KU LEUVEN

FACULTEIT INGENIEURSWETENSCHAPPEN

Automatisch uitrol van database systemen en vergelijking van beschikbaarheid

Thomas Uyttendaele

Thesis voorgedragen tot het behalen van de graad van Master of Science in de ingenieurswetenschappen: computerwetenschappen, hoofdspecialisatie Gedistribueerde systemen

Promotor:
Prof. dr. ir. Wouter Joosen

Assessor:
Ir.

Begeleider:
Ir. B. Vanbrabant

Academiejaar 2013 – 2014

Voorwoord

Sed feugiat. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Ut pellentesque augue sed urna. Vestibulum diam eros, fringilla et, consectetuer eu, nonummy id, sapien. Nullam at lectus. In sagittis ultrices mauris. Curabitur malesuada erat sit amet massa. Fusce blandit. Aliquam erat volutpat. Aliquam euismod. Aenean vel lectus. Nunc imperdiet justo nec dolor.

Voorwoord schrijven

Etiam euismod. Fusce facilisis lacinia dui. Suspendisse potenti. In mi erat, cursus id, nonummy sed, ullamcorper eget, sapien. Praesent pretium, magna in eleifend egestas, pede pede pretium lorem, quis consectetuer tortor sapien facilisis magna. Mauris quis magna varius nulla scelerisque imperdiet. Aliquam non quam. Aliquam porttitor quam a lacus. Praesent vel arcu ut tortor cursus volutpat. In vitae pede quis diam bibendum placerat. Fusce elementum convallis neque. Sed dolor orci, scelerisque ac, dapibus nec, ultricies ut, mi. Duis nec dui quis leo sagittis commodo.

Aliquam lectus. Vivamus leo. Quisque ornare tellus ullamcorper nulla. Mauris porttitor pharetra tortor. Sed fringilla justo sed mauris. Mauris tellus. Sed non leo. Nullam elementum, magna in cursus sodales, augue est scelerisque sapien, venenatis congue nulla arcu et pede. Ut suscipit enim vel sapien. Donec congue. Maecenas urna mi, suscipit in, placerat ut, vestibulum ut, massa. Fusce ultrices nulla et nisl.

Thomas Uyttendaele

Inhoudsopgave

Todo list

Samenvatting

Abstract schrijven

In dit `abstract` environment wordt een al dan niet uitgebreide samenvatting van het werk gegeven. De bedoeling is wel dat dit tot 1 bladzijde beperkt blijft. Sed feugiat. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Ut pellentesque augue sed urna. Vestibulum diam eros, fringilla et, consectetuer eu, nonummy id, sapien. Nullam at lectus. In sagittis ultrices mauris. Curabitur malesuada erat sit amet massa. Fusce blandit. Aliquam erat volutpat. Aliquam euismod. Aenean vel lectus. Nunc imperdiet justo nec dolor.

Etiam euismod. Fusce facilisis lacinia dui. Suspendisse potenti. In mi erat, cursus id, nonummy sed, ullamcorper eget, sapien. Praesent pretium, magna in eleifend egestas, pede pede pretium lorem, quis consectetuer tortor sapien facilisis magna. Mauris quis magna varius nulla scelerisque imperdiet. Aliquam non quam. Aliquam porttitor quam a lacus. Praesent vel arcu ut tortor cursus volutpat. In vitae pede quis diam bibendum placerat. Fusce elementum convallis neque. Sed dolor orci, scelerisque ac, dapibus nec, ultricies ut, mi. Duis nec dui quis leo sagittis commodo.

Aliquam lectus. Vivamus leo. Quisque ornare tellus ullamcorper nulla. Mauris porttitor pharetra tortor. Sed fringilla justo sed mauris. Mauris tellus. Sed non leo. Nullam elementum, magna in cursus sodales, augue est scelerisque sapien, venenatis congue nulla arcu et pede. Ut suscipit enim vel sapien. Donec congue. Maecenas urna mi, suscipit in, placerat ut, vestibulum ut, massa. Fusce ultrices nulla et nisl.

Lijst van figuren en tabellen

Lijst van figuren

Lijst van tabellen

Lijst van afkortingen en symbolen

Afkortingen

IMP	Infrastructure Management Platform
DBMS	Databasemanagementsysteem
RDBMS	Relationeel Databasemanagementsysteem
Range Query	Het opvragen van een set van records met behulp van een enkele query

Symbolen

42	aaa

Hoofdstuk 1

Inleiding

Hoofdstuk 2

Overzicht van de technologie

2.1 Geschiedenis van de databasemanagementsystemen

Doorheen de geschiedenis, heeft de mens verschillende manieren gebruikt om data op te slaan en nadien terug te vinden. Doorheen de geschiedenis zijn er verschillende stappen van data management geweest, tot voor het ontstaan van de computer ging dit met pen en papier of met ponskaarten[7]. Met de opkomst van de computer, werden nieuwe methodes gebruikt die doorheen de tijd zijn mee geëvolueerd met de vooruitgang in de technologie en de veranderingen in het gebruik van de data. De hiervoor ontwikkelde software wordt gecategoriseerd onder het **databasemanagementsysteem**(DBMS). De ontwikkeling en opkomst van de DBMS kan in verschillende fasen opgedeeld worden.

De eerste DBMS zijn er gekomen met de introductie van de mainframes zoals UNIVAC1 en de ontwikkeling van specifieke programmeertalen voor het werken met deze data, onder andere conferenties zoals CODASYL hebben de ontwikkeling van COBOL en andere standaarden mee ontwikkeld[7].

Mss ook het toevoegen van IBM hier + (IDS)?

De volgende grote verandering in DBMS is er gekomen door het artikel van E. Codd over het relationele model in 1969 [4]. De sleutel concepten van het relationele model is dat de data georganiseerd is in relaties (tabellen) die gekoppeld zijn door middel van keys (constraints) waarbij redundante data wordt vermeden. Voorbeelden van populaire RDBMS (relationele DBMS) die het model implementeren zijn Oracle, MySQL en PostgreSQL. Meer informatie komt aanbod in de volgende sectie.

De laatste nieuwe generatie zijn NoSQL databases die sinds 2000 zijn begonnen, NoSQL staat voor '*Not only SQL*'. Deze systemen zijn er gekomen als reactie op het relationele model voor een meer flexibele database, lagere complexiteit, hogere doorvoer van data, horizontale schaalbaarheid en het draaien op commodity hardware. Verschillende voorbeelden van NoSQL systemen zijn Google BigTable, Amazon Dy-

namo, HBase, MongoDB, ... [17] Meer informatie en een vergelijking met relationele databases komt aanbod in de volgende sectie.

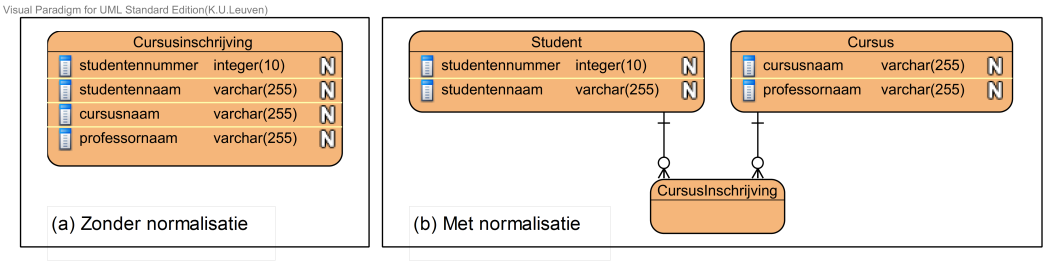
2.2 Relationele en NoSQL databases

Yeaah

2.2.1 Relationele database

Een RDBMS is een DBMS gebaseerd op relationele model voor het structuren van de database.

Het relationele model is gebaseerd op theoretische wiskundige principes zoals de set-theorie en eerste-orde predicaten logica. Het model organiseert de data in tabellen en relaties tussen de relaties. De tabel heeft kolommen die verschillende velden voorstellen en elke rij een collectie van gerelateerde datawaardes is. De relaties tussen de verschillende tabellen toont mogelijke connecties. Een belangrijke eigenschap is dat de tabellen en relaties genormaliseerd worden, hiermee wordt redundante informatie verwijderd. Dit zorgt voor een hogere data integriteit en een vermindering in data anomalieën die kunnen optreden bij een update.[6]
Dit kan geïllustreerd worden met het korte voorbeeld in figuur 2.1: de professor voor een vak zal bij elke student hetzelfde zijn, het veranderen van een professor voor een vak zou in het eerste geval een update van alle ingeschreven studenten inhouden, in het tweede geval is dit maar één enkel record, hetzelfde geldt voor de student.
Interactie met de RDBMS gebeurt op basis van SQL (Structured Query Language), een taal gebaseerd op de relationele logica geeft uitgebreide query mogelijkheden aan de gebruiker van de software.



Figuur 2.1: Relationeel datamodel (a) zonder en (b) met normalisatie

Een belangrijk concept in een relationele database is ACID tussen verschillende transacties:

Atomair (Atomicity) Een database transactie moet oftewel volledig uitgevoerd worden oftewel heeft geen enkele databasebewerking plaatsgevonden.

Consistent (Consistency) Een transactie behoudt de consistentie als de volledige uitvoering van de transactie de database van één consistente staat naar een andere brengt. Een consistente staat is een staat die ervoor zorgt dat waardes van een instantie consistent zijn met de andere waarden in dezelfde staat. Een voorbeeld is het overschrijven van €50 van persoon A naar B, op het einde moet de totale som nog steeds gelijk zijn, A €50 minder en B €50 meer.

Geïsoleerd (Isolation) Een transactie moet uitgevoerd worden alsof ze geïsoleerd is van andere transacties, die eventueel gelijktijdig uitgevoerd worden.

Duurzaam (Durability) Een voltooide transactie kan later niet ongedaan gemaakt worden. Deze verschillende concepten bieden de gebruiker van het RDBMS garanties die de programmeur van de database kan gebruiken voor zijn systeem. Daartegen over staat wel dat dit de complexiteit van de RDBMS groeit, ook indien dit voor bepaalde toepassingen misschien niet nodig is.

2.2.2 NoSQL database

NoSQL DBMS zijn ontstaan als reactie op 'one size fits all'-gedachte die RDBMS volgt. Dit is ook zichtbaar in de verschillende NoSQL systemen, ze bestaan in verschillende variëteiten, elk voor met hun eigen eigenschappen en toepassingsgebied, toch is er een rode draad te vinden tussen de verschillende systemen vergeleken met RDBMS:

- **Lagere complexiteit**: NoSQL systemen bieden minder opties en features aan dan de RDBMS omdat veel applicaties die nu eenmaal niet nodig hebben. Bijvoorbeeld in een sociale netwerk moet een post niet onmiddellijk beschikbaar zijn voor al de vrienden van een persoon, maar dit mag even duren.

- **Hogere doorvoer**: Talrijke NoSQL systemen bieden een hogere doorvoer van data aan, meestal gecombineerd met een lagere complexiteit.

- **Horizontale schaalbaarheid en werkend op commodity hardware**: Waar grote RDBMS draaien op dure high-end systemen, was het bedoeling van NoSQL databases om te werken met eenvoudige machines (commodity hardware).
  Horizontale schaalbaarheid staat voor het toevoegen extra machines aan een systeem voor extra resources, in tegenstelling tot verticale schaalbaarheid waar krachtigere machines worden gebruikt. In horizontale opschaling wordt de data van een enkele database verspreid over verschillende machines die elk maar een deel opslaan.
  NoSQL systemen combineert deze twee elementen en biedt hierdoor een schaalbaar systeem aan met basis componenten.

- **Datamodel dichter bij objecten**: De meeste NoSQL systemen zijn zodanig ontworpen dat deze de vertaling van objecten naar opslag eenvoudiger of meer gelijkend maken dan RDBMS.

Deze verschillende argumenten, leiden vervolgens tot een tegenreactie op ACID: BASE.

- Basis beschikbaarheid (**B**asically **Availability**): Een applicatie werkt zo goed als heel de tijd.
- **S**oft State: De data moet op een bepaald moment niet volledig consistent zijn.
- Eventuele consistentie (**E**ventual Consistency): De database zal na enige tijd in een consistente status uitkomen. Eventuele consistentie kan op zijn beurt opnieuw onderverdeeld worden in 4 categorieën [10, slide 16]:
  - *Read your own writes* consistentie: Ongeachte van de server waarop een gebruiker leest, zal hij zijn update onmiddellijk correct lezen.
  - *Session* consistentie: De gebruiker zal zijn updates onmiddellijk kunnen lezen binnen dezelfde sessie, een sessie is hierdoor meestal gelimiteerd tot een enkele databaseserver.
  - *Casual* consistentie: Als een gebruiker versie X leest en vervolgens versie Y schrijft, zal elke gebruiker die versie Y leest ook versie X lezen.
  - *Monotonic Read* consistentie: Dit levert monotone tijdsgaranties dat een gebruiker enkel recentere data versies in de toekomst zal lezen.

Classificatie van NoSQL systemen

Er zijn vele NoSQL systemen ontworpen gedurende de laatste jaren, elk met hun eigen variëteit en functionaliteit. Er bestaan verschillende manieren om de systemen te classificeren, maar één van de meest gebruikte doet dit op basis de data modelering, een korte vergelijking bevindt zich in tabel 2.1.

Soort	Performantie	Schaalbaarheid	Flexibiliteit	Complexiteit	Functionaliteit
Column	hoog	hoog	gematigd	laag	minimaal
Document	hoog	variabel(hoog)	hoog	laag	variabel (laag)
Graph	variabel	variabel	hoog	hoog	graph theory
Key-Value	hoog	hoog	hoog	geen	variabel (geen)

Tabel 2.1: Classificatie en categorisatie van NoSQL DBMS door Scofield en Popescu. [16] [13]

Column Model In een column-gebaseerd systeem wordt de data opgeslagen per kolom in plaats van de traditionele manier, per rij. Deze aanpak werd in eerste instantie gedaan voor analyse van business intelligentie. Het systeem is geïnspireerd door de paper van Google's Bigtable [2]. [17]

Graph Model In een grafen model, wordt de data voorgesteld en opslagen volgens de grafen theorie: knopen, lijnen en eigenschappen op de knopen en lijnen. [1].

Document Model Document systemen zijn volgens vele de volgende stap in key-value systemen, waar deze complexere structuren toe laten, dit door middel van meerdere key/value paren per element. [17]
Een document moet geen vaste structuur hebben maar elk document op zich kan verschillende velden hebben, dit kan bijvoorbeeld gezien worden bij boeken. Waar een bepaald boek een recept is, kan een ander een deel zijn van een trilogie. Bij het eerste kan de kooktijd bijvoorbeeld apart opgeslagen worden en bij de tweede de andere boeken.

Key/Value Model Key-value systemen hebben een heel eenvoudig data model, data kan opgeslagen, opgevraagd en verwijderd worden op basis van een key. De informatie die in de database zit, is de waarde voor die key.
Met dit eenvoudig model en functionaliteit die weinig complexiteit introduceren, kan er gestreefd worden naar een hoge performantie, schaalbaarheid en flexibiliteit. [17]

extra conclusion text?

2.3 Bespreking van verschillende DBMS

De verschillende categorieën databases zijn besproken, voor deze thesis zullen enkele relationele en NoSQL DBMS in meer detail worden besproken. Databases uit de volgende 4 categorieën komen verder aanbod, er is gekozen om de Graph NoSQL DBMS niet te bespreken omdat deze een heel andere dataset ondersteunen als de overige categorieën.

- Column NoSQL DBMS: Cassandra, HBase
- Document NoSQL DBMS: Apache CoucheDB, MongoDB
- Key-Value NoSQL DBMS: LightCloud (Tokyo), MemCache, Redis, Riak, Project Voldemort
- Relationele NoSQL DBMS: MySQL, Pgpool-II (PostgreSQL)

2.3.1 Column database

Nalezen

Cassandra

Website: `http://cassandra.apache.org/`
Cassandra is een database die gebaseerd is op 2 verschillende systemen, Amazon's

Dynamo en Google's Bigtable, wat voor een combinatie van een column- en key-value-based database zorgt.

De query taal is beperkt tot 3 operaties: get, insert en delete [9], waar de laatste waarde in geval van een conflict zal opgeslagen worden.

De database kan gedistribueerd uitgerold worden waar door middel van partitionering en een consistent hashing algoritme de data verspreid wordt over de verschillende instanties. Om beschikbaarheid van de data te hebben bij een failure, wordt deze gerepliceerd over verschillende instanties met verschillende configuratie modellen.

HBase

Website: *http://hbase.apache.org/*
HBase is een database die gebaseerd is op Google's BigTable en draait boven op HDFS, Hadoop Distibuted File System.

De query taal voor HBase bestaat uit 4 elementen, een get, put en delete als standaard operaties en een scan om over verschillende rijen te gaan.

Voor het gedistribueerd draaien van de database, wordt de database ingedeeld in regio's. Vervolgens is een verantwoordelijk voor regio's Regionserver om de data van regio's op te slaan. Daarnaast is er ook nog afhankelijkheid van Zookeeper voor management van regio's en name en data-instanties als opslag.

2.3.2 Document database

Apache CoucheDB

Website: *http://couchdb.apache.org/*
Apache CoucheDB is een document database waar alles wordt voorgesteld met behulp van JSON. De database kan bevraagd worden door middel van Map-Reduce, de map gebeurd door een *view*, een JavaScript-functie die de gegevens zal selecteren. Nadien kan met een reduce view de data geaggregeerd worden.

Bij het gedistribueerd uitrollen zal de data met consistent hashing over verschillende instanties verdeeld worden waar elke instantie een zelfde rol heeft. Nu zal CoucheDB enkel updates van data van instantie veranderen en niet data automatisch load-balancen. Ook is het mogelijk om een exacte replica van de ene naar de andere instantie te sturen, dit wordt bijvoorbeeld handig indien documenten naar een laptop gesynchroniseerd worden om later offline te kunnen werken.

In een gedistribueerde omgeving ziet CouchDB conflicten niet als een exceptie maar als een normale omstandigheid. Wel zullen updates atomisch per rij afgewerkt worden op een enkele instantie, zodat hier geen conflict in kan bestaan. Maar indien een conflict optreedt, is het aan de bovenliggende applicatie om deze af te handelen.

MongoDB

Website: *http://www.mongodb.org/*
MongoDB is een document database waar de data wordt voorgesteld aan de hand van BSON, een binaire vorm vergelijkbaar met JSON. Data kan ingegegeven worden via JSON aangezien er een eenvoudige map mogelijk is.

Er is een uitgebreide query taal, waar er naast het invoegen, verwijderen en opvragen van een document ook talrijke zoekparameters meegegeven kunnen worden: dit gaat van zoeken op een enkel veld tot conjuncties, sorteren, projecties, ...

MongoDB kan in een gedistribueerde omgeving opgezet worden met een opsplitsing tussen het redundant opslaan van data en het verdelen van data. Het redundant opslaan kan gedaan worden door het combineren van instanties in een ReplicaSet waar er een master-slave configuratie is. Daarnaast kan data ook verdeeld worden over verschillende instanties of replica sets, dit kan door middel van het configureren van shards. Conflicts worden opgevangen door de master waar er telkens een meerderheid van de instanties nodig is voor de data.

2.3.3 Key-Value database

LightCloud (Tokyo)

Website: *http://opensource.plurk.com/LightCloud/*
LightCloud is een gedistribueerde uitbreiding van Tokyo Tyrant. Tokyo Tyrant is op zijn beurt een uitbreiding op Tokyo Cabinet en voegt de mogelijkheid tot externe connecties aan Cabinet toe. Cabinet is het basis pakket.

De query taal is gelimiteerd tot 5 operaties: get, put, delete, add en een iterator om over de keys te gaan. Met add wordt er data aan een bestaand element toegevoegd.

LightCloud levert een gedistribueerde database met master-master synchronisatie. Met behulp van een consistent hashing algoritme en 2 hash rings, wordt de data verdeeld over verschillende instanties met de nodige redundantie. De eerste ring is verantwoordelijk voor de lookups oftewel het lokaliseren van de keys, de storage ring is verantwoordelijk voor het opslaan van de verschillende waarden.

MemCache

Website: *http://memcached.org/*
MemCache is een veel gebruikt systeem waarin al de data in RAM geheugen wordt gehouden en alhoewel er ondersteuning is met behulp van MemCacheDB voor persistentie is deze database niet bedoeld voor persistente opslag.

Omdat één van de vereisten was dat de data persistent moest zijn, is dit systeem niet verder onderzocht in deze context.

Redis

Website: http://www.redis.io/
Redis is een key-value database met de mogelijkheid tot opslaan van complexe datastructuren zoals lijsten, sets en mappen. Naast de standaard instructies om een enkele waarde toe te voegen, zijn er specifieke commando's om operaties op de complexere objecten te doen, het verwijderen van een lid van een bepaalde set. Redis biedt ook ondersteuning voor transacties en heeft deze de mogelijkheid tot expire, hierdoor zal een waarde automatisch vergeten worden na een meegegeven tijd.

De database wordt volledig in geheugen geplaatst maar ondersteund 2 soorten van persistentie, oftewel door middel van RDB, oftewel met een AOF log. Bij RDB worden er over tijd snapshots gemaakt van de database en weggeschreven op harde schijf. In het geval van AOF wordt elke schrijfoperaties weggeschreven en kan de database opgebouwd worden met behulp van deze lijst.

Tenslotte heeft Redis momenteel een relatief beperkte mogelijkheid tot een gedistribueerde database. Het is mogelijk om data over verschillende instanties te distribueren met behulp van sharding welke op voorhand gedefinieerd dient te worden en is er ook de mogelijkheid tot master-slave opstelling met automatische failure detection. De laatste is nog wel in beta al is het al mogelijk om te gebruiken. Tenslotte is er naar de toekomst ook meer uitbreiding op komst met behulp van Redis Cluster waar data automatisch verspreid wordt over verschillende instanties.

Riak

Website: http://basho.com/riak/
Riak is een key-value database met de mogelijkheid tot opslaan van strings, JSON en XML. Daarnaast heeft deze standaard operaties maar hier enkele uitbreidingen op gemaakt. Allereerst is het mogelijk om secundaire indexen te definiëren op de elementen, MapReduce toe te passen en een full-text search.

Riak is gebouwd om gedistribueerd te draaien waar al de instanties evenwaardig zijn. Data wordt verdeeld over de verschillende instanties en elk element wordt standaard op 3 verschillende instanties opgeslagen. Indien een bepaalde instantie faalt, wordt dit met een gossiping algoritme verspreid over de verschillende instanties waarmee een naburige instantie overneemt. Daarnaast is er automatische recovery indien een instantie terug online komt.

Project Voldemort

Website: http://www.project-voldemort.com/
Project Voldemort is een key-value store met enkel 3 basis operaties: get, put en delete met de mogelijkheid voor als keys en values strings, serializable objecten, protocol buffers of raw byte arrays te gebruiken.

Deze database ondersteunt verschillende modes van distributie. De opbouw bestaat uit verschillende lagen, elk met hun eigen gedefinieerde functie. Met behulp van deze lagen kan de ontwikkelaar extra functionaliteit toevoegen met behulp van een extra laag om de applicatie meer te finetunen naar zijn uitwerking. Data wordt verdeeld met behulp van consistent hashing over de verschillende servers, waarbij data verschillende keren wordt bijgehouden om ervoor te zorgen dat de data nog beschikbaar is in het geval van falen.

2.3.4 Relationele database

MySQL

Website: *http://www.mysql.com/*
MySQL is een relationele database waarin data kan voorgesteld worden in verschillende vormen, beginnend met een bool tot een blok tekst. Daarnaast zijn de query mogelijkheden uitgebreid.

De uitbreiding van een gedistribueerd systeem is bij MySQL ingebouwd door middel van een Master-Slave configuratie. Als mysqlfailover een faal detecteert in één van de slaven, zal de database verder werken, bij het falen van de master zal een nieuwe master handmatig aangeduid moeten worden. Ook de recovery moet handmatig gestart worden, waarna indien gewenst de originele master opnieuw als master kan gezet worden (bv. omdat deze de krachtigste computer is).

Pgpool-II (PostgreSQL)

Website: *http://www.pgpool.net/*
PostgreSQL is een relationele database en heeft soortgelijke specificaties als MySQL op een enkele computer, verschillende soorten data kunnen voorgesteld worden met uitgebreide query mogelijkheden.

Enkel als de database ook gedistribueerd moet uitgerold worden, is er een verschil. Bij PostgreSQL is er hiervoor standaard geen ondersteuning hiervoor maar moet er op externe elementen vertrouwd worden. Er bestaan verschillende componenten soorten systemen, maar aangezien er load-balancing nodig is en een zo recent mogelijk pakket, is er gekozen voor Pgpool-II, een vergelijking van de systemen kan gevonden worden op de wiki van PostgreSQL [14].

Pgpool-II heeft verschillende mode, maar aangezien er de nood was voor replicatie, is er gekozen om de replicatie mode in te stellen. Hierdoor komt de database in een Master-Slave situatie waar er het falen gedetecteerd wordt op het moment een connectie actief is. Indien een instantie terug online is, moet er manueel het recovery process opgestart worden.

Andere configuraties zijn mogelijk, onder andere met het opzetten van caching en nog andere elementen.

2.4 Conclusie

De databasemanagementsystemen zijn voor meer dan 40 jaar actief. Met de opkomst van het relationele model in 1969 door E. Codd [3], zijn gedurende vele jaren de relationele DMBS de leidende technologie geweest. Maar de afgelopen 10 jaar is er in het landschap veel veranderd met de opkomst van de NoSQL DBMS die afstappen van het ACID naar BASE, meer gefocust op het werken op grote data in een gedistribueerde omgeving. Er worden talrijke assumpties gemaakt over NoSQL, ze zouden een hogere performantie leveren, een hogere beschikbaarheid (availability) en zouden leiden onder het principe van eventuele consistentie (eventual consistency).

2.4.1 Performantie benchmarking

Nog is bekijken binnenkort

Indien men verschillende DBMS wilt vergelijken bestaan er al enkele tools en studies om de performance te kunnen vergelijken, een blogpost van A. Popescu [12] geeft een overzicht van verschillende benchmarking tools.

Als eerste hebben vele DBMS interne benchmarking tools, waarmee de database op verschillende configuraties kan getest worden en vergeleken worden. Deze resultaten zijn nuttig indien het DBMS al gekozen is en men bezig is met het aanpassen van de parameters of om te onderzoeken wat net de bottleneck is in een bepaald systeem. Een voorbeeld hiervan is mongoperf[1] voor MongoDB.

Andere studies focussen op het testen van verschillende systemen en daarbij kunnen verschillende doelstellingen zijn: het ontwikkelen van een breed toepasbare tool, het testen van een grote verscheidenheid van DBMS of het testen van een specifieke categorie van systemen. Elke van deze benchmarking brengt extra kennis bij maar heeft ook zijn beperkingen. Het totaal pakket van al de testen kan een gebruiker een overzicht geven.

De eerste categorie, het **ontwikkelen van een tool**, heeft als grote voordeel dat andere gebruikers nadien de testen opnieuw kunnen uitvoeren met de huidige systemen. Het is namelijk niet gegarandeerd dat het resultaat van een jaar geleden met de nieuwste versie nog te vergelijken is. Het grootste nadeel is het soort testen dat kan uitgevoerd worden, er is een grote variëteit aan systemen elk met hun eigen datastructuur en query mogelijkheden. De tool moet dus een gemeenschappelijke subset zoeken en enkel dit soort queries kunnen getest worden. Een voorbeeld van zulk een tool de opensource tool YCSB[5]. In deze tool kan elk DBMS getest worden zolang een basisset van 5 queries kan implementeren: het invoegen, updaten, verwijderen en opvragen van een enkel record met daarnaast ook de mogelijkheid tot range queries, met behulp van 1 query een verzameling van records tegelijk op te vragen.
Sommige systemen ondersteunen bepaalde queries niet onmiddellijk maar bevatten wel de functionaliteit om deze via via te implementeren, bijvoorbeeld een update kan

[1] http://docs.mongodb.org/manual/reference/program/mongoperf/

geïmplementeerd worden door het opvragen, verwijderen en vervolgen invoegen van de geüpdatete record.

Een volgende categorie zijn de **resultaten van gerelateerde DBMS**, in deze categorie zijn er voornamelijk resultaten te vinden van systemen met hetzelfde datamodel. Het grote voordeel hieraan is dat deze systemen in de meeste gevallen een vrij gelijkaardige set aan query mogelijkheden bevatten waardoor er meer diepgang is dan tussen meer verschillende systemen. Een voorbeeld van zulk onderzoek is gedaan door P. Pirzadeh et al[11] voor de key-value systemen, meer specifiek is er gefocust op het uitvoeren van range queries tussen Cassandra, HBase en Voldemort. Hoewel in de categorisatie van deze thesis de eerste twee onder column DBMS vallen, zijn deze nog vrij gelijklopend.
In deze categorie vallen ook de resultaten die meestal getoond worden op de website van de DBMS, een vergelijkende benchmark met andere soortgelijke systemen. Hoewel de resultaten niet altijd volledig objectief zijn, kan de gevolgde methode wel interessant zijn. Een voorbeeld van deze studie is de key-value benchmarking van VoltDB[8] waar Cassandra en VoltDB vergeleken worden, een belangrijke kanttekening is dat de auteur zelf al aanhaalt dat de systemen vrij verschillend zijn zoals appels en appelsienen.

GLS check

Als laatste categorie, zijn er de **resultaten van verschillende DMBS** waar zeer verscheidene systemen met elkaar getest worden. De belangrijkste voordeel is dat er resultaten zijn die verschillende soorten met elkaar vergelijken en waardoor niet alleen verschillen in het datamodel kunnen vergeleken worden in toekomstige studies maar ook performantie verschillen. Het nadeel is ook hier dat er een gemeenschappelijke subset gevonden moet worden, hierdoor kunnen bepaalde databases hun kracht net niet laten zien. Verschillende van deze onderzoeken zijn [18] en [15]. Deze laatste maakt gebruik van de YCSB tool die hierboven besproken was.

2.4.2 Beschikbaarheids en consistentie testen

In de vorige sectie is

Hoofdstuk 3

Methodiek van de testen

3.1 Vereisten van de systemen

- **Persistentie**:
- **Data Distributie**
- **Replicatie**
- **Open-source**

3.1.1 Gekozen systemen

HBase

MongoDB

Pgpool-II (PostgreSQL)

Hoofdstuk 4

Implementatie

4.1 Doelstelling

4.2 Gebruik IMP

4.3 Gebruik YCSB

Hoofdstuk 5

Observaties

Hoofdstuk 6

Analyse van de resultaten

Hoofdstuk 7

Conclusie

Bijlagen

Bibliografie

[1] Kurt Bollacker e.a. „Freebase: a collaboratively created graph database for structuring human knowledge”. In: *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. ACM. 2008, p. 1247–1250.

[2] Fay Chang e.a. „Bigtable: A distributed storage system for structured data”. In: *ACM Transactions on Computer Systems (TOCS)* 26.2 (2008), p. 4.

[3] E. F. Codd. „A Relational Model of Data for Large Shared Data Banks”. In: *Commun. ACM* 13.6 (jun 1970), p. 377–387. ISSN: 0001-0782. DOI: 10.1145/362384.362685. URL: http://doi.acm.org/10.1145/362384.362685.

[4] Edgar F Codd. „A relational model of data for large shared data banks”. In: *Communications of the ACM* 13.6 (1970), p. 377–387.

[5] Brian F Cooper e.a. „Benchmarking cloud serving systems with YCSB”. In: *Proceedings of the 1st ACM symposium on Cloud computing*. ACM. 2010, p. 143–154.

[6] Ramez Elmasri en Shamkant Navathe. *Fundamentals of Database Systems*. 6th. USA: Addison-Wesley Publishing Company, 2010. ISBN: 0136086209, 9780136086208.

[7] Jim Gray. „Data Management: Past, Present, and Future”. In: *arXiv preprint cs/0701156* (2007).

[8] J Hugg. *Key-value benchmarking*. 2010. URL: http://voltdb.com/blog/voltdb-benchmarks/key-value-benchmarking/ (bezocht op 06-07-2014).

[9] Avinash Lakshman en Prashant Malik. „Cassandra: A Decentralized Structured Storage System”. In: *SIGOPS Oper. Syst. Rev.* 44.2 (apr 2010), p. 35–40. ISSN: 0163-5980. DOI: 10.1145/1773912.1773922. URL: http://doi.acm.org/10.1145/1773912.1773922.

[10] Todd Lipcon. „Design Patterns for Distributed Non-Relational Databases”. In: *Design Patterns for Distributed Non-Relational Databases* (2009).

[11] Pouria Pirzadeh, Junichi Tatemura en Hakan Hacigumus. „Performance evaluation of range queries in key value stores”. In: *Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on*. IEEE. 2011, p. 1092–1101.

[12] A. Popescu. *NoSQL benchmarks and performance evaluations*. 2010. URL: http://nosql.mypopescu.com/post/734816227/nosql-benchmarks-and-performance-evaluations (bezocht op 06-07-2014).

[13] Alex Popescu. *Presentation: NoSQL at CodeMash – An Interesting NoSQL categoriza- tion.* Feb 2010. URL: http://nosql.mypopescu.com/post/396337069/presentation-nosql-codemash-an-interesting-nosql (bezocht op 03-02-2014).

[14] Postgresql. *PostgreSQL - Replication, Clustering, and Connection Pooling*. Okt 2013. URL: http://wiki.postgresql.org/wiki/Replication,_Clustering,_and_Connection_Pooling (bezocht op 03-02-2013).

[15] Tilmann Rabl e.a. „Solving big data challenges for enterprise application performance management". In: *Proceedings of the VLDB Endowment* 5.12 (2012), p. 1724–1735.

[16] Ben Scofield. *NoSQL – Death to Relational Databases(?)* Jan 2010. URL: http://www.slideshare.net/bscofield/nosql-codemash-2010 (bezocht op 03-02-2013).

[17] Christof Strauch. *NoSQL Databases*. 2010. URL: http://www.christof-strauch.de/nosqldbs.pdf.

[18] Bogdan George Tudorica en Cristian Bucur. „A comparison between several NoSQL databases with comments and notes". In: *Roedunet International Conference (RoEduNet), 2011 10th*. IEEE. 2011, p. 1–5.

Fiche masterproef

Student: Thomas Uyttendaele

Titel: Automatisch uitrol van database systemen en vergelijking van beschikbaarheid

Engelse titel: Automatisch uitrol van database systemen en vergelijking van beschikbaarheid

UDC: 681.3

Korte inhoud:

Hier komt een heel bondig abstract van hooguit 500 woorden. LaTeX commando's mogen hier gebruikt worden. Blanco lijnen (of het commando `\par`) zijn wel niet toegelaten!

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Thesis voorgedragen tot het behalen van de graad van Master of Science in de ingenieurswetenschappen: computerwetenschappen, hoofdspecialisatie Gedistribueerde systemen

Promotor: Prof. dr. ir. Wouter Joosen

Assessor: Ir.

Begeleider: Ir. B. Vanbrabant