat uit de partitie kolommen. Het doel van de partitie kolommen is om de data gebalanceerd over alle nodes te spreiden. Op deze manier kan de fysieke locatie van de data ook snel achterhaalt worden (Kan, 2014).

Bij het kiezen van de partitie kolommen dient er met een aantal zaken rekening gehouden te worden om de data evenwichtig over alle nodes te spreiden, maar er is slechts één fysieke restrictie. Iedere rij kan slechts 2 miljard kolommen bevatten (McFadin, 2013). In de meeste gevallen is dit ruim voldoende. Tijdsreeksen vormen hier een

uitzondering op. Deze reeksen kunnen al gauw miljarden entries bevatten. Hier is het dus zeer belangrijk om goede partitie kolommen te kiezen of men komt hier al snel in de problemen.

5.3 Clustering kolom

Het laatste deel van de primaire sleutel bestaat uit de clustering kolommen. Deze bepalen de volgorde van de data op de fysieke media (Strickland, 2014). De clustering kolommen hebben echter geen invloed over op welke node de data wordt opgeslagen. Toch zullen ze een belangrijke invloed hebben op welke query's er uitgevoerd kunnen worden.

5.4 De WHERE clause

Zoals Lerer (2015) aanhaalt is een van de grootste verschillen tussen CQL en SQL de WHERE clause. In SQL kent de WHERE clause geen restricties. Bij CQL is dit anders. Hier worden restricties opgelegd door de partitie en clustering kolommen. Eveneens kan er enkel op de partitie en clustering kolommen gefilterd worden.

5.4.1 Restricties opgelegd door de partitie kolommen

Een eerste restrictie die wordt opgelegd door de partitie kolommen is dat ofwel alle partitie kolommen worden opgenomen in de WHERE clause ofwel geen enkel. Dit is nodig omdat Cassandra anders de hash niet kan berekenen. Deze hash is echter nodig om te weten op welke node de data zich bevind.

Een volgende restrictie die wordt opgelegd door de partitie kolommen is het feit dat enkel de '=' en IN operator rechtstreeks gebruikt kunnen worden. Tot Cassandra 2.2 kon zelfs de IN operator enkel op de laatste gedefinieerde partitie kolom.

De laatste restrictie die wordt opgelegd door de partitie kolommen heeft te maken met de >, >=, < en <= operators. Deze zijn enkel rechtstreeks toepasbaar als de partitioner ingesteld op ByteOrderedPartitioner. Indien dit niet het geval is moet men een omweg maken via de token functie. Op het eerste zicht lijkt het gebruik van de token functie minder efficiënt voor het selecteren van data, maar dit weegt niet op tegen de nadelen die ByteOrderedPartitioner heeft op gebied van de distributie van de data, zoals reeds eerder werd vermeld.

5.4.2 Restricties opgelegd door de clustering kolommen

Voor de restricties op de clustering kolommen uitgelegd kunnen worden, moet eerst uitgelegd worden hoe de data in Cassandra binnen een partitie opgeslagen wordt. Dit zal nu aan de hand van een voorbeeld uitgelegd worden (Lerer, 2015). Neem de onderstaande tabel:

```
CREATE TABLE NumberOfTwitterMessages (  
    userid bigint, date text, hour int, minute int,  
    nrOfTweets int,  
    PRIMARY KEY ((userid, date), hour, minute)  
);
```

Hier wordt de data nu als volgt opgeslagen binnen de partitie:

```
{hour: 20  
  {minute: 4 {nrOfTweets: 6}}  
  {minute: 7 {nrOfTweets: 1}}  
  {minute: 21 {nrOfTweets: 16}}  
  ...  
}
```

De eerste restrictie wordt hier opgelegd door de manier waarop de data opgeslagen zit. Als men een voorwaarde wil vastleggen voor een clustering kolom, dan dienen de clustering kolommen die voor deze komen in de primaire sleutel ook vastgelegd te worden. Dit is nodig omdat Cassandra de data anders niet efficiënt kan terugvinden.

Tot Cassandra 2.2 was de IN operator enkel toegelaten op de laatste clustering kolom. Sinds Cassandra 2.2 is deze restrictie vervallen en kan men nu zelf multi-kolom IN restricties opleggen.

De >, >=, < en <= operators ook enkel toegestaan op de laatste clustering kolom waar een restrictie is aan opgelegd. Toch is hier een oplossing voor aangezien men deze operators kan toepassen over meerdere kolommen. Bij deze multi-column slices is het echter wel belangrijk dat de restrictie van de WHERE clause met dezelfde kolom start.

5.5 Doelen bij datamodellering in Cassandra

Zoals in bovenstaande sectie duidelijk werd, dient er bij Cassandra met heel wat rekening gehouden te worden als men een datamodel creëert. Hobbs (2015) definieerde

wat de doelen zijn bij het opstellen van een datamodel in Cassandra. Zoals door Hobbs aangegeven wordt, lijkt de querytaal van Cassandra CQL sterk op SQL, maar kan het gebruik ervan zeer verschillend zijn.

Een eerste doel bij het opstellen van een datamodel in Cassandra is om de data evenwichtig over alle nodes te verspreiden. Dit kan bekomen worden door de partitie kolommen goed te kiezen. Zoals eerder vermeld is, worden de partitie kolommen bepaald door het eerste deel van de primaire sleutel.

Een tweede doel is om zo weinig mogelijk partitie reads te moeten doen. Doordat iedere partitie op een andere node kan staan is dit belangrijk. Als de partities effectief op verschillende nodes staan moet de afzonderlijke commando's naar elke node apart verstuurd worden en dit zorgt voor overhead. Het is zelfs zo dat als de data op dezelfde node staat, de partitie reads nog steeds inefficiënt zijn. Dit komt door de manier waarop Cassandra de rijen opslaat.

Zaken die bij relationele databanken belangrijk zijn zoals het aantal writes en data duplicatie minimaliseren zijn binnen Cassandra geen doelen. In Cassandra zijn write goedkoop omdat Cassandra hiervoor geoptimaliseerd is. Denormalisatie en duplicatie van data zijn ook zeer normaal binnen Cassandra. Dit komt door de architectuur van Cassandra. Hierbij gaat men ervan uit dat schijfruimte goedkoop is in vergelijking met andere resources zoals CPU, geheugen, netwerk ... Ook geeft Cassandra geen JOIN waardoor het ook hoogst inefficiënt en onpraktisch zou zijn om geen duplicate data te hebben.

5.6 Datamodel opstellen voor Cassandra

Hobbs (2015) definieerde niet enkel de doelen van een datamodel in Cassandra, maar haalt ook aan hoe deze bekomen kunnen worden. Voor men begint aan het opstellen van een datamodel moet al nagedacht worden over de query's die ondersteunt moeten worden. Dit staat in schril contrast met relationele databanken waar het datamodel wordt bepaald door de objecten en hun relaties.

Door na te denken over de te ondersteunen query's kan het aantal partitie read al drastisch verminderd worden. Ook dient men rekening te houden met de restricties die de partitie en clustering kolommen opleggen aan de where clause.

Voor iedere query zou er slechts één partitie read uitgevoerd mogen worden. Hiervoor is het belangrijk dat de tabellen geoptimaliseerd zijn voor de reads die uitgevoerd gaan worden.

Een goed voorbeeld wordt ook gegeven door Hobbs (2015) waar alle zaken meteen

duidelijk worden. In dit voorbeeld is het de bedoeling om users op het halen volgens hun email of username. De oplossing voor Cassandra zijn de volgende twee tabellen:

```
CREATE TABLE users_by_username (
  username text PRIMARY KEY,
  email text ,
  age int
)
```

```
CREATE TABLE users_by_email (
  email text PRIMARY KEY,
  username text ,
  age int
)
```

Bij dit datamodel is de krijg iedere user zijn eigen partitie. Cassandra kan zo de data evenwichtig verdelen over de nodes. Ook dient om een user op te zoeken via een email of username slechts één partitie gelezen te worden.

Stel dat men nu zou proberen om redundante data te verminderen, wat in Cassandra geen doel mag zijn, met de volgende tabellen:

```
CREATE TABLE users (
  id uuid PRIMARY KEY,
  username text ,
  email text ,
  age int
)
```

```
CREATE TABLE users_by_username (
  username text PRIMARY KEY,
  id uuid
)
```

```
CREATE TABLE users_by_email (
  email text PRIMARY KEY,
  id uuid
)
```

De data zal met deze tabellen nog steeds evenwichtig gedistribueerd zijn over de nodes, maar nu moet men meer dan één partitie read doen. Dus door een niet-doel proberen te bereiken is hier een belangrijk doel verloren gegaan.

Hoofdstuk 6

Data importeren in Cassandra

6.1 Importeren via cqlsh

Om de data via cqlsh te kunnen importeren dient eerst een keyspace aangemaakt te worden. Bij het aanmaken van deze keyspace dient de replicatie strategie en de replicatie factor meegegeven te worden. Na het aanmaken dient de tabel waarin de data zal worden geïmporteerd worden aangemaakt te worden.

Nu kunnen we via het "COPY"commando, dat binnen cql voorzien is om data importeren in Cassandra (Cannon, 2012). Een aantal zaken dient hierbij opgemerkte te worden.

1. De volgorde van de kolommen kan gespecificeerd worden aangezien Cassandra de kolommen automatisch alfabetisch sorteert en dit niet noodzakelijk het geval is bij het csv bestand.
2. De scheidingstekens van de velden in het csv bestand kan gespecificeerd worden evenals de encapsulering van de velden.
3. Er kan specifiek meegegeven worden wat Cassandra moet aanvangen met de null waarde.

Dit is een zeer eenvoudige manier om data te importeren in Cassandra. Toch wordt dit deze methode niet aangeraden om te gebruiken bij het importeren van grote hoeveelheden data. Bij ca. 1 miljoen rijen, afhankelijk van het aantal kolommen, kan het zijn dat deze methode vast loopt. Dit kan men op een aantal manieren oplossen. De drie meest voorkomende zijn:

1. Het opsplitsen van één groot csv bestand in verschillende kleinere bestanden.

2. Het gebruik van de sstableloader die Cassandra voorziet.
3. Het gebruik van cassandra-loader

6.2 Importeren via sstableloader

Het importeren van data via sstableloader is eveneens een eenvoudig proces. Hier zijn reeds verschillende implementaties van, zoals cassandra bulkloader.

Enkele belangrijke nadelen van deze manier zijn:

- Men moet een aangepaste applicatie schrijven om dit te kunnen gebruiken.
- Om sstableloader te kunnen gebruiken dienen alle nodes van de cluster online te zijn.
- De SSTable dient aangemaakt te zijn vooraleer men deze methode kan gebruiken.

6.3 Importeren via cassandra-loader

Dit is een Java programma van Brain Hess. Dit programma maakt gebruik van de CQL driver die voorzien is door DataStax. Het ganse principe van dit programma is om asynchroon cql inserts te doen.

6.4 Een vergelijking van de drie methodes

Hoofdstuk 7

Gedrag bij uitvallen van een node

Hoofdstuk 8

Nood aan backups in Cassandra

Hoofdstuk 9

Conclusie

Bibliografie

- Cannon, P. (2012). Simple data importing and exporting with cassandra. <http://www.datastax.com/dev/blog/simple-data-importing-and-exporting-with-cassandra>. Geraadpleegd op: 2016-03-04.
- Cantoni, B. (2016). MultiNode Template. <https://github.com/bcantoni/vagrant-cassandra/tree/master/2.MultiNode>. Geraadpleegd op 2016-03-25.
- DataStax (2016). Initializing a multiple node cluster (single data center). <https://docs.datastax.com/en/cassandra/2.1/cassandra/initialize/initializeSingleDS.html>. Geraadpleegd op 2016-02-26.
- Hobbs, T. (2015). Basic Rules of Cassandra Data Modeling. <http://www.datastax.com/dev/blog/basic-rules-of-cassandra-data-modeling>. Geraadpleegd op 2016-03-20.
- Kan, C. (2014). *Cassandra Data Modeling and Analysis*. Packt Publishing Ltd.
- Lerer, B. (2015). A deep look at the cql where clause. <http://www.datastax.com/dev/blog/a-deep-look-to-the-cql-where-clause>. Geraadpleegd op 2016-04-25.
- McFadin, P. (2013). Cassandra 2.0 and timeseries. <http://www.slideshare.net/patrickmcfadin/cassandra-20-and-timeseries>. Geraadpleegd op 2016-04-25.
- Strickland, R. (2014). *Cassandra High Availability*. Packt Publishing Ltd.

Lijst van figuren

4.1	Cassandra: Instellingen deel 1	9
4.2	Cassandra: Instellingen deel 2	10
4.3	Installatie van Cassandra door OpsCenter	10
4.4	Rondleiding in OpsCenter	11
4.5	Nodetool	11
4.6	Toevoegen van een node via OpsCenter	12

Lijst van tabellen

4.1 Configuratie van de Cassandra Cluster	9
---	---