Technolution
AUTOMATION TECHNOLOGY

Bomb-Proof Server

Benodigde technieken en producten voor een High Availability systeem en hun consequenties voor beschikbaarheid

Arend-Jan Tetteroo (s0002615)

Afstudeeropdracht
Periode: Februari – Juli 2006
Vakgroep: Databases (DB)
Opleiding: Informatica (CS),
Universiteit Twente (UT)

Beoordelingscommissie:
Dr. M.M. Fokkinga (UT)
Ir. H.J.W. van Heerde (UT)
Ing. J.M. Molenmaker (Technolution)

Juli 2006

SAMENVATTING

Titel: Bomb-Proof Server, benodigde technieken en producten voor een High Availability system en hun consequenties voor beschikbaarheid.

Binnen het Bomb-Proof Server project is onderzoek gedaan naar welke technieken er beschikbaar zijn en noodzakelijk zijn om data en applicaties beschikbaar te maken en te houden. High Availability wordt steeds belangrijker in de 24-uurs economie en de afhankelijkheid van de ICT-systemen neemt steeds verder toe.

Er is een inventarisatie gemaakt van de noodzaak tot High Availability, wat het inhoudt en welke definities voor High Availability gebruikt worden. Ook de impact en invloedsfactoren bij organisaties komt aan bod. Benodigde technieken voor beschikbaarheid zoals redundantie, replicatie van data en het clusteren van machines zijn in kaart gebracht. Van alle technieken zijn producten gezocht op de markt, die deze implementeren en daarmee een oplossing kunnen zijn om systemen beschikbaar te houden als componenten uitvallen. Uiteindelijk zijn drie verschillende soorten technieken getest om te evalueren of deze geschikt zijn voor een projectopstelling. Niet één van de geteste producten bleek een volledige oplossing te bieden, veelal waren nog externe softwareaanpassingen nodig of moest met bepaalde nadelen rekening worden gehouden. Daarnaast is getracht een inventarisatie te maken van best-practices binnen Nederlandse organisaties op het gebied van High Availability.

SUMMARY

Title: Bomb-Proof Server, needed technologies and products for a High Availability system and their consequences for availability.

For the Bomb-Proof Server project research has been done to see what technologies are available and necessary to make and keep data and applications available. High Availability is getting more and more important because of the 24-hour economy and the increasing dependability on ICT systems in our lives.

In this project the need for High Availability, the history and definitions have been reviewed, as are the impact and influence factors for organizations that want or need to implement a High Availability solution. Needed technologies for availability like redundancy, replication and clustering of machines have been summarized. A market scan has taken place in which the products were matched onto technologies. Three different technologies were tested through three products to evaluate their use for a project. None of the three provided a complete solution for the problem, they all needed other products or had downsides that need to be taken into account when using those products. Besides that a small review was made of best-practices in Dutch organizations with regard to High Availability.

VOORWOORD

Voor u ligt het verslag van mijn afstudeeropdracht bij het bedrijf Technolution in Gouda in het kader van mijn doctoraalopleiding Informatica (Computer Science) aan de Universiteit Twente. Binnen deze afstudeeropdracht heb ik onderzoek gedaan naar de benodigde technieken en producten om systemen beschikbaar te maken en te houden, ook al vallen componenten of volledige locaties uit waar deze systemen zich bevinden.

Graag wil ik op deze plek Technolution bedanken voor de geboden gastvrijheid tijdens mijn afstuderen. Vooral Jasper en Enno zijn erg behulpzaam geweest in het aandragen van ideeen en het sturen in een goede richting. Tevens wil ik de andere collega's bedanken voor de getoonde interesse en de verschillende inzichten over mijn onderzoek.

Ook mijn begeleiders vanuit de Universiteit, Maarten en Harold, wil ik graag bedanken voor hun hulp en het doorlezen van de rapporten.

Ik wens u veel plezier bij het lezen van dit verslag.

Arend-Jan Tetteroo
Gouda, Juli 2006

INHOUD

1. INLEIDING

Het Bomb-Proof Server project is een afstudeerproject met als doel het opdoen van kennis en ervaring met technieken en producten voor het komen tot een hoog beschikbaar systeem. De context van het Bomb-Proof Server project wordt beschreven in paragraaf 1.1. Paragraaf 1.2 geeft de doelstelling van deze afstudeeropdracht. Paragraaf 1.3 geeft de probleemstelling en een overzicht van de verschillende onderzoeksvragen die bij dit project aan de orde zijn gekomen, waarna in paragraaf 1.4 de aanpak wordt toegelicht. Paragraaf 1.5 beschrijft de opbouw van dit document.

1.1 Context

Technolution is een toonaangevend projectbureau in de technische automatisering. Technolution ontwikkelt hard- en softwareoplossingen voor technische informatiesystemen en embedded systemen. Een van de grote opdrachtgevers is Rijkswaterstaat, waarvoor projecten als het GladheidMeldSysteem en Weigh in Motion (Realtime vrachtwagenmetingen op snelwegen) worden ontwikkeld. Voor deze projecten wordt hardware en software ontwikkeld die de verschillende sensors kan uitlezen en de gegevens kan opslaan en weergeven in een informatiesysteem voor de klant. Ook andere meetnetten zoals het Nationaal Meetnet Radioactiviteit voor het Rijksinstituut voor Volksgezondheid en Milieu worden door Technolution ontwikkeld. Andere oplossingen zijn in-car informatiesystemen, embedded systemen in Douwe Egberts-koffieautomaten, geavanceerde printkoppen voor Agfa A0-printers of image processing machines voor medische toepassingen.

Voor een aantal systemen en oplossingen wordt de beschikbaarheid van de oplossing of gegevens steeds belangrijker. Klanten vragen om hogere garanties met betrekking tot de beschikbaarheid van gegevens en de geboden oplossing. Redenen zijn het verlies van klanten of gewerkte uren door werknemers bij uitval van de systemen. Systemen, zoals het genoemde gladheidmeldsysteem, waar materiele of zelfs persoonlijke schade kan ontstaan als een systeem niet meer beschikbaar is, vragen zelfs om 100% beschikbaarheid. In tegenstelling tot het verleden wordt Technolution steeds vaker gevraagd om ook de hosting van de oplossing te verzorgen en wordt dit niet meer slechts door de klant gedaan. Daarbij komt dat de opdrachtgever in een aantal gevallen zelf niet voor de gevraagde garanties kan zorgen in de eigen omgeving.

Omdat in de toekomst de vraag naar 100% availability alleen maar zal toenemen, wil Technolution een project opstarten waarin de aanwezige kennis wordt uitgebreid met de verschillende technieken en standaarden om een zo hoog mogelijke beschikbaarheid te kunnen garanderen. Dit project zal zich zowel richten op de verschillende hardware oplossingen als op database en applicatie oplossingen. Hierbij kan gedacht worden aan redundante hardware, geografisch verspreide systemen, load balancing, replicatie en synchronisatie van databases en filesystemen, clusteringoplossingen en transactiesystemen.

Drie belangrijke aspecten bij beschikbaarheid, namelijk de benodigde systemen voor communicatie met de buitenwereld, de beveiliging van data toegang en het optreden van fouten door menselijk handelen, vallen buiten het bereik van dit project. Dit project is het Bomb-Proof Server project genoemd, waarbij de “Bomb” zowel letterlijk als figuurlijk opgevat dient te worden, zodat zelfs bij de uitval van een complete “site” de oplossing gewoon door kan draaien.

1.2 Doelstelling

Het doel van het Bomb-Proof Server project is het uitbreiden van de kennis over High Availability (HA). Onderzocht dient te worden of het mogelijk is om tot een 100% beschikbaarheid van data te komen. De theorie achter High Availability dient op een overzichtelijke manier in kaart te worden gebracht, waarbij ook aandacht wordt geschonken aan de verschillende oplossingen die op de markt beschikbaar zijn. Door middel van een praktische test dient aangetoond te worden welke technieken bruikbaar zijn in projecten van Technolution en welke garanties daarmee voor de beschikbaarheid geboden kunnen worden.

1.3 Onderzoeksvragen

Om de hierboven gedefinieerde doelstelling te behalen, zijn een aantal verschillende onderzoeksvragen opgesteld waarop een antwoord gevonden dient te worden. Deze paragraaf geeft een overzicht van de verschillende onderzoeksvragen die aan bod komen binnen dit project. Deze vragen variëren van de theorie en technieken voor High Availability tot welke producten er in de praktijk gebruikt worden, welke toepasbaar zijn voor Technolution en hoe andere organisaties omgaan met de beschikbaarheidproblematiek. Tussen haakjes staat aangegeven in welke hoofdstukken een antwoord wordt gegeven op deze vragen.

Welke definitie wordt gebruikt voor High Availability?

- Wat is de geschiedenis van HA, waar komt het vandaan? (2.1)
- Wat wordt er verstaan onder beschikbaarheid, wanneer is een systeem wel of niet beschikbaar? (2.2)
- Hoe worden de beschikbaarheidcijfers en percentages berekend en gemeten? Wat zeggen deze cijfers? (2.2)

Wat zijn de voornaamste kwesties die zorgen voor een niet optimale beschikbaarheid?

- Is het juiste product voldoende voor hoge beschikbaarheid of bieden deze producten niet alles wat benodigd is? Wat is er dan nog meer noodzakelijk? (2.4, hoofdstuk 4 en hoofdstuk 7)
- Welke aspecten spelen een rol bij beschikbaarheid? (2.3)
- Waar moeten organisaties rekening mee houden bij HA oplossingen (2.4)

Welke technieken zijn er voorhanden, wat doen ze, hoe werkt het, wat zijn de voor en nadelen van een techniek en wanneer zijn ze goed om te gebruiken? (Hoofdstuk 3)

Welke technieken worden toegepast door database vendors in hun producten en hoe verhouden die zich tot elkaar? (Hoofdstuk 4)

- Hoe bepaal je welke techniek nodig is voor een bepaald probleem? (4.1)
- Welk product biedt deze techniek aan? (4.1)
- Op welke onderdelen verschillen de producten van elkaar? (4.2)

Op welke manier pakken op het oog professionele organisaties dit aan?

- Wat zijn de best-practices op High Availability gebied? (Hoofdstuk 5)
- Welke bedrijfsmatige maatregelen moet je nemen om een oplossing beschikbaar te houden?

Op welke argumenten bepaal je de keuze voor een bepaalde oplossing?

- Technische haalbaarheid en complexiteit (hoofdstuk 3, 4 en 7)
- Organisatorische aspecten, als kosten en baten, imago van de organisatie en processen (hoofdstuk 2 en 5)

Welke producten zijn toepasbaar voor Technolution in projecten?

- Welke producten zijn interessant voor Technolution om te testen? (Hoofdstuk 6)
- Op welke manier kan je deze producten testen? (Hoofdstuk 6)
- Aan welke eisen moeten deze producten dan voldoen? (Hoofdstuk 6)
- Bieden oplossingen op database niveau een volledige oplossing of zijn extra producten benodigd? (Hoofdstuk 7)
- Wat voor invloed hebben de verschillende HA functies op de performance van een systeem? Blijft het systeem voldoende performance bieden om nog bruikbaar te zijn? (Hoofdstuk 7)

1.4 Aanpak

Om een antwoord te kunnen geven op de verschillende onderzoeksvragen is een literatuuronderzoek gedaan. Uit dit onderzoek kwamen de verschillende definities en mogelijke technieken naar voren om tot een hoog beschikbaar systeem te komen.

De verschillende producten die een implementatie bieden van de gevonden technieken zijn door middel van een marktverkenning in kaart gebracht. Om te bepalen welke producten voor Technolution interessant zijn voor gebruik in projecten is vervolgens een testopstelling opgesteld waarin een bepaalde projectsetting werd gesimuleerd.

Op deze simulatie zijn een drietal verschillende producten uitgekozen om te bepalen of deze een goede oplossing bieden en aan de gestelde eisen van het project voldoen. Buiten het correct functioneren van de beschikbaarheidopties is ook gekeken naar de invloed van deze opties op de performance van de verschillende producten, omdat een systeem dat niet voldoende performance biedt niet gebruikt zal worden.

Daarnaast is gekeken naar de best-practices bij grote organisaties met betrekking tot beschikbaarheid, om erachter te komen hoe deze organisaties in de praktijk omgaan met deze kwestie.

1.5 Opbouw van het verslag

Voordat ingegaan kan worden op de verschillende technieken en producten die een rol spelen in het High Availability vraagstuk, is het noodzakelijk eerst te weten wat dit High Availability begrip betekent. Hoofdstuk twee biedt een introductie tot het begrip High Availability. Het biedt een antwoord op de vragen wat High Availability inhoudt, waar het vandaan komt en waarom het steeds belangrijker wordt. Zowel een woordelijke als een rekenkundige definitie zullen worden gegeven van het begrip High Availability. De verschillende onderdelen waarmee rekening moet worden gehouden bij een beschikbaarheidoplossing, zoals betrouwbaarheid en performance worden behandeld. Ook de invloedsfactoren voor organisaties in het organiseren van beschikbaarheid voor applicaties of services komen aan bod.

In hoofdstuk drie worden de verschillende technieken beschreven die nodig zijn om tot een beschikbaar systeem te komen. Aan bod komen redundantie, replicatie en het clusteren van database, besturingssysteem en middleware.

De verschillende producten die deze technieken implementeren om zo een beschikbaarheidoplossing aan te bieden op de High Availability markt worden beschreven en vergeleken in hoofdstuk vier. Via een beslissingsdiagram kunnen de juiste techniek en daarbij het juiste product worden gevonden bij een bepaalde gewenste situatie.

Hoofdstuk vijf beschrijft een aantal praktijkcases waarin bij organisaties wordt bekeken hoe wordt omgegaan met beschikbaarheid in de praktijk.

Van de beschreven producten in hoofdstuk vier zijn er een drietal uitgekozen om te gebruiken voor een testopstelling. In deze testopstelling wordt bepaald welke producten goed functioneren, wat de gevolgen zijn van het toepassen van de beschikbaarheidopties en welke producten goed toepasbaar zijn in projecten van Technolution. Deze testopstelling staat beschreven in hoofdstuk zes. De resultaten van deze testen zijn beschreven in hoofdstuk zeven waarin tevens een uitspraak wordt gedaan over de geschiktheid van de producten voor het gesimuleerde project.

Ten slotte wordt in hoofdstuk acht een afsluitende conclusie en aanbevelingen gegeven.

Een aantal verschillende afkortingen en begrippen zijn opgenomen in een verklarende woordenlijst achter in dit verslag.

2. INTRODUCTIE TOT HIGH AVAILABILITY

Dit hoofdstuk geeft een introductie tot het begrip High Availability (HA). Voordat ingegaan kan worden op de verschillende technieken die bij High Availability gebruikt worden, is het noodzakelijk om te weten wat High Availability inhoud.

Paragraaf 2.1 beschrijft de oorsprong en geschiedenis van High Availability. In paragraaf 2.2 worden de verschillende definities van Availability en High Availability beschreven zoals die worden gebruikt in de ICT-wereld en welke definitie in dit onderzoek gebruikt wordt. Ook de beschikbaarheidcijfers en niveaus, die bepalen in welke klasse een oplossing valt, komen aan bod.

Behalve High Availability zijn ook andere aspecten van belang bij een HA oplossing, zoals de betrouwbaarheid en performance. Deze en andere aspecten worden beschreven in paragraaf 2.3. Tenslotte komen in paragraaf 2.4 de invloedsfactoren voor organisaties aan bod met betrekking tot de beschikbaarheid van systemen. Hierbij gaat het niet alleen om de producten, maar ook om de processen, procedures en werknemers die het onderhoud moeten uitvoeren. Ook de kosten voor het investeren in een oplossing en het maken van afspraken met de leverancier of hosting partij door middel van Service Level Agreements (SLA) worden beschreven.

2.1 Waarom High Availability

High Availability vindt zijn oorsprong in de steeds meer toenemende globalisering van diensten die door middel van ICT en in het bijzonder internet worden aangeboden. Door de globalisering en 24-uurs economie is de beschikbaarheid van deze ICT-systemen steeds belangrijker aan het worden. Niet alleen voor mission-critical systemen is hoge beschikbaarheid gewenst, ook voor andere applicaties is de beschikbaarheid steeds belangrijker. Het ontstaan van High Availability wordt in paragraaf 2.1.1 beschreven. Paragraaf 2.1.2 beschrijft de verschillende redenen voor de toename van High Availability in de markt.

2.1.1 Geschiedenis

De afgelopen twintig jaar heeft een grote evolutie plaatsgevonden van mainframe gebaseerde systemen naar open gedistribueerde (UNIX) systemen. Deze evolutie had een aantal oorzaken volgens Veritas [69]. De gedistribueerde systemen waren kleiner en goedkoper dan de grote mainframes. Afdelingen binnen bedrijven waren met deze systemen niet meer afhankelijk van de centrale IT-afdeling om hun eigen applicaties te kunnen gebruiken en hadden zelf controle over hun eigen systemen. Er was een keuze in “off-the-shelf” producten die het meest geschikt waren voor afdelingen en er hoefde bij een nieuwe applicatie niet te worden gewacht op budgetten van de centrale IT-afdeling. Daarbij was de plaatsing van een klein systeem minder problematisch dan een uitbreiding van het mainframesysteem.

Bijkomend voordeel was dat de impact van een uitval van een systeem beduidend kleiner was door de decentralisatie dan bij het mainframesysteem. Bij de uitval van het mainframe kon geen enkele werknemer die afhankelijk was van het mainframe nog doorwerken. Bij een uitval van een decentraal systeem had slechts de afdeling van dat systeem een probleem. Het voordeel van decentralisatie heeft echter ook een nadeel. Doordat er veel verschillende systemen worden gebruikt en deze niet allemaal op dezelfde plek staan, dient een omvangrijke administratie te worden bijgehouden en nemen de kosten en complexiteit van het beheer toe.

Door de steeds groter wordende invloed van IT op de businessprocessen, werd de impact van de uitval van deze gedecentraliseerde systemen steeds groter. Omdat componenten van systemen, zoals voedingen en hardeschijven een beperkte levensduur hadden, werden deze door de industrie als eerste voorzien van redundantie [69]. Op deze manier kon een systeem blijven doordraaien als een van de componenten die redundant waren uitgevoerd het begaf. De term Reliability, Availability and Serviceability (RAS) wordt door de industrie op dat moment geïntroduceerd. Door het verhogen van de betrouwbaarheid (Reliability) van de componenten, neemt de beschikbaarheid (Availability) toe. Door componenten eenvoudig vervangbaar te maken, zijn de systemen makkelijker te beheren en onderhouden (Serviceability). Een lijst van mogelijke RAS features wordt gegeven in [71].

Na het redundant uitvoeren van componenten, was het nog steeds mogelijk dat een volledig systeem uitviel. Om dit op te vangen werd gebruik gemaakt van zogenaamde “spares” of reservesystemen. Deze systemen hadden dezelfde hardware als het systeem waarop de applicatie draaide. Bij een uitval van het hele systeem werd de hardeschijf van dit systeem overgezet naar het reservesysteem, dat daarna de taken overnam, mits de hardeschijf niet de oorzaak van de uitval was. Dit is de basis van een handmatige fail-over, die tegenwoordig alleen nog maar softwarematig wordt uitgevoerd bij High Availability systemen. Om te voorkomen dat de kabels en hardeschijven overgezet moesten worden, zijn dual-hosting systemen ontwikkeld [69]. Deze systemen voorzagen twee systemen van opslagcapaciteit. Als één van de twee systemen uitviel, kon de tweede machine zonder het omwisselen van kabels en schijven de taken overnemen. Doordat de benodigde tijd om de zaken te repareren kleiner werd, er was immers geen ombouwactie meer nodig, nam de beschikbaarheid van de systemen toe.

Doordat er geen handmatige wissels van hardware meer nodig waren, kon door middel van software de take-over of fail-over door het secundaire systeem geautomatiseerd worden. Het verschil tussen een take-over en fail-over zit in de initiator van de overstap. Bij een take-over gebeurt dit door een persoon, omdat bijvoorbeeld onderhoud gepleegd moet worden aan het primaire systeem. In het geval van een fail-over wordt de overstap geïnitieerd door het optreden van een fout in het systeem [72]. Door middel van scripts kon, zonder tussenkomst van een beheerder, het systeem worden voortgezet. Deze scripts waren de voorlopers van de huidige Fail-over Management Software (FMS), waarmee via een grafische interface de systemen beheerd kunnen worden [69]. Een tweede probleemgebied wat wordt opgelost in FMS systemen is het kunnen detecteren

van fouten en het monitoren van systemen, zodat een fail-over plaats kan vinden. Zonder het monitoren van systemen is automatische fail-over niet mogelijk. Een systeem zonder foutdetectie zal dan ook geen hoge beschikbaarheid kunnen garanderen.

De laatste jaren is er een sterke tendens om systemen te consolideren. Consolidatie betekent dat één systeem meerdere taken uitvoert, waar in het verleden meerdere systemen voor nodig waren. Doordat de processingpower van systemen zeer groot is geworden, is het aantrekkelijker en goedkoper om meerdere taken op hetzelfde systeem uit te voeren. Dit scheelt aanzienlijk in de kosten van bijvoorbeeld stroom- en ruimtegebruik en ook het beheer wordt eenvoudiger, doordat minder systemen onderhouden moeten worden. Deze consolidatie betekent wel dat er weer gecentraliseerd wordt, waarmee de oude mainframetheorie weer terug lijkt te komen. De wensen en eisen voor een decentraal systeem zijn er echter nog steeds, waardoor nu naar een combinatie van beide wordt gezocht, die de voordelen kan combineren.

Voor geconsolideerde systemen is het nog belangrijker om beschikbaar te zijn dan voor de gedistribueerde systemen, doordat de uitval een veel grotere impact op de services heeft. Hierdoor wordt High Availability steeds belangrijker. De redenen gezien vanuit het oogpunt van de gebruiker worden beschreven in paragraaf 2.1.2.

Van de in begin jaren ‘80 gebruikte HA systeemproducenten werken er nu nog slechts enkele onder dezelfde naam. Het Tandem NonStop System is via Compaq tegenwoordig onderdeel van HP geworden als HP NonStop [70]. Stratus/32 Continuous Processing System levert als Stratus nog steeds HA oplossingen [4][21]. Tegenwoordig richten de meeste fabrikanten zich slechts op een deelgebied, zoals storage replicatie, database replicatie of een clusteroplossing. Er zijn nog slechts enkele bedrijven die een totaaloplossing bieden zoals in de jaren ‘80 het geval was.

Behalve de ontwikkeling van mainframes naar gedistribueerde systemen was er binnen vooral de academische wereld een grote behoefte aan rekenkracht voor onderzoeksdoeleinden. Doordat mainframes of supercomputers veelal te duur waren of niet voldoende performance konden bieden, werd onderzoek gestart naar het combineren van capaciteiten van systemen. Het bouwen van een systeem met meerdere COTS (Commodity/Commercial Off The Shelf) componenten bood ofwel een grotere performance ofwel een goedkopere oplossing. Een van deze oplossingen is de Beowulf Cluster technologie [73]. Doordat in een aantal gevallen geen speciale redundante hardware werd toegepast, moest gelijk een oplossing worden bedacht voor de uitval van één van de systemen. Hierdoor werd in clustering zowel het uitbreiden van capaciteit als het bieden van beschikbaarheid ontwikkeld.

2.1.2 Redenen voor High Availability

Er kunnen veel redenen worden gegeven waarom de beschikbaarheid belangrijk is bij huidige systemen en nog belangrijker wordt in de toekomst gezien vanuit het oogpunt van de stakeholders. De vijf belangrijkste zijn hieronder weergegeven:

1. **Concurrentie** - Er is grote concurrentie tussen marktpartijen, vooral op internet. Als de site van een bepaalde webwinkel niet beschikbaar is, gaat de klant binnen een paar seconden naar een andere winkel. Consumenten verwachten van internet dat dit 24 uur per dag, 7 dagen per week beschikbaar is, of dat nou een winkel, een veilingsite of een nieuwssite is. Door het *altijd open* imago is het nauwelijks mogelijk op een geschikt tijdstip onderhoud te plegen, tenzij door de website alleen een lokale functie wordt geboden. Globaal opererende websites ontbreekt het helemaal aan geschikte onderhoudstijden. Door de grote kans op imago schade, is het beschikbaar zijn van de website, inclusief backend, erg belangrijk. Een intern systeem dat niet beschikbaar is, is alleen zichtbaar voor werknemers en eventuele klanten die met dat systeem werken. Doordat internet steeds vaker wordt ingezet voor deze systemen wordt de beschikbaarheid ervan zichtbaar voor de buitenwereld.
2. **Prestatiecontracten** - In een aantal bedrijfstakken en bij de overheid is het gebruikelijk om prestatiecontracten af te sluiten. In deze prestatiecontracten wordt aangegeven welke prestatie verwacht wordt van de partijen. Omdat een deel van deze prestaties afhankelijk is van de IT-systemen, is de beschikbaarheid van deze systemen erg belangrijk. Als het systeem niet bruikbaar is, dan kan de afgesproken prestatie niet gehaald worden. Een uitval van de IT-systemen in het verkeerscentrum van de NS heeft direct tot gevolg dat er geen treinen meer kunnen rijden, waardoor het prestatiecontract van het aantal op tijd te laten rijden treinen niet gehaald kan worden. In sommige gevallen worden de prestaties zelfs afgedwongen door de wet, waardoor indirect de beschikbaarheid ook door de wet wordt verplicht.
3. **Bedrijfsprocessen** – Voor veel bedrijven is de spreuk *"IT is the business"* inmiddels waarheid geworden. Zonder een beschikbaar ICT-systeem kan het bedrijf niet meer functioneren. Bedrijven waarbij de business processen afhankelijk zijn van IT hebben groot belang bij het beschikbaar zijn van deze IT. Omdat de IT-processen steeds belangrijker worden voor deze bedrijfsprocessen en steeds vaker onmisbaar worden, wordt de beschikbaarheid van die processen ook steeds belangrijker. Het uitvallen van de elektriciteit geeft bij veel bedrijven direct een duidelijk beeld van de afhankelijkheid van technologie voor het doordraaien van de bedrijfsprocessen.
4. **Schade** - Bij systemen waarbij een uitval risico's creëert voor materiele of persoonlijke schade is High Availability zeer van belang en meestal ook geëist. Beslissingondersteunende systemen die afhankelijk zijn van meetsystemen van bijvoorbeeld gevaarlijke stoffen kunnen bij hun uitval niet gebruikt worden om goede besluiten te nemen. Dit levert dan extra risico's voor persoonlijke of materiele schade op, doordat de benodigde informatie om correct op te kunnen treden ontbreekt.

5. **Rampen** – Het kunnen optreden van rampen of terroristische acties is een gebied binnen de High Availability markt waar sinds 11 september 2001 steeds meer aandacht aan wordt besteed. Ook de vele orkanen in Amerika worden door de producenten aangewend om hun producten te verkopen. Bij deze systemen wordt veelal gesproken over “Disaster Recovery”, ofwel rampherstel. Hoewel veel aandacht uitgaat naar terrorisme, is de kans op zaken als overstromingen en brand veel groter en meer voorkomend in datacentra. Oplossingen op dit gebied spreken vaak over het gebruik van meerdere geografisch gescheiden locaties, zodat de uitval van één enkel datacenter geen onoverkomelijke gevolgen heeft voor de beschikbaarheid van de systemen.

Hoewel niet genoemd in bovenstaande redenen dient bij systemen die beschikbaar moeten zijn ook rekening gehouden te worden met de menselijke factor. Het uitvallen van de stroom in een datacenter gebeurt vaker door een menselijke fout dan doordat de elektriciteit uitvalt van de leverancier. Oplossingen die ook rekening houden met de menselijke factor leveren in veel gevallen een betere beschikbaarheid.

Redenen 1 tot en met 3 zijn vooral van toepassing op bedrijven en organisaties waarbij het falen van systemen vooral invloed heeft op hun eigen positie in de markt en hun imago bij de klanten. Daarbij zal vooral een afweging worden gemaakt tussen de kosten van falen en de kosten die gemaakt kunnen of moeten worden ter voorkoming daarvan.

Belangrijk voor de maatschappij zijn vooral redenen vier en vijf, systemen die in deze categorie vallen hebben grote invloed bij falen op de omgeving en de burger. Voor deze systemen is veelal een groter budget beschikbaar om te zorgen dat ze beschikbaar blijven, waardoor ook speciale oplossingen mogelijk zijn.

2.2 Definitie High Availability

Omdat er onder High Availability meerdere zaken gevat kunnen worden is het belangrijk om een goede definitie van High Availability te bepalen. In paragraaf 2.2.1 worden verschillende woordelijke definities van High Availability beschreven. Paragraaf 2.2.2 beschrijft de verschillende aannames die gedaan worden bij het definiëren, ontwerpen en ontwikkelen van een High Availability oplossing. Paragraaf 2.2.3 geeft een overzicht van beschikbaarheidcijfers waarmee de oplossingen vaak worden gekwantificeerd en in 2.2.4 worden deze cijfers aan beschikbaarheidniveaus gekoppeld.

2.2.1 Definitie

De term High Availability kan in het Nederlands vertaald worden met hoge beschikbaarheid. De term (hoge) beschikbaarheid kan verschillende betekenissen hebben. Een aantal definities is hieronder opgenomen:

- Beschikbaarheid = *“continue toegang tot applicaties, met een voorspelbaar service-level: End-to-End”* Muijzer [3].
- *“In information technology, High Availability refers to a system or component that is continuously operational for a desirably long length of time. Availability can be measured relative to "100% operational" or "never failing." A widely-held but difficult-to-achieve standard of availability for a system or product is known as "five 9s" (99.999 percent) availability.”* WhatIs Definitie [15].
- *“High Availability (HA for short) refers to the availability of resources in a computer system, in the wake of component failures in the system.”* IEEE Task Force on Cluster Computing [14].
- *“A High Availability system is designed, implemented and deployed with sufficient components to satisfy the system's functional requirements but which also has sufficient redundancy in components (hardware, software and procedures) to mask certain defined faults.“* BitPipe HA definition [16].

Belangrijke aspecten in deze definities zijn: het maskeren van optredende fouten op zowel software- als hardwareniveau en het beschikbaar en bruikbaar zijn van applicaties voor de gebruiker. Een systeem dat optredende fouten niet kan tolereren of maskeren zal dan automatisch niet of slechts gedeeltelijk beschikbaar zijn.

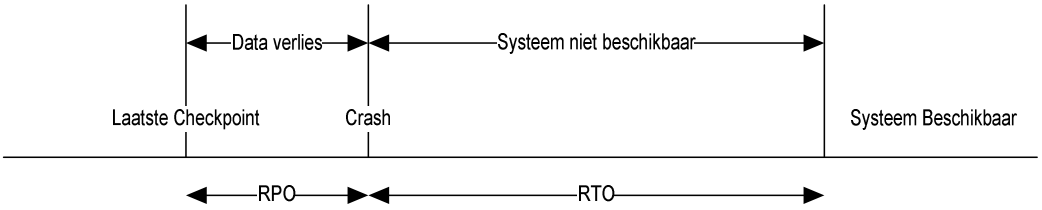
Tolereren is vooral belangrijk voor het doordraaien van het systeem zelf, bij maskeren gaat het om de gevolgen voor gebruikers. Als een systeem een fout kan tolereren, maar een gebruiker dient daardoor opnieuw in te loggen, is de fout niet gemaskeerd voor deze gebruiker. Afhankelijk van de gebruikte definitie kan een systeem dan niet voldoen aan de eis van beschikbaarheid. Daarbij is het belangrijk om op te merken dat de beschikbaarheid altijd gedefinieerd en gemeten moet worden vanuit het oogpunt van de gebruiker, omdat de beschikbaarheid van een systeem slechts interessant is als het gebruikt kan worden. Een gebruiker kan zowel een persoon als een apparaat zijn, dat afhankelijk is van het systeem om zijn taak goed uit te kunnen voeren.

Als het systeem zelf beschikbaar is, maar de gebruiker kan het niet gebruiken door het uitvallen van een netwerkverbinding, dan is het systeem vanuit het oogpunt van de gebruiker dus niet beschikbaar. Een ander criterium waarmee rekening moet worden gehouden, is de geboden response tijd. Als deze niet binnen een bepaalde marge ligt, is het systeem misschien wel beschikbaar, maar niet (goed) bruikbaar.

Een systeem kan zich in drie verschillende toestanden bevinden met betrekking tot de beschikbaarheid van het systeem voor gebruikers. Dit zijn de toestanden *niet beschikbaar*, *gedeeltelijk beschikbaar* en *volledig beschikbaar*. Een systeem dat een hoge beschikbaarheid heeft, kan zich in alle drie de toestanden bevinden. De term hoog wordt gebruikt om aan te geven dat het systeem zich bijna nooit, en als het gebeurt zo kort mogelijk, in de niet of gedeeltelijk beschikbare toestand bevindt. Bij een systeem met hoge beschikbaarheid zijn meestal speciale maatregelen genomen om ervoor te zorgen dat bepaalde fouten geen gevolgen hebben voor de beschikbaarheid. Er wordt ook wel gesproken over fouttolerante systemen.

De term High Availability wordt in veel gevallen aan een systeem gekoppeld als het meer dan 99% van de tijd beschikbaar is voor de gebruiker. Een overzicht van de berekening van deze percentages en andere formules staat in paragraaf 2.2.3. Voor 100% beschikbaarheid worden de termen Continuous Availability (CA) ofwel Disaster Recovery (DR) gehanteerd. Het verschil tussen High Availability en Continuous Availability is het kunnen maskeren van ongeplande en geplande downtime [18]. Beide kunnen ongeplande downtime, bijvoorbeeld door falende componenten, maskeren door hun fouttolerantie. Bij een High Availability systeem is het in sommige gevallen nog acceptabel en/of noodzakelijk om het systeem offline te halen om onderhoud te kunnen plegen. Deze offlineperiode kan slechts een beperkte duur hebben, omdat anders de gewenste of geëiste beschikbaarheid in het gedrang komt. Een CA-systeem kan zelfs deze geplande downtime maskeren en de gebruiker altijd een volledige beschikbaarheid van de applicatie bieden.

Bij Continuous Availability is het noodzakelijk dat het systeem blijft functioneren bij een complete locatie crash, zonder dat data verloren gaat of het systeem niet beschikbaar is voor een bepaalde tijd. Bij Disaster Recovery wordt gesproken over de Recovery Time Objective (RTO) en Recovery Point Objective (RPO) [112], die beide staan weergegeven in *Figuur 1: Recovery Time en Point Objectives*. De RTO bepaalt in welke tijd het systeem weer beschikbaar is na een ramp, de RPO bepaalt hoeveel data er verloren mag gaan. Een RTO van 24 uur betekent dat het systeem binnen 24 uur weer online moet zijn, een RPO van 4 uur betekent dat dataverlies wordt beperkt tot 4 uur voor het moment van de crash. Een DR-oplossing met een RTO = 0 uur (gelijke overschakeling naar secundaire locatie) en een RPO = 0 uur (geen dataverlies doordat checkpoint altijd gelijk is aan het crash moment) vallen onder de CA-systemen. Een DR-systeem kan dus afhankelijk van de gestelde RTO en RPO en toegepaste technologie ingedeeld worden tussen een hoog beschikbaar en een continu beschikbaar systeem.



Figuur 1: Recovery Time en Point Objectives

2.2.2 Failure Assumptions

Bij het ontwerpen van een High Availability oplossing voor een bepaald systeem is het belangrijk om te weten welke failure assumptions zijn gebruikt. Een failure assumption is een aanname die gedaan is met betrekking tot het optreden van het aantal fouten en het soort fouten tijdens het ontwerpen van het systeem. Bij een “single failure assumption” is er sprake van de aanname dat er slechts één fout tegelijkertijd optreedt. Slechts een beperkt aantal ontwerpen houden rekening met het kunnen optreden van meerdere falende componenten tegelijkertijd, de zogeheten “multiple failure assumption”. Systemen die met deze aannames zijn ontworpen kunnen meer fouten tolereren, voordat de beschikbaarheid negatief wordt beïnvloed, dan bij “single failure assumption” systemen.

Een RAID1 diskarray, waarbij een tweede schijf een complete kopie is van de eerste, is een systeem ontworpen met een single failure assumption. Eén falende schijf kan door het array getolereerd worden, omdat dan van de tweede schijf gelezen kan worden. Als beide schijven kapot gaan dan zal het systeem alsnog uitvallen. Hetzelfde principe geldt voor een actief/passief cluster systeem, waarbij er van wordt uitgegaan dat het passieve systeem niet uitvalt als het actieve systeem uitvalt.

Op moment dat een uitval van een component de uitval van een volledig systeem veroorzaakt spreken we van een Single Point of Failure (SPOF). Het optreden van twee fouten tegelijkertijd leidt bij de meeste “single failure assumption” systemen alsnog tot uitval van het volledige systeem. Bij de eerder genoemde diskarray treedt deze als SPOF op als deze volledig uitvalt en de enige storagearray is binnen het systeem. Het repareren van een opgetreden fout is bij een single failure assumption systeem dus belangrijk, omdat een tweede optredende fout in bijna alle gevallen uitval van het volledige systeem betekent.

Door middel van statistiek en kansberekening kan worden bepaald hoe groot het risico van het optreden van meerdere fouten is en of dat de investering waard is ten opzichte van de kosten die gemaakt worden als het systeem niet meer functioneert. Broadwell [80] beschrijft een techniek voor het voorspellen van component failures met een Bayes classificatie algoritme. Kuball [82] gebruikt Bayes voor het voorspellen van systeem failures op basis van component failures.

2.2.3 Beschikbaarheidcijfers

Bij High Availability wordt binnen de industrie vaak gesproken over het aantal *"nines"* van de oplossing. Het aantal opvolgende negens in het beschikbaarheidcijfer bepaalt hoe beschikbaar een oplossing zal zijn of is geweest. Over het algemeen worden beschikbaarheidcijfers aangegeven als een percentage dat het systeem beschikbaar was van de totale tijd. Om het aantal uren niet beschikbaar zijn uit te rekenen bij een bepaald beschikbaarheidcijfer wordt dan *Formule 1 – Niet beschikbare uren* gebruikt.

$$NietBeschikbareUren = AantalUrenInPeriode(1 - Beschikbaarheidcijfer)$$

Formule 1 – Niet beschikbare uren

Een beschikbaarheidcijfer van 99,9% (0.999), ofwel drie "nines", over een heel jaar (365 dagen maal 24 uur) geeft dan 8,75 uur niet beschikbaar in een jaar. Een overzicht van veel gebruikte cijfers staat in *Tabel 1 - Beschikbaarheidcijfers*, gebaseerd op een 24 uurs beschikbaarheid per dag.

Tabel 1 - Beschikbaarheidcijfers

Beschikbaarheidcijfer in %	Downtime in %	Downtime per jaar	Downtime per maand	Downtime per week
90	10	36,5 dagen	3 dagen	16 uur
95	5	18,25 dagen	1,5 dagen	8 uur
98	2	7,3 dagen	14,6 uur	3,36 uur
99	1	3,65 dagen	7,3 uur	1,68 uur
99,9	0,1	8,75 uur	43,76 min	10,10 min
99,99	0,01	52,56 min	4,3 min	1 min
99,999	0,001	5,25 min	26 sec	6 sec
99,9999	0,0001	31,5 sec	2,6 sec	0,6 sec
100	0	Geen	Geen	Geen

Deze beschikbaarheidcijfers kunnen vanuit twee oogpunten bezien worden. Aan de ene kant de geleverde beschikbaarheid over een bepaalde periode, aan de andere kant de gewenste of geëiste beschikbaarheid over een bepaalde periode. Door Won Kim wordt in een artikel uit 1984 [4] de beschikbaarheid van een database gespecificeerd als de ratio tussen de tijd dat eind gebruikers en applicaties de database konden gebruiken en de tijd dat eind gebruikers en applicaties de database wilden gebruiken. In formule:

$$Beschikbaarheidsratio = \frac{TijdBeschikbaarheid}{TijdGewensteBeschikbaarheid}$$

Formule 2 – Beschikbaarheidratio

Op een werkdag van 8 uur met 8 uur gewenste beschikbaarheid en slechts 6 uur beschikbaar zijn van de database, geeft dit een ratio van 0.75 over een periode van 8 uur. Een ratio van 1.00 geeft een continue of volledige beschikbaarheid over de berekende periode.

In Formule 3 - Beschikbaarheidpercentage wordt door Otey [6] een variant op de formule van Kim gebruikt, die tot hetzelfde cijfer leidt, maar dan uitgedrukt in procenten. Uit het voorbeeld: (8-2)/8 = 75%

$$Beschikbaarheidspercentage = \frac{(TotaleVerstrekenTijd - \sum Downtijd)}{TotaleVerstrekenTijd}$$

Formule 3 - Beschikbaarheidpercentage

Fabrikanten van producten geven de beschikbaarheid van hun oplossing vaak aan met *Formule 4 – Beschikbaarheid MTBF en MTTR*, zoals ook door Muijzer [3] gebruikt wordt.

$$Beschikbaarheid = \frac{MTBF}{(MTBF+MTTR)} * 100\%$$

Met
MTBF = Mean Time Between Failure, de gemiddelde tijd totdat een failure optreedt
MTTR = Mean Time To Repair, de gemiddelde tijd die een reparatie van de failure kost
Formule 4 – Beschikbaarheid MTBF en MTTR

Deze formule geeft de verhouding weer tussen de gemiddelde tijd dat een component of systeem kapot gaat (MTBF) en de tijd die het kost om het systeem te herstellen (MTTR). Met deze formule kan vooraf een schatting worden gegeven van de beschikbaarheid van een component. Uit de formule wordt duidelijk dat als de MTBF stijgt, de tijd die nodig is voor reparatie minder belangrijk wordt. Als de reparatietijd afneemt en nul nadert, dan nadert de beschikbaarheid honderd procent. Het verbeteren van de beschikbaarheid kan dus door de componenten betrouwbaarder te maken of te zorgen dat de reparatie minder tijd kost.

Hoewel de betrouwbaarheid van een component vaak wordt uitgedrukt in MTBF is het niet verstandig om hier alleen op af te gaan. Een MTBF van 500.000 uur voor een harde schijf is geen raar getal, maar dat wil niet zeggen dat één harde schijf 500.000 uur, ofwel 57 jaar, mee zal gaan. MTBF zegt namelijk alleen iets over het gemiddelde van een complete groep van componenten in een bepaalde periode van zijn levensduur. Een uitgebreide uitleg wordt gegeven in Daly [46] of Vargas [81]. Het kiezen van een component met een hogere MTBF wil dus niet garanderen dat het systeem beter beschikbaar wordt, maar kan wel een betere beschikbaarheid opleveren.

Het meten van deze beschikbaarheidcijfers is een lastige taak, omdat het meten over een willekeurige periode geen goed beeld hoeft te geven van de beschikbaarheid van een systeem. De meting moet eigenlijk plaatsvinden over de gehele looptijd van een systeem, de zogenaamde “mission-duration” [4], om een goede uitspraak over de beschikbaarheid van het systeem te kunnen doen. Deze looptijd is voor sommige systemen goed omschreven, zoals bij spaceshuttles waarbij een bepaalde looptijd is vastgesteld van het moment van opstijgen tot het moment van terugkeren. Bedrijfsondersteunende systemen

hebben een niet bepaalde looptijd, deze dienen te blijven werken totdat ze vervangen worden, waarbij de vervangingsdatum niet van te voren vast staat. Veelal wordt daarom een periode van een maand of een jaar gebruikt om de beschikbaarheid te meten.

Een tweede probleem bij meten treedt op als een systeem slechts gedeeltelijk beschikbaar is. Op het moment dat 10% van de gebruikers niet met een systeem kan werken, is het systeem niet volledig beschikbaar. Bij het berekenen van het beschikbaarheidcijfer met bovenstaande formules kan hiermee geen rekening worden gehouden.

Een derde probleem is de invloed van omgevingsfactoren op de beschikbaarheid. Een beschikbaarheidcijfer dat puur over de database gaat, zal niet dalen op het moment dat er geen netwerkverkeer meer mogelijk is. Bij het meten is het daarom belangrijk vanuit het oogpunt van de gebruiker te kijken naar het systeem en niet vanuit de grenzen van het systeem of vanuit het systeem zelf.

2.2.4 Beschikbaarheidniveaus

Om bepaalde technieken en oplossingen te kunnen vergelijken is het nuttig om deze te classificeren naar het beschikbaarheidniveau waarop ze gericht zijn. Door deze classificatie kunnen bepaalde karakteristieken van oplossingen eenvoudig worden overzien.

Aan de onderkant van het spectrum bevinden zich de systemen waarbij geen speciale maatregelen zijn genomen om te zorgen dat ze beschikbaar zijn. Deze systemen vallen uit op het moment dat één van de hardware of software componenten het begeeft of de omgevingsfactoren, zoals stroom en netwerk, niet meer beschikbaar zijn.

Aan de andere kant bevinden zich de fouttolerante systemen, die zelfs beschikbaar blijven bij het uitvallen van bepaalde hardware, complete systemen, software of omgevingsfactoren. Voor een aantal mission-critical applicaties is het belangrijk dat deze zelfs blijven werken als een complete locatie uitvalt door een stroomuitval of een rampscenario. Slechts een beperkt aantal oplossingen en aanbieders neemt ook een totale site crash mee in hun definitie van het hoogste niveau van beschikbaarheid. Is dit scenario niet opgenomen, dan is de oplossing in de meeste gevallen ook niet geschikt voor het gebruik van meerdere locaties.

In de literatuur worden de beschikbaarheidniveaus op verschillende manieren ingedeeld. Roosenboom [1] gebruikt de volgende vier verschillende niveaus:

1. De basisbeschikbaarheid van een systeem;
2. Een systeem met redundante hardware componenten;
3. Meerdere machines, redundantie op systeem niveau;
4. Een geografisch gescheiden systeem of gedistribueerd systeem waarbij zelfs een locatie crash geen gevolgen heeft voor de beschikbaarheid van de applicatie.

Uit het interview met Ilse Media [Paragraaf 5.5] blijkt dat zij dezelfde vier niveaus gebruiken om hun applicaties in te delen. Applicaties met hoge beschikbaarheideisen komen in niveau 4, applicaties in de ontwikkelingsfase beginnen op niveau 1 of 2.

Door Chao [2] worden slechts drie niveaus beschreven waarbij het derde niveau de term fouttolerant krijgt. Een fouttolerant systeem dient volgens deze definitie 365x24x7 te draaien en fouten in zowel hard- en software als omgeving te kunnen tolereren zonder dat dit ten koste gaat van de beschikbaarheid en bruikbaarheid van de applicatie. Als deze fouttolerantie ook het uitvallen van een locatie omvat, dan is er sprake van het hierboven genoemde niveau 4, anders gaat het om niveau 3.

De kwaliteit van standaard hardware is door de jaren heen steeds verder gestegen, waarmee de basisbeschikbaarheid van een systeem, door het verhogen van de Mean Time Between Failure (MTBF), ook toeneemt, zoals besproken in paragraaf 2.2. De componenten die veelal als eerste kapot gaan, zijn hardeschijven, ventilatoren en voedingen. Deze componenten worden als eerste redundant uitgevoerd bij een systeem dat in niveau 2 past. Verdere redundantie kan worden bereikt met dubbele processoren, geheugen en netwerkkaarten. Om te zorgen dat kapotte componenten niet alsnog tot downtime leiden tijdens de vervanging ervan, kan gebruik worden gemaakt van “hot-swappable” componenten. Componenten die “hot-swappable” zijn, kunnen terwijl het systeem online is en beschikbaar is, vervangen worden. Hierdoor wordt de Mean Time To Repair (MTTR) sterk gereduceerd. Bij het gebruik van “cold-swappable” componenten dient het systeem uit te worden gezet, voordat het component vervangen kan worden. Net zoals bij het gebruik van het basissysteem leidt dit dan tot downtime, maar wel met een belangrijk voordeel. Bij het gebruik van “cold-swappable” redundantie kan het moment van de downtime zelf worden gekozen door de beheerder in plaats van dat dat moment afhankelijk is van de uitval van het component bij een basissysteem.

Hoewel hardware redundantie de beschikbaarheid kan verhogen, levert het geen oplossing voor het crashen van het besturingssysteem, database of applicaties. Om hiervoor een oplossing te bieden wordt gebruik gemaakt van High Availability oplossingen die zich op niveau 3 en 4 bevinden. Door het gebruik van meerdere systemen wordt het mogelijk om de applicatie vanaf een ander systeem uit te laten voeren als een van de systemen niet meer correct functioneert door soft- of hardware problemen. In dit project zal dan ook worden ingegegaan op deze twee niveaus, aan het uitbreiden van hardware beschikbaarheid door redundantie en de basisbeschikbaarheid zal verder geen aandacht worden besteedt.

2.3 High Availability Aspecten

Een High Availability oplossing die zich alleen richt op de beschikbaarheid van de applicatie zal niet succesvol zijn op een aantal andere gebieden en daardoor nauwelijks worden toegepast. Bij deze andere gebieden gaat het om load balancing, betrouwbaarheid, onderhoud en beheer, schaalbaarheid, data consistentie, back-ups en de performance van de oplossing. De volgende paragrafen beschrijven de noodzaak per onderdeel om tot een goede High Availability oplossing te komen.

2.3.1 Load Balancing

Load balancing wordt binnen een High Availability oplossing toegepast voor twee doelen. De belangrijkste taak van load balancing is het verdelen van de belasting over de verschillende systemen. Zonder deze verdeling komt alle belasting op één systeem, waardoor de overige systemen geen bijdrage kunnen leveren aan de verwerking en het voor de gebruiker een traag of niet goed werkend systeem oplevert. Het tweede doel dat met load balancing kan worden afgehandeld is het versturen van de inkomende connecties naar actieve systemen. Met een intelligente load-balancer is het mogelijk om systemen die niet correct functioneren geen connecties meer te geven. De load balancer maskeert op die manier het uitvallen van een systeem voor de gebruikers.

2.3.2 Reliability

Een High Availability systeem moet niet alleen beschikbaar zijn, maar ook betrouwbaar. Het systeem dient goed te werken en gegevens betrouwbaar op te slaan. Een systeem dat beschikbaar is, maar niet betrouwbaar functioneert, zal door gebruikers worden vermeden, omdat geen zekerheid bestaat over het correct functioneren van het systeem. Zo zal een applicatie die altijd beschikbaar is, maar waarbij de data niet altijd correct wordt opgeslagen, een groot probleem opleveren. De betrouwbaarheid is dus even zo zoniet nog belangrijker dan de beschikbaarheid.

2.3.3 Manageability

De Manageability van een oplossing houdt in hoe goed het systeem te onderhouden is. Welke tools zijn beschikbaar om onderhoud te plegen en hoe ingewikkeld zijn deze tools. Kan er gebruik worden gemaakt van grafische applicaties of dient alles vanaf de commandline te worden aangestuurd. Een oplossing die eenvoudig is in installatie en gebruik biedt minder mogelijkheden tot het maken van fouten, waardoor de oplossing veelal beter beschikbaar zal zijn. Een complex systeem levert meer risico op tot het maken van fouten die de beschikbaarheid negatief zullen beïnvloeden.

2.3.4 Scalability

Schaalbaarheid betekent dat door middel van het toevoegen van resources het systeem meer belasting aankan en dus meer transacties kan verwerken in een bepaalde tijd. Een systeem dat goed schaalbaar is, kan zonder problemen worden uitgebreid met nieuwe resources, waarbij de performance bijna lineair meestijgt. Scalability is verbonden met het concept van load balancing. Zonder

load balancing is de oplossing niet schaalbaar, omdat de extra resources niet gebruikt kunnen worden. Een systeem met slechte schaalbaarheid zal, indien het systeem vol belast wordt, niet uitgebreid kunnen worden en zal op dat moment slechter beschikbaar worden of niet meer correct kunnen functioneren.

2.3.5 Consistency

Consistentie van data is belangrijk voor de applicaties die de data gebruiken. Als data op meerdere systemen moet worden opgeslagen voor redundantie moet ervoor worden gewaakt dat op alle systemen de data consistent is. Bepaalde technieken brengen tijdelijke inconsistentie van data met zich mee waarmee rekening moet worden gehouden in de applicatie.

Consistentie is vooral belangrijk op het moment van een fail-over of take-over van servers. De transacties die op het master systeem zijn uitgevoerd maar nog niet op de slave zijn verwerkt, gaan verloren tijdens een fail-over, waardoor dataverlies optreedt. Het is afhankelijk van de opgestelde High Availability definitie of dit acceptabel is of niet.

2.3.6 Back-ups

Bij High Availability oplossingen zijn back-ups het laatste redmiddel als de overige technieken het hebben laten afweten. Door middel van een back-up kan de data worden hersteld die op het moment dat de back-up gemaakt werd beschikbaar was. Back-ups worden veelal toegepast voor dataherstel, nadat een gebruiker door een fout data heeft gewijzigd of verwijderd. Gebruikersfouten zijn lastig te voorkomen en daarmee moet rekening worden gehouden in een HA oplossing. Een goede HA oplossing heeft bijna nooit back-ups nodig voor het herstel na uitval, maar het uitvallen van de volledige oplossing en gebruikersfouten maken het gebruik van back-ups niet overbodig.

2.3.7 Performance

Een systeem dat hoog beschikbaar is, moet ook een redelijke performance neerzetten. In sommige gevallen mag de beschikbaarheid een deel van de performance kosten, omdat de bescherming van data en applicatie tegen uitval belangrijker zijn dan het aantal te verwerken transacties. De geleverde performance is echter wel van belang, een systeem dat niet voldoende prestaties levert, zal niet worden toegepast hoe goed het ook beschikbaar is.

Een afweging die bij elk systeem gemaakt moet worden is de benodigde performance ten opzicht van de consistentie van de data en eventueel dataverlies bij uitval van een component. Een systeem dat altijd consistente data moet hebben zal door de communicatieoverhead altijd slechter presteren dan een systeem dat door asynchroniteit geen communicatieoverhead heeft, maar wel het risico loopt van dataverlies. Een systeem dat zijn data lokaal op kan halen is sneller dan een systeem dat de data via het netwerk moet ophalen. Hierdoor zijn de responsetijden van een enkel systeem veelal beter dan van een clustersysteem. Wel kan een clustersysteem meer aanvragen verwerken dan een enkel systeem. Bij performance metingen is het dus belangrijk om de goede meeteenheden te gebruiken als oplossingen vergeleken worden.

2.4 Organisaties

Voor organisaties die een bepaald systeem hoog beschikbaar willen maken, zijn een aantal factoren van belang. Alleen het kopen van een technologie of product zal vaak niet het gewenste effect brengen. In paragraaf 2.4.1 worden deze factoren besproken. De problemen die bij het maken van een goede en noodzakelijke kosten/baten analyse komen kijken worden beschreven in paragraaf 2.4.2. Service Level Agreements waarmee afspraken worden vastgelegd tussen een leverancier en een klant, zowel binnen organisaties als tussen organisaties, over de te behalen service, leveren niet de garantie dat de gewenste beschikbaarheid ook behaald wordt. In paragraaf 2.4.3 wordt op de SLA problematiek ingegaan.

2.4.1 Invloedfactoren

Een High Availability oplossing bestaat niet alleen uit een bepaalde technologie, maar ook uit de werknemers die het systeem moeten installeren en onderhouden en de verschillende processen en procedures die gedefinieerd moeten worden. Er wordt ook wel gesproken over de 3 P's, People, Processes and Products [3], ofwel People, Process en Technology [6]. Volgens Muijzer [3] heeft een gebruikt product of technologie slechts voor 20% invloed op de ongeplande downtime. De overige 80% komt door verkeerde of niet goed afgesproken procedures of door menselijk handelen. Dat betekent dat niet alleen over het te gebruiken product moet worden nagedacht, maar veel meer nog over de procedures en werknemers die het systeem in de lucht moeten houden.

Als slechts één persoon binnen een organisatie ervaring heeft met een bepaalde oplossing en de handelingen zijn verder niet goed gedocumenteerd, dan wordt daarmee een single point of failure gecreëerd. Als deze persoon ziek wordt of op vakantie is en het systeem heeft een probleem, zal het niet meer of na grote vertraging hersteld kunnen worden. Daarmee wordt de beschikbaarheid van het systeem direct afhankelijk van de beschikbaarheid van de werknemer. Omdat één werknemer geen 99 of 100% beschikbaarheid haalt, is dat geen verstandige situatie.

Afspraken over de back-up en restore procedures en over het onderhoud van de oplossing zijn zeer belangrijk. Deze procedures dienen ook getest en bijgehouden te worden. De grootste risico's voor een hoog beschikbaarheidsysteem zijn menselijke fouten, zoals het verwijderen van bestanden of verkeerd wijzigen van een database. Hoe beter de processen beschreven zijn, hoe minder belangrijk de vaardigheden van de personen zijn en hoe beperkter het optreden van deze fouten wordt. Hoe beter de vaardigheden, hoe minder er beschreven hoeft te worden.

Het gebruik van een simpele technologie maakt het mogelijk om meer mensen op te leiden en kennis op te laten doen met het product. Dit kan dan leiden tot een betere beschikbaarheid van het product, doordat er minder fouten worden gemaakt en problemen beter kunnen worden opgelost. Een goede beschikbaarheid wordt dus slechts beperkt door het product bepaald en voor het

grootste deel door training, afspraken van procedures en het goed doortesten van deze procedures en het product [6].

2.4.2 Kosten versus Baten

Bij het investeren in een High Availability oplossing is het belangrijk om een goede kosten/baten analyse te maken. Het investeren in de beschikbaarheid van een systeem dat geen problemen veroorzaakt als het een paar dagen niet beschikbaar is, is geen verstandige investering. Hier zijn de kosten van het beschikbaar maken van het systeem veel hoger dan de kosten die geleden worden als het systeem niet beschikbaar is. Voor een applicatieserver die een dag offline mag zijn is het veel goedkoper en simpeler om een reserve systeem in voorraad te hebben, dan om speciale maatregelen voor dat systeem te treffen met betrekking tot hardware redundantie of een geclusterde HA oplossing.

Organisaties die afhankelijk zijn van hun ICT systemen om producten of diensten te kunnen verkopen, leiden meestal meer kosten door downtime dan die gemaakt worden om deze downtime gedeeltelijk te voorkomen. Voor een webwinkel als Amazon kost het niet beschikbaar zijn van hun website klanten, en daarmee inkomsten. Het is niet zo eenvoudig om te kunnen berekenen hoeveel kosten het niet beschikbaar zijn van een systeem met zich meebrengt. Uit een survey van Contingency Planning Research uit 2000 zou dit voor Amazon een bedrag van $180.000 bedragen voor elk uur downtime.[23][19].

Bij het berekenen van de kosten van downtime moet niet alleen rekening worden gehouden met de gederfde inkomsten door gemiste klanten, maar ook met de kosten van de werknemers en eventueel verloren productie. De invloed van een downtime is vaak niet volledig vast te stellen, waardoor veelal met schattingen wordt gewerkt. Een simpele formule die is voorgesteld door Patterson [19] wordt hieronder weergegeven:

Verwachte gemiddelde kosten van 1 uur downtime =
    Werknemerskosten per uur * fractie van werknemers getroffen door downtime +
    Gemiddeld inkomen per uur * fractie inkomen getroffen door de downtime.

Deze formule geeft een beeld van de mogelijke kosten die het niet beschikbaar zijn van een systeem oplevert. Door middel van de fracties kan een gedeeltelijk onbeschikbaar systeem worden meegenomen in de formule. Als bijvoorbeeld het systeem niet beschikbaar is voor 10% van de werknemers, dan kan daarmee gerekend worden. Voor een productiebedrijf, waarbij de productie afhankelijk is van de systemen, is deze formule minder geschikt, aangezien de kosten van de verminderde productie niet meegenomen kunnen worden. Kosten van reparatie en seizoensinvloeden worden niet meegenomen in de formule, maar zijn relatief klein ten opzichte van de meegenomen kosten. Deze formule geeft een eerste inzicht in de kosten die gemaakt worden bij uitval, een uitgebreide analyse kan dan meer uitsluitsel geven [19]. Om te bepalen wat dan geïnvesteerd kan worden moet ook worden bepaald hoe groot het risico is van de uitval. Indien dat risico beperkt is, zoals bij uitval van een datacenter, is het in veel gevallen een te dure investering om deze dubbel uit te voeren, tenzij de kosten van uitval hoog genoeg zijn om dit te verantwoorden.

2.4.3 Service Level Agreements

Service Level Agreements (SLA) worden gebruikt om afspraken te maken over welke service de klant mag verwachten en hoeveel de klant daarvoor moet betalen. In een SLA kan bijvoorbeeld worden afgesproken dat een leverancier binnen vier uur een apparaat komt vervangen als het geleverde exemplaar stuk is. Belangrijke zaken die in een SLA worden afgesproken zijn de tijd die een reparatie mag duren, de beschikbaarheid van de geleverde systemen of diensten en de consequenties als deze beschikbaarheid niet gehaald wordt. Hoewel een SLA kan helpen om de beschikbaarheid te verhogen, doordat er betere afspraken zijn gemaakt tussen de leverancier en klant, heeft een SLA op zichzelf geen invloed op de beschikbaarheid van een systeem. Een SLA zijn slechts afspraken op papier.

Belangrijk bij het opstellen van een SLA is het goed definiëren van de afspraken en te zorgen dat deze afspraken meetbaar en testbaar zijn. Zo is een best-effort clausule niet meetbaar of testbaar, de leverancier kan altijd zeggen dat deze zijn best heeft gedaan. In een SLA dienen resultaatverplichtingen te worden opgenomen en geen inspanningsverplichtingen. Beschikbaarheidcijfers dienen duidelijk gedefinieerd te zijn over de periode waarover de beschikbaarheid gaat en over de onderdelen waar de beschikbaarheid voor geldt. Een beschikbaarheid van 99% over de database, terwijl het netwerk maar 90% behaalt, geeft een klant slechts een beschikbaarheid van ten hoogste 90%. Zo heeft Ilse Media de periode waarover gemeten wordt, één maand, vast laten leggen in haar SLA, samen met de beschikbaarheidcijfers [Paragraaf 5.5].

Vaak worden boeteclausules opgesteld als de leverancier niet aan de beschikbaarheideis voldoet. Deze boeteclausules vergoeden echter meestal alleen de direct gemaakte kosten voor de gevraagde service, en niet de indirecte kosten van het niet beschikbaar zijn, zoals verloren inkomsten en kosten van werknemers. Het is voor de klant veel belangrijker dat het systeem beschikbaar is, dan dat deze de boete kan innen. Meer informatie over de problemen van SLA’s kan worden gevonden in Scheffel [17].

Hoewel een SLA dus op zichzelf geen betere beschikbaarheid biedt, is het afspreken van een SLA toch belangrijk om de verwachtingen en resultaten op het gebied van beschikbaarheid af te spreken. Zo weten beide partijen wat ze mogen verwachten en wat ze moeten leveren en ontstaat een betere omgeving om de beschikbaarheid te waarborgen.

3. TECHNIEKEN

Om een systeem hoog beschikbaar te maken en te houden, zijn verschillende technieken noodzakelijk of mogelijk. Dit hoofdstuk beschrijft de technieken die het hoog beschikbaar maken van een databasesysteem of data in dit systeem mogelijk maakt.

Om te zorgen dat componenten in een systeem niet als Single Point Of Failure optreden, is het redundant uitvoeren van deze componenten noodzakelijk. Om data van redundantie te voorzien is replicatie van data benodigd. Paragraaf 3.1 beschrijft de redundantie, 3.2 beschrijft de replicatie in een HA systeem.

Om systemen samen te laten werken kan gebruik worden gemaakt van clustering. Clustering kan op meerdere niveau's worden toegepast, namelijk op database, middleware en operating system niveau. Deze worden respectievelijk in paragraaf 3.3, 3.4 en 3.5 behandeld. Tenslotte wordt in paragraaf 3.6 een klein overzicht gegeven van mogelijke toekomstige technieken en trends die een rol kunnen gaan spelen in het beschikbaar maken en houden van systemen.

3.1 Redundantie

De basis van een High Availability oplossing is het zorgen voor redundantie van componenten die een single point of failure vormen in een systeem. Een single point of failure is een component dat door zijn uitval ervoor zorgt dat het volledige systeem niet meer kan functioneren. Indien de voeding van een systeem kapot gaat, dan valt het hele systeem uit doordat het geen stroom meer kan ontvangen van de voeding. Door een voeding redundant uit te voeren, heeft het uitvallen van een van de voedingen geen gevolgen voor het systeem. Redundantie kan gaan om hardware componenten in een enkele server, maar ook over meerdere servers in een clusteroplossing. Deze redundantie kan op meerdere niveaus worden toegepast.

Op hardware niveau kunnen meerdere componenten worden gebruikt voor redundantie, zoals RAID arrays, voedingen, processors, geheugen en netwerkkaarten. Systemen waarbij alles dubbel redundant of zelfs triple redundant zijn uitgevoerd, worden meestal verkocht als Fout-Tolerante (FT) systemen, zoals de Stratus/FT servers [21]. Stratus gebruikt een lock-step techniek voor haar hardware. Met deze techniek worden de taken op beide systemen tegelijkertijd uitgevoerd en wordt een niet goed functionerend component uitgeschakeld als problemen optreden. Een variant op deze techniek is het gebruiken van twee aparte systemen, die niet zoals bij Stratus geïntegreerd zijn, maar die door middel van software in een lock-step draaien. Een voorbeeld hiervan is de Marathon FT/virtual server [22].

Een groot probleem van deze oplossingen is dat zij slechts een oplossing bieden voor hardwarematige fouten. Als een fout optreedt in de gebruikte software, waardoor het systeem stopt met functioneren, is het volledige systeem niet meer beschikbaar. Doordat hardware tegenwoordig steeds betrouwbaarder is geworden, zijn deze systemen nog slechts interessant voor omgevingen

waarin de uitval van een enkel systeem grote schade oplevert en een cluster geen oplossing kan bieden, bijvoorbeeld bij gespecialiseerde systemen in ziekenhuizen, vliegtuigen of het leger. Combinaties met clusteroplossingen zijn ook mogelijk, maar door de hoge kosten van deze FT systemen is het vaak niet rendabel om de verschillende cluster componenten ook volledig fouttolerant uit te voeren. De clusteroplossing biedt deze functionaliteit al aan, waardoor fouttolerante systemen in veel situaties overkill zijn.

3.2 Replicatie

Voor een High Availability oplossing die gebruik maakt van meerdere systemen is het nodig om de data op de verschillende systemen op te slaan. Als data slechts op een plek wordt opgeslagen, dan valt het systeem uit als de data niet meer beschikbaar is. Om te voorkomen dat de data een single point of failure vormt, dient deze data over meerdere systemen te worden verspreid. Daarbij is redundantie van de data wel een vereiste. Zonder redundantie is verspreiding van data een groter risico dan het centraal op een plek opslaan. De kans dat één van vijf systemen kapot gaat is groter dan de kans dat één systeem kapot gaat. Uitval van een enkel systeem waarvan de data niet redundant op een ander systeem is opgeslagen, betekent dan data verlies en het niet beschikbaar zijn van de data voor de applicatie.

Voor het opslaan van data op meerdere systemen kan gekozen worden tussen data replicatie of het gebruiken van een shared-storage systeem. Shared-storage houdt in dat meerdere systemen hetzelfde opslag systeem gebruiken en dus bij dezelfde data kunnen. Het gebruik van shared-storage wordt beschreven in paragraaf 3.3.1. Bij replicatie gebruiken alle systemen hun eigen opslag, waarop een kopie van de data wordt opgeslagen.

Binnen de replicatietechniek kan onderscheid worden gemaakt tussen een synchrone en asynchrone replicatie en tussen de plaats waar schrijfacties op de data mogen plaatsvinden. Het verschil tussen een synchrone en asynchrone replicatie bevindt zich in het moment van de replicatie van de data. Bij een synchrone techniek wordt de data op alle plaatsen weggeschreven voordat de applicatie een bevestiging krijgt van de opslagactie. De data staat op alle systemen bij deze bevestiging en uitval van een van de systemen heeft geen dataverlies tot gevolg van de behandelde transactie. Bij asynchrone replicatie wordt een bevestiging gestuurd als op het systeem dat de actie behandelt de data is weggeschreven en er wordt niet gewacht op de bevestiging van andere systemen. Bij asynchrone replicatie is er de keuze tussen direct wegschrijven naar secundaire systemen of het opsparen van schrijfacties en deze in één batch uitvoeren. Hiermee kan de beschikbare bandbreedte en latency optimaal benut worden.

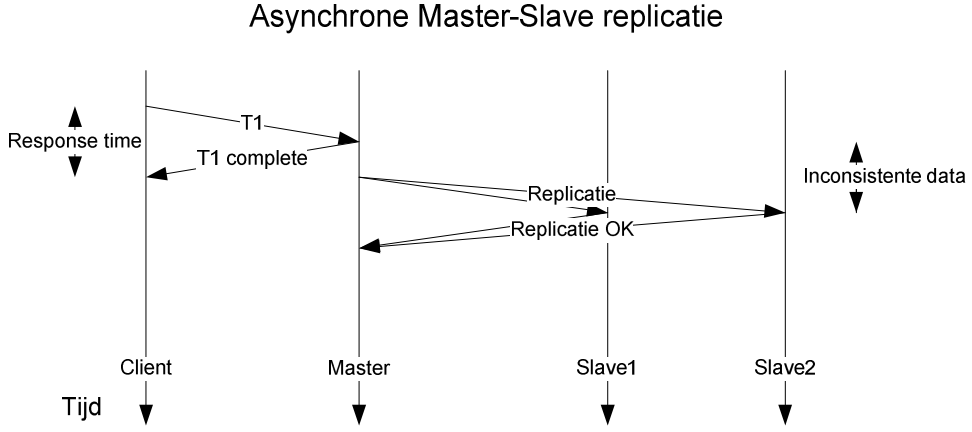
Een tweede onderscheid kan gemaakt worden tussen de plaats waar schrijfacties plaats mogen vinden. Als er maar één systeem is waarop veranderingen mogen plaatsvinden, is er sprake van een Master-Slave systeem. Het systeem dat als Master fungeert, verwerkt dan alle schrijfacties. Afhankelijk van de gebruikte replicatietechniek zijn slaves bruikbaar voor leesacties of slechts als een passief stand-by-systeem. Bij een replicatiesysteem waarop op

elke machine geschreven mag worden, spreken we van een Multi-Master systeem. Dat betekent dat het voor de applicatie niet uitmaakt met welk systeem er wordt gecommuniceerd voor het uitvoeren van zijn taken. Bij een Master-Slave dient de applicatie met schrijfacties altijd het Master systeem te gebruiken.

Figuren 2,3 en 6,7 geven de verschillende combinaties van deze technieken aan, respectievelijk asynchrone Master-Slave, synchrone Master-Slave, asynchrone Multi-Master en synchrone Multi-Master.

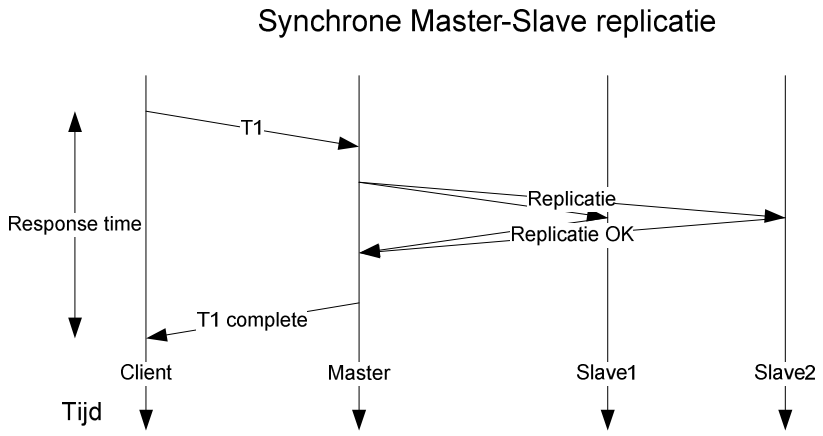
In de figuren 2 en 3 is duidelijk te zien wat het verschil is tussen een synchrone en asynchrone replicatietechniek. Bij het gebruik van de asynchrone methode krijgt de client direct antwoord terug als de transactie op de master verwerkt is, waardoor de response time niet afhankelijk is van de replicatietijd. Dit is bij synchrone replicatie wel het geval, hier krijgt de client pas een bevestiging als de replicatie succesvol is uitgevoerd op één of alle slaves. Voor de synchrone methode is de latency die optreedt bij het repliceren dus zeer belangrijk voor de performance die het systeem kan behalen.

Het grote nadeel van asynchrone replicatie is het niet altijd volledig consistent zijn van de data binnen het systeem. De data op de slaves kan achterlopen op de data die op de master is ingevoerd. Dit probleem heeft de synchrone techniek niet.



Figuur 2: Asynchrone Master-Slave replicatie

De Master-Slave techniek heeft als grote voordeel dat geen transactieconflicten of deadlocks optreden, zoals respectievelijk bij asynchrone of synchrone Multi-Master replicatie. Nadeel is dat de master in dit geval als single point of failure optreedt. Als de master uitvalt, kunnen geen schrijfacties meer op het systeem worden uitgevoerd en zal een van de slaves als nieuwe master benoemd moeten worden om het systeem snel weer in de lucht te brengen.

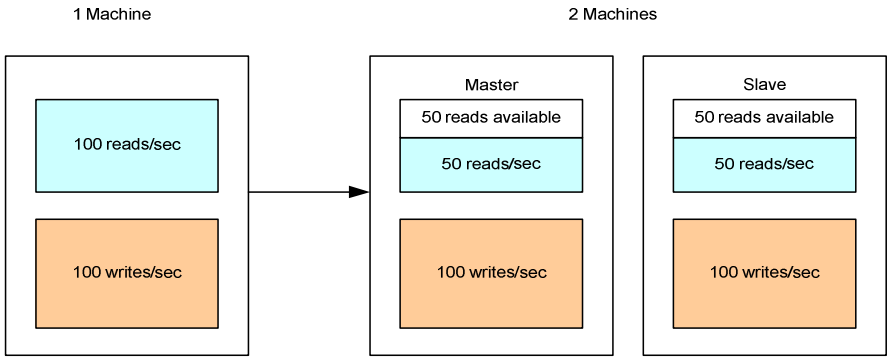


Figuur 3: Synchrone Master-Slave replicatie

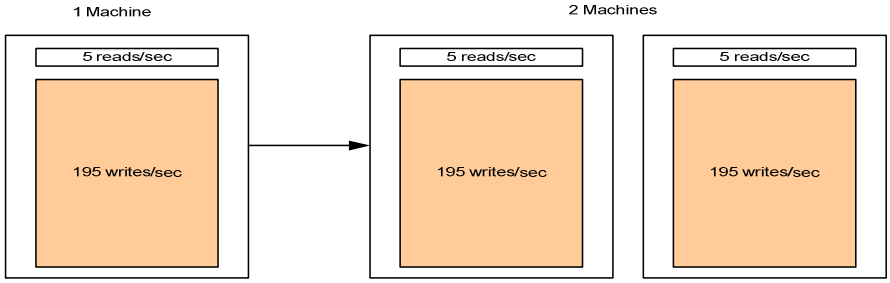
Een tweede probleem is het niet kunnen schalen van de schrijfacties door het bijplaatsen van een extra machine. Doordat alle schrijfacties op de master moeten plaatsvinden is deze machine de bottleneck. Als de master de hoeveelheid schrijfacties niet meer aankan, zal een grotere machine gekocht moeten worden of dient met data partities gewerkt te gaan worden. Bij applicaties die zowel veel schrijfacties als veel leesacties hebben, levert het bijplaatsen van een slave voor leesacties weinig voordeel op. Doordat alle schrijfacties ook op de slave uitgevoerd moeten worden, houdt deze slechts beperkte resources over om mee te lezen. De bandbreedte van de master is vaak beperkend voor het aantal slaves wat aangesloten kan worden.

Figuur 4: Schaalproblemen – Extra capaciteit leesacties bij uitbreiding laat zien dat bij het bijplaatsen van een extra machine extra capacitieit voor leesacties wordt gecreeerd. Aangenomen dat per seconde in totaal 200 acties mogelijk zijn per machine en dat schrijf en leesacties evenveel kosten, heeft een machine een verdeling van 100 schrijf en 100 leesacties per seconde bij het gebruik van een bepaalde applicatie. Het bijplaatsen van een extra systeem levert nu een verdeling van de 100 leesacties op over de twee systemen, terwijl de schrijfacties op beide uitgevoerd moeten worden, zoals zichtbaar in de bovenste helft. Omdat beide systemen ook 200 acties per seconden kunnen uitvoeren, kunnen er 100 leesacties meer worden verwerkt. Elke toegevoegde machine levert 100 extra leesacties. Leesacties zijn dus goed schaalbaar, totdat de bandbreedte van de master het aantal slaves niet meer aankan.

Het probleem wordt zichtbaar in *Figuur 5: Schaalproblemen – Schrijfacties blokkeren leesacties*. Indien het aantal schrijfacties zou stijgen tot 195 per sec, dan zijn er per machine nog maar 5 leesacties mogelijk. Elke slave levert dan slechts een toename van 5 leesacties per seconde. Het systeem is dus niet schaalbaar voor de schrijfacties, een toename van de schrijfacties betekent een afname van de totale performance van het systeem, omdat er steeds minder leesacties mogelijk zijn. Bij dergelijke omvang van de schrijfacties is het uitbreiden met meerdere slaves niet zinvol, en kan beter gebruik worden gemaakt van een multimaster setting met verschillende data partities.

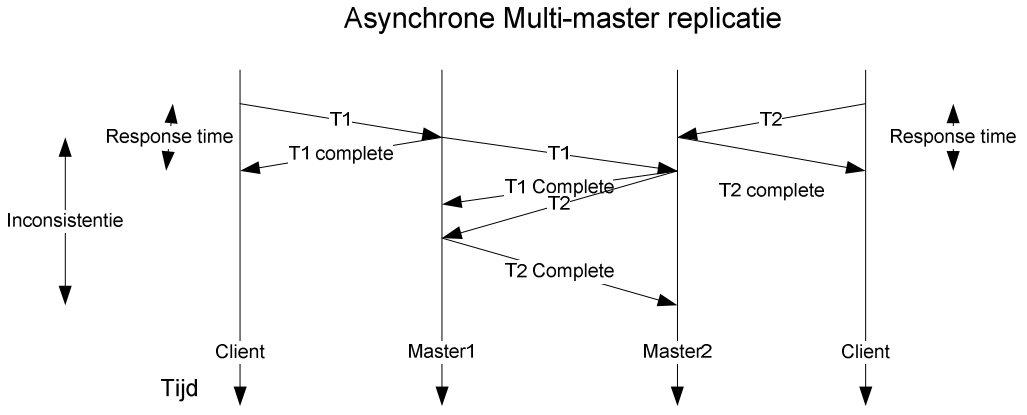


Figuur 4: Schaalproblemen – Extra capaciteit leesacties bij uitbreiding



Figuur 5: Schaalproblemen – Schrijfacties blokkeren leesacties

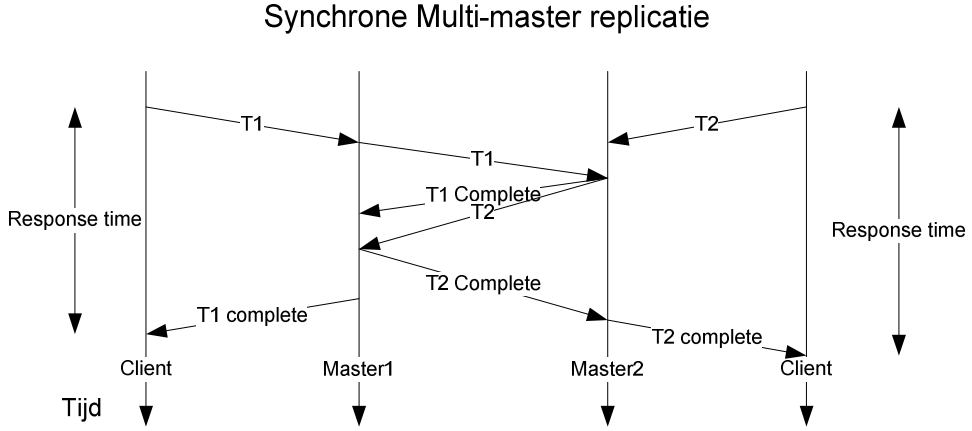
Figuur 6: Asynchrone Multi-master Replicatie en *Figuur 7: Synchrone Multi-Master replicatie* laten de Multi-Master replicatie zien. Belangrijkste verschil met figuren 2 en 3 is het op meerdere plekken kunnen uitvoeren van zowel lees- als schrijfacties. Hier is dat weergegeven door twee verschillende clients die op twee verschillende masters hun transacties uitvoeren. Voor zowel de synchrone als asynchrone techniek geldt hier hetzelfde als bij de Master-Slave configuratie met betrekking tot response time en inconsistentie van data.



Figuur 6: Asynchrone Multi-master Replicatie

Bij een Multi-Master systeem kan onderscheid worden gemaakt tussen een update everywhere (UE) en een primary copy (PC) aanpak. Bij een update

everywhere wordt de transactie op alle systemen tegelijk uitgevoerd (eventueel asynchroon), terwijl bij een primary copy de schrijfactie op één systeem wordt uitgevoerd, waarna deze de wijzigingen doorgeeft aan de andere systemen. Primary copy is het meest gebruikelijk bij asynchrone replicatie, aangezien deze qua implementatie vergelijkbaar is met de Master-Slave asynchrone replicatie. In een Multi-Master setting dienen echter zaken als conflict resolutie te worden opgelost, die in een master slave setting geen rol spelen

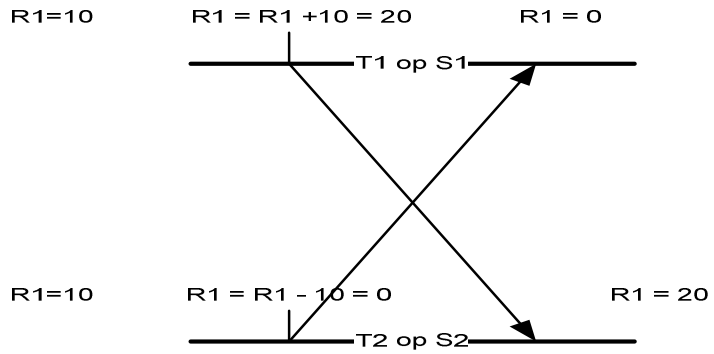


Figuur 7: Synchrone Multi-Master replicatie

Doordat de replicatie asynchroon verloopt, kunnen er update, uniqueness en delete conflicten optreden [50]. Als op 2 systemen tegelijkertijd een transactie wordt uitgevoerd waarbij eenzelfde rij wordt gewijzigd, dan levert dit een update conflict op. Een uniqueness conflict treedt op als op beide systemen eenzelfde attribuut in een rij wordt toegevoegd op een kolom waarin alleen unieke waardes mogen voorkomen. Bijvoorbeeld het toevoegen van de unieke gebruikersnaam piet, voor zowel piet a op systeem a als piet b op systeem b. Bij synchronisatie treedt dan een conflict op tussen beide systemen. Een delete conflict treedt op als het delete statement op het eerste systeem niet op het tweede systeem uitgevoerd kan worden, omdat rijen al verwijderd zijn of waardes uit die rijen al bijgewerkt zijn door een andere transactie.

Een voorbeeld van een update-conflict is het volgende: Bij transactie T1 op systeem S1 in *Figuur 8: Update Conflicten* wordt het banksaldo veranderd van 10 naar 20 van rij R1. Als de transactie T2 op S2 dezelfde rij R1 het banksaldo verandert van 10 naar 0, dan is de database niet meer consistent. Bij asynchrone replicatie zal S1 na de replicatie van S2 het banksaldo op 0 hebben staan, terwijl S2 het banksaldo op 20 heeft staan. Beide saldo's zijn niet correct, het goede saldo zou 10 moeten zijn. Ook al wordt een van beide transacties niet goedgekeurd, dan nog klopt het uiteindelijke saldo niet.

Conflict resolutie in een Multi-Master asynchrone setting is belangrijk om consistente data te krijgen, maar ook ingewikkeld. Om conflicten automatisch op te lossen kunnen een aantal verschillende conflictresolutieregelaars gebruikt worden. De twee meest gebruikte zijn "site priority" en "timestamp priority" [50].



Figuur 8: Update Conflicten

Bij site priority krijgt een van de systemen altijd prioriteit boven andere systemen, bij conflicten wordt één master aangewezen waaraan de andere systemen zich conformeren. Bij timestamp priority krijgt een van de transacties prioriteit op basis van de tijd wanneer de transactie gestart is. Dit kan zowel de laatst gestarte transactie zijn, als de eerste, afhankelijk van de gekozen procedure. Uiteraard dienen alle systemen gesynchroniseerd te worden met dezelfde tijd, anders zal één van de systemen altijd prioriteit krijgen over de andere systemen.

Voor automatische conflictoplossing is meestal domeinspecifieke informatie benodigd, waardoor deze systemen niet veel worden toegepast. In veel gevallen wordt het oplossen van dergelijke problemen echter aan een database beheerder overgelaten, die handmatig deze conflicten moet oplossen. Deze replicatie technologie wordt daarom niet veel gebruikt in database replicatie, maar vooral voor het synchroniseren van bijvoorbeeld agenda's of contactpersonen, omdat de gebruiker in deze gevallen gelijk een conflict kan oplossen indien dit voorkomt. Zowel Oracle als IBM ondersteunen met haar databases Multi-Master asynchrone replicatie.

Bij de synchrone variant treden in plaats van conflicten deadlocks op, omdat beide transacties op elkaars locks moeten wachten. Hierbij is het dus belangrijk een deadlock detectie of preventie algoritme te implementeren die bepaalde transacties kan weigeren of een rollback forceert indien een deadlock is opgetreden. Deze deadlocks hebben grote invloed op de performance van het systeem, doordat het de verwerking van transacties vertraagt of compleet stilzet. Een goede implementatie is dus belangrijk voor de performance en betrouwbaarheid van een synchroon Multi-Master systeem.

Tabel 2 en *Tabel 3* geven een samenvattend overzicht van de verschillende voor en nadelen van de vier genoemde technieken.

Tabel 2: Voor- en nadelen Master-Slave

	Master-Slave	
	Asynchroon	Synchroon
Voordelen	• Geen invloed op performance van het systeem • Latency's geen probleem, lange afstanden mogelijk • Simpelste en meest toegepaste replicatie	• Consistente data op alle systemen • Geen dataverlies bij master crash
Nadelen	• Geen consistente view op beide systemen • Mogelijk dataverlies bij crashes • Niet goed schaalbare schrijfacties • Master is single point of failure • Alleen load balancing voor reads	• Performance issues, langzame response tijden • Niet mogelijk over grote afstanden • Niet goed schaalbare schrijfacties • Master is single point of failure • Alleen load balancing voor reads

Tabel 3: Voor en nadelen Multi-Master

	Multi-Master	
	Asynchroon	Synchroon
Voordelen	• Geen single point of failure • Load balancing voor reads en writes	• Consistente data • Geen single point of failure • Load balancing voor reads en writes
Nadelen	• Conflict oplossing noodzakelijk • Inconsistente data mogelijk • Dataverlies mogelijk	• Deadlocks • Beperkt schaalbaar • Beperkt in overbrugbare afstanden

3.3 Database Clustering

Database clustering is het toepassen van verschillende clustertechnieken om de data uit een database toegankelijk te maken en te houden voor de gebruikers. Bij database clustering zijn drie verschillende gebieden te identificeren, namelijk de opslag van data, het delen van data en het toegankelijk maken van data.

Een architectuur keuze die gemaakt moet worden voor de opslag is de plek waar de data wordt opgeslagen, ofwel gedeelde ofwel ongedeelde opslag. Paragraaf 3.3.1 beschrijft deze architectuur keuze. Het delen van de data binnen het cluster kan door middel van shared-storage of de al beschreven replicatieoplossingen in paragraaf 3.2. Om de data toegankelijk te maken voor de gebruiker van de data zijn er verschillende oplossingen. Belangrijk bij deze toegang is het kunnen balanceren van de belasting over de verschillende nodes, zodat er geen overbelasting op een van de systemen optreedt en het volledige systeem niet meer beschikbaar is of zeer trage response tijden oplevert. Deze oplossingen staan in paragraaf 3.3.2.

3.3.1 Shared-storage VS Shared-nothing

Bij het ontwikkelen van een cluster oplossing is er de keuze tussen een shared-storage of een shared-nothing architectuur. Bij het gebruik van shared-storage is vaak een vorm van een Storage Area Network (SAN) benodigd, waarop alle data wordt opgeslagen [84][85]. Een uitgebreide SAN tutorial staat op Dot Hill [83], in deze paragraaf worden enkele zaken belicht die van belang zijn voor de beschikbaarheid van de data. Een SAN bestaat uit één of meer disk arrays waar via glasvezel of SCSI verschillende machines op aangesloten kunnen worden. Elke machine die op het SAN is aangesloten heeft toegang tot de data, waardoor geen replicatie van data tussen nodes onderling hoeft plaats te vinden. Als een node faalt, dan kunnen de andere nodes nog bij dezelfde data en kunnen de taken van deze node worden overgenomen.

Een groot nadeel van het gebruik van een SAN is het introduceren van een single point of failure. Als de SAN uitvalt, dan werkt het cluster ook niet meer doordat er geen toegang meer is tot de data. Om dit te voorkomen dient de SAN redundant te worden uitgevoerd en moet door mirroring of replicatie de data op beide SAN's bewaard worden. Een groot nadeel van deze oplossing zijn de hoge kosten die gepaard gaan met het dubbel uitvoeren van een SAN oplossing. Wel bieden SAN's veelal snapshot functies aan waarmee meerdere online back-ups gemaakt kunnen worden. De gebruikte database dient dan wel "SAN-aware" te zijn om te zorgen dat dit snapshot ook een consistente data image oplevert. Gebeurt dit niet, dan zal de database zelf een recovery procedure op moeten starten om na het terugzetten van een snapshot in een consistente status te geraken.

Een shared-nothing architectuur geeft aan dat er geen faciliteiten gedeeld worden door de verschillende cluster nodes. Elke node heeft zijn eigen processor, geheugen en opslagruimte. Doordat de nodes niet van elkaars opslagruimte gebruik kunnen maken, is replicatie van de data noodzakelijk om

de data beschikbaar te houden als een node uitvalt. De verschillende replicatietechnieken zijn al beschreven in paragraaf 3.2.
Door de redundantie van de data is meer schijfruimte benodigd dan in de shared-storage variant, waarbij alle data slechts één keer wordt opgeslagen. Een 4-node shared-nothing cluster zal, uitgaande van volledige redundantie (elke node één kopie van de data), vier keer de ruimte voor de opgeslagen data nodig hebben vergeleken met een shared-storage oplossing waarbij slechts één keer de data wordt opgeslagen (exclusief extra redundantie door RAID systemen). Doordat gewone opslagsystemen zoals harde schijven een stuk goedkoper zijn dan een SAN-systeem is het gebruik van extra opslagruimte veelal goedkoper dan het gebruik van een SAN.

Doordat in een shared-storage omgeving de data slechts een keer wordt opgeslagen is een RAID oplossing die voor redundantie van de data zorgt wel een must. Dit is bij een shared-nothing architectuur in principe niet nodig, door de redundantie over meerdere systemen, maar wordt wel vaak toegepast voor extra veiligheid en de voordelen voor het beheer van de systemen. Hierdoor valt een node minder snel uit en is het vervangen van één schijf vrij eenvoudig.

Zowel Oracle's Real Application Clusters (RAC) als Microsoft's SQL Server N-Way Clustering zijn als clusters ontworpen om gebruik te maken van een shared-storage architectuur. Door gebruik te maken van gedeelde opslag hoeft in deze systemen geen replicatieoplossing gebruikt te worden om vanaf elke node bij alle data te kunnen. MySQL Cluster is een voorbeeld van een shared-nothing architectuur. Informatie over deze producten staat in hoofdstuk 4.

3.3.2 Cluster toegang

Een High Availability systeem dat gebruik maakt van een cluster techniek, kan op meerdere manieren de toegang tot de database mogelijk maken. Het balanceren van verbindingen is nodig als gebruikt wordt gemaakt van twee of meer actieve systemen. Bij het niet balanceren van de verbindingen, kan een te hoge belasting ontstaan op één van de systemen. Alle gebruikers die op dit systeem binnenkomen, kunnen daardoor niet worden afgehandeld, waardoor het volledige systeem niet beschikbaar is. Alle overige machines kunnen dan nog prima functioneren, maar de gebruiker kan dan deze machines niet bereiken. Als de verbindingen gebalanceerd moeten worden over de verschillende nodes, is het belangrijk dat deze verbindingen via een punt binnenkomen.

Om toegang te krijgen via één punt van het cluster kan gebruik worden gemaakt van een aparte hard- of software load-balancer of het gebruik van een virtueel IP-adres. Een load-balancer kan de binnenkomende verbindingen delen over de verschillende nodes en kan door middel van monitoring zien welke nodes actief zijn en dus verbindingen kunnen accepteren. Een nadeel van een load-balancer is het introduceren van een single point of failure. Als de load-balancer uitvalt, is het systeem niet meer beschikbaar. Om dit op te lossen kan een tweede load-balancer als stand-by opereren, die het IP-adres van de primaire load-balancer overneemt als deze faalt, zodat clients geen aanpassingen hoeven aan te brengen om het systeem te kunnen benaderen.

Een virtuele IP-adres kan over meerdere clusternodes verdeeld worden, waardoor deze geen single point of failure oplevert. Het adres kan door andere nodes worden overgenomen als problemen optreden. Bij het gebruik van een virtuele IP-adres is de cluster software zelf verantwoordelijk voor het balanceren van de verbindingen over de nodes. Er komt dan een extra belasting op het cluster door het moeten verwerken van de verschillende verbindingen. Bij het gebruik van een load balancer neemt deze de taak van het balanceren op zich.

Een derde mogelijkheid is de intelligentie naar de clientside te verplaatsen. Dit is meestal alleen goed mogelijk als slechts weinig nodes worden gebruikt die niet snel wijzigen, omdat elke verandering van het cluster op alle clients veranderd moet worden. Bij het gebruik van een enkel fail-over cluster kan door middel van de client software de verbinding automatisch naar de back-up machine worden verplaatst. Op deze manier is het voor de gebruiker niet zichtbaar dat de verbinding overgezet is. Zowel Transparent Application Failover [52][53] van Oracle, als Automatic Client Reroute [54] van IBM en Transparent Client Redirection in de Microsoft Data Access Components (MDAC) [7] van Microsoft maken van deze technologie gebruik. Ook Sequoia biedt een dergelijke techniek aan voor de connectie van de clients met de controllers, waarbij ook Load Balancing vanaf de clients wordt geregeld.

3.4 Middleware Clustering

Bij het gebruik van middleware wordt de cluster techniek ingebouwd tussen de applicatie en de database. Door het toepassen van middleware kunnen een aantal features worden toegevoegd, zoals automatische fail-over en load balancing van client requests, replicatie van data over meerdere nodes en het verbeteren van de performance door caching van resultaten.

Er zijn twee verschillende soorten te onderscheiden, namelijk middleware die gebruikt maakt van het JDBC niveau of software direct tussen database en applicatie. JDBC middleware wordt beschreven in paragraaf 3.4.1, de directe middleware in paragraaf 3.4.2.

3.4.1 Java Database Connectivity (JDBC)

De Java Database Connectivity (JDBC) API [74] is een interface tussen een JAVA programma en een database. Elke applicatie die gebruik maakt van standaard JDBC kan op verschillende databases worden toegepast. Elke database die de JDBC standaard ondersteunt, kan dan door deze applicatie gebruikt worden. Als een fabrikant een JDBC driver heeft ontwikkeld, kan deze gebruikt worden door de applicatie.

JDBC middleware maakt gebruik van een eigen ontwikkelde JDBC driver, die door de applicatie wordt gebruikt in plaats van de database driver. Deze JDBC driver kan dan verschillende features aanbieden aan de applicatie of de gebruiker. De database driver van de database wordt door deze JDBC driver vervangen, waarna de applicatie contact maakt met de JDBC middleware. De middleware maakt vervolgens met de driver van de producent contact met de database. Doordat gebruik wordt gemaakt van standaard JDBC, hoeft de applicatie code niet herschreven te worden. De enige wijziging die moet worden aangebracht is dus een configuratiewijziging van de te gebruiken driver en database systeem.

Een tweede voordeel van middleware met JDBC maakt het mogelijk om gebruik te maken van heterogene databases. Doordat standaard JDBC wordt gebruikt, kunnen verschillende databases met een JDBC driver gebruikt worden als clusteroplossing. Hiermee wordt het mogelijk om bijvoorbeeld een MySQL en Oracle database samen te gebruiken voor één applicatie. Door het gebruik van JDBC kan ook gebruik worden gemaakt van bijvoorbeeld de Java Transaction API (JTA) [75]. Hierdoor kan transactie management worden toegepast op de binnenkomende transacties van de client en kan een transactieprocesmonitor worden toegevoegd aan de middleware.

Een nadeel van middleware is het niet kunnen gebruiken van specifieke database eigenschappen. Zo kunnen niet alle speciale features van een database worden ondersteund in een JDBC driver en kunnen eventuele optimalisaties van transacties of replicatie in de database niet gebruikt worden.

De JDBC middleware methode wordt gebruikt door Sequoia (het vroegere C-JDBC) [28] en HA-JDBC [29]. Beide worden beschreven in A.6.

3.4.2 Middleware

De tweede techniek die gebruikt wordt bij middleware is het onderscheppen van de requests van de clients, voordat deze naar de database software gestuurd worden. Dit onderscheppen gebeurt door de middleware door zich voor te doen als de databaseserver richting de applicatie. De requests van de applicatie worden opgesplitst in lees- of schrijfacties.

De leesacties worden door middel van load balancing doorgestuurd naar een van de database nodes, in de meeste gevallen de lokale database. Dit doorsturen gebeurt door de middleware door zich voor te doen als client richting de database.

Schrijfacties moeten naar alle database nodes, waarop deze data staat, worden doorgestuurd, zodat ze op alle databases worden uitgevoerd. Dit gebeurt door deze requests door te sturen naar andere middleware instanties die op de overige databasenodes draaien. Door middel van een protocol wordt vervolgens het request in de juiste volgorde verwerkt op alle nodes, zodat ze allemaal consistente data bevatten. Zowel de applicatie als de databaseserver weten niet dat er een stuk middleware software zich tussen hen in bevindt. Dit zorgt ervoor dat de applicatie of database niet aangepast hoeft te worden aan het cluster concept. Een voorbeeld van deze techniek is m/cluster 2005 van Continuent [30].

Software zoals m/cluster is alleen geschikt voor de database waarvoor het ontwikkeld is, in dit geval MySQL. In tegenstelling tot een JDBC middleware oplossing, kan deze software dus niet universeel worden gebruikt, maar is het specifiek ontwikkeld voor een bepaald database product en meestal ook een specifieke versie. Dat betekent dat je voor een dergelijke product gebonden bent aan een specifieke versie van de database, die door de ontwikkeling van de middleware veelal achterloopt op het nieuwste product van de database producent. Hoewel een heterogene database infrastructuur dan niet mogelijk is, kunnen wel specifieke features van de ondersteunde database gebruikt worden.

De keuze tussen één van beide middleware technieken moet dus gemaakt worden op de gebruikte cliëntsoftware en de wens om verschillende databaseproducten te kunnen gebruiken. Ook is het afhankelijk van de gewenste features die van het databaseproduct gebruikt moeten worden of middleware wel mogelijk is.

3.5 Operating System Clustering

Bij het clusteren op Operating System (OS) niveau worden de verschillende services die op een node draaien voorzien van mogelijkheden om op een andere node verder te draaien. Door middel van een fail-over of take-over kunnen services worden verplaatst binnen het cluster. Dit maakt het mogelijk om onderhoud uit te voeren en zorgt dat falende nodes geen invloed hebben op de beschikbaarheid van de service. Deze service kan zowel een database als een webserver, email server of een andere service zijn. Operating system clustering is dus niet specifiek ontwikkeld voor database clustering, zoals die beschreven is in paragraaf 3.3. Deze paragraaf beschrijft de verschillende technieken die bij OS clustering een rol spelen.

Om geen complexiteit toe te hoeven voegen aan de applicaties die een dergelijk cluster gebruiken, wordt een Single System Image (SSI) toegepast. De Single System Image laat het cluster fungeren alsof het één enkel systeem is, waardoor de applicatie niet weet dat er van een cluster systeem gebruik wordt gemaakt. De SSI wordt beschreven in paragraaf 3.5.1.

Om het systeem als een SSI te laten optreden is het noodzakelijk om services van de ene node naar de andere node te kunnen verplaatsen, zowel voor onderhoud als bij falende nodes. Voor de gebruiker moet het niet zichtbaar zijn op welke node de service draait, voor de gebruiker is er slechts één systeem. Om services van de ene naar de andere node te kunnen verplaatsen is allereerst een monitor nodig, die de service of de node in de gaten houdt. Hier wordt een heartbeat protocol voor gebruikt, samen met service specifieke monitors. Heartbeat wordt beschreven in paragraaf 3.5.2.

Om een single system image naar buiten toe als een systeem te laten communiceren is het nodig dat alle communicatie via één bepaalde weg naar buiten toe gaat en binnen komt. Een SSI is op twee manieren van een dergelijke ingang te voorzien, door middel van een load-balancer ofwel met een virtueel IP-adres. Beide technieken zijn al beschreven in paragraaf 3.3.2.

Binnen de clustering kan nog een onderscheid worden gemaakt tussen actieve en passieve systemen. Een fail-over cluster biedt meestal slechts een actief/passief systeem, high performance clusters bieden meestal volledig actieve systemen. Het verschil wordt beschreven in paragraaf 3.5.3.

3.5.1 Single System Image

Een Single System Image (SSI) [76] is een beeld wat door het systeem bij de applicatie wordt gewekt. Om te zorgen dat de complexiteit van het cluster niet bekend hoeft te zijn bij de applicatieprogrammeur, wordt een SSI gebruikt. Hierdoor is het voor de applicatie niet relevant of een cluster wordt gebruikt, de applicatie weet niet beter dan dat slechts een enkelvoudig systeem aanwezig is. In de applicatie hoeft daardoor geen rekening te worden gehouden met cluster specifieke technieken en kan er ontwikkeld worden alsof een simpel standalone systeem gebruikt zal gaan worden.

De cluster software neemt de taak op zich om de verschillende services over de nodes te verdelen, de services en nodes te monitoren en eventueel services naar andere resources te verhuizen. Alle nodes in het cluster moeten wel over dezelfde informatie kunnen beschikken, om bij een fail-over van nodes deze services goed over te kunnen nemen, zonder dat daar een herstart van de client voor nodig is.

Net zoals bij database clustering is ook hier de keuze benodigd tussen een shared-storage of distributed storage. In het geval van distributed storage is de opslag over de nodes verdeeld en niet centraal op een plek opgeslagen. In beide gevallen kan elke node bij de opslag op alle andere nodes. Een vorm van redundantie is benodigd om de beschikbaarheid van de data te kunnen garanderen. Er zijn verschillende cluster file systemen die deze functionaliteit bieden, zoals CODA [37] en GFS [37].

3.5.2 Heartbeat

Heartbeat [117] protocollen worden gebruikt om te controleren of de systemen nog online zijn. Elke node in een cluster stuurt om een bepaalde tijd een bericht naar de andere nodes om te laten weten dat deze nog aanwezig is. Heartbeats worden vooral op operating system niveau gebruikt voor controle. Bij applicaties worden veelal specifieke applicatiecontroles en monitoring gebruikt. Een aantal van deze monitors gebruikt ook heartbeat functionaliteit, maar meestal worden ook specifieke functies uitgevoerd om te controleren of de applicatie goed werkt. Dit is bij een heartbeat protocol niet gegarandeerd, daarbij is alleen zeker dat het systeem nog reageert.

Als geen heartbeat meer ontvangen wordt, dient de service overgenomen te worden van de falende node. Bij het optreden van een netwerkuitval treedt echter een probleem op. Op dat moment is niet meer duidelijk of het systeem niet meer reageert doordat het is uitgevallen of omdat het netwerk is uitgevallen. In veel gevallen is het actief zijn van beide systemen niet gewenst of levert dat zelfs grote problemen op. Als gebruik wordt gemaakt van shared-storage, kan via de storage een van beide nodes worden uitgeschakeld door het plaatsen van een bepaald disk pakketje, een zogenaamde “poison pill” [86]. Andere mogelijkheden zijn het gebruik van een powerswitch die door beide systemen gebruikt kan worden [87]. Het systeem dat als eerste de switch bereikt, wint en sluit daarmee het andere systeem af. Daarna zal een operator de afgesloten machine weer in de lucht moeten brengen en de correcte situatie moeten herstellen. Het is daarbij dus niet mogelijk om automatisch weer terug te gaan naar de originele situatie.

3.5.3 Stand-by modes

Bij clusters kan onderscheid worden gemaakt tussen systemen waarbij alle nodes actief zijn en systemen waarbij een actieve en passieve helft bestaat. Een systeem waarvan alle nodes actief zijn, kan de services op elke node draaien. Deze services kunnen worden overgezet naar een andere actieve node als er problemen optreden op één van de nodes.

Bij een systeem met actieve en passieve nodes draaien de services op de actieve node en wacht de passieve node af totdat de actieve node uitvalt. Behalve de afwacht- en monitorfunctie van de passieve node heeft deze verder geen actieve functies. Hoewel de passieve nodes geen verbetering van de performance leveren, is deze oplossing meestal makkelijker te implementeren, doordat niet op meerdere systemen in dezelfde data hoeft te worden geschreven. Bij een volledig actief systeem dienen daar wel voorzieningen voor gemaakt te worden. Een voordeel van alleen actieve systemen is verder dat bij deze duidelijk is dat ze allemaal functioneren. Bij een passief systeem is het mogelijk dat deze niet correct functioneert, maar dat dit pas wordt opgemerkt als een fail-over moet plaatsvinden. Een actief-actief systeem biedt op dat gebied meer veiligheid.

Binnen de stand-by modes kan nog onderscheid worden gemaakt tussen cold-stand-by en hot-stand-by. In het eerste geval dient een beheerder van het systeem zelf de overgang in gang te zetten, in het tweede geval wordt automatisch door de software zelf de switch gemaakt. Een hot-standby zal daardoor een hogere uptime kunnen behalen, doordat niet op menselijk handelen hoeft te worden gewacht bij problemen. Indien de monitoring correct werkt, zal het passieve systeem automatisch de services van het actieve systeem overnemen.

3.6 Toekomst

Deze paragraaf geeft een kleine blik op de mogelijke toekomst van High Availability oplossingen.

Als overtreffende trap voor het internet en samenwerkende systemen in cluster wordt het global GRID gezien. Het grid wordt gezien als een globale pool van computing resources waarin door gebruikers opdrachten kunnen worden uitgevoerd die te groot zijn voor de eigen machines binnen de organisatie. Instanties stellen binnen dit grid hun resources beschikbaar en gebruiken resources van anderen als ze deze nodig hebben [79].

De defacto standaard op het gebied van grids is de Globus Toolkit [77] middleware software die het mogelijk moet maken meerdere computers in een grid op te nemen die samen aan een project kunnen werken. Het is mogelijk om machines van verschillende organisaties te combineren in een grid en deze samen te laten werken. Een goed voorbeeld van het grid concept is het SETI@Home project waar naar buitenaards leven wordt gezocht door vele personen die hun machine beschikbaar stellen. Dit project voldoet echter niet aan alle voorwaarden van de grid definitie, zoals die gegeven is door Ian Foster [78], een van de ontwikkelaars van de Globus Toolkit, maar is wel een van de eerst succesvolle grid implementaties die aantonen hoe het grid concept kan werken.

Oracle biedt met de 10g (grid) database een product aan waarmee ze willen aangeven ook een soort grid te kunnen leveren. Hoewel sterk in ontwikkeling zal de grid architectuur nog een hoop hobbels moeten overwinnen, voornamelijk op organisatorisch en financieel vlak met betrekking tot de openstelling van resources tot het grid. Ook op het gebied van beschikbaarheid van resources en

het kunnen blijven functioneren van het volledige grid voor de applicaties zullen de nodige stappen moeten worden gezet om het aantrekkelijk te maken voor organisaties om hun resources beschikbaar te stellen en het grid te willen gebruiken.

Een ander gebied waar op dit moment veel onderzoek naar is, is het virtualisatie concept, waarin vooral VMware voorop loopt. VMware biedt met haar ESX server product virtualisatiesoftware die direct op de hardware draait [113]. Er zijn voorzieningen aanwezig die bij het uitvallen van een van de systemen de virtuele machines die op dat systeem draaien naar andere systemen kunnen verplaatsen. Door het virtualiseren wordt de software losgekoppeld van de hardware en wordt het veel eenvoudiger om de services te verplaatsen. Hetzelfde gebeurt bij OS clustering, maar bij virtualisatie kan ook het OS eenvoudig verplaatst worden, wat bij OS clustering niet mogelijk is. Voor het OS lijkt de virtuele machine een normale hardware machine, zodat het OS niet aangepast hoeft te worden. De emulatie van de hardware brengt wel een kleine vertraging met zich mee ten opzichte van directe hardware aansturing door het OS.

4. PRODUCTEN

Van de besproken technieken in hoofdstuk 3 zijn verschillende implementaties beschikbaar in verschillende producten. Een aantal producten biedt dezelfde opties aan, maar met een verschillende implementatie, waardoor ze verschillend functioneren. Dit hoofdstuk geeft een vergelijking van een aantal verschillende producten die gebruikt kunnen worden om data van een hoge beschikbaarheid te voorzien. Een uitgebreide beschrijving van elk product met de verschillende features bevindt zich in Appendix A. In Appendix B is een product overzicht opgenomen.

Omdat niet alle producten alle technieken implementeren en niet alle technieken even geschikt zijn voor een bepaalde situatie, moet voordat naar een product gekeken wordt eerst bepaald worden welke techniek nodig is. Een cluster oplossing is bijvoorbeeld uitstekend geschikt voor beschikbaarheid en performance, maar een cluster over twee locaties verdelen is meestal niet mogelijk. Paragraaf 4.1 geeft de mogelijkheid om via een beslissingsdiagram te bepalen welke techniek mogelijk is. Bij deze techniek volgt dan een overzicht van de mogelijke producten.

Paragraaf 4.2 biedt een vergelijking van de verschillende opties die mogelijk zijn bij de vijf database producenten die in dit onderzoek zijn vergeleken. Dit geeft een samenvatting van de informatie die in Appendix A en Appendix B beschikbaar is.

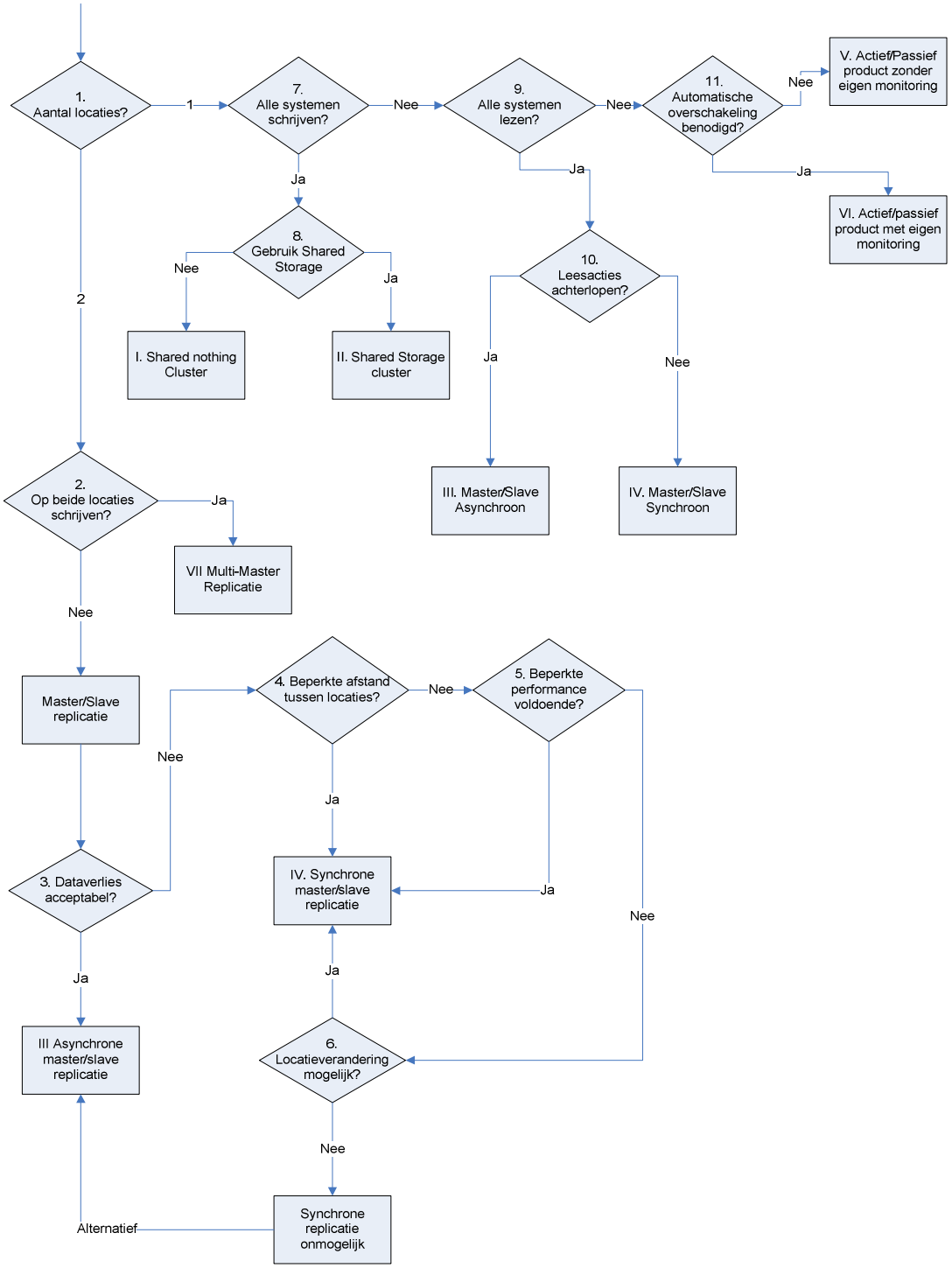
4.1 Welke techniek?

Om te bepalen welke techniek benodigd is voor de eisen die aan de applicatie worden gesteld met betrekking tot beschikbaarheid kan het beslissingdiagram in *Figuur 9: Beslissingdiagram benodigde techniek* gevolgd worden. Dit beslissingsdiagram geeft uitsluitsel over welke techniek het meest geschikt is voor de applicatie om toe te passen.

De belangrijkste beslissing die genomen moet worden is het gebruik van één of meerdere locaties. Bij het gebruik van meerdere locaties gaat de afstand tussen deze locaties een belangrijke rol spelen voor de mogelijke technologie die gebruikt kan worden. Het kiezen voor een tweede locatie en de afstand tussen deze twee locaties bepaalt in grote mate de keuze die overblijft voor een bepaalde techniek of product.

Is een tweede locatie niet noodzakelijk, dan is van belang wat er van het systeem verwacht wordt. Als alleen beschikbaarheid van belang is, kunnen alle technieken worden gebruikt. Als ook performance van belang is en op welke systemen gelezen en geschreven kan worden, wordt de keuze tot een bepaalde techniek beperkt. De voor en nadelen van een bepaalde techniek zijn in hoofdstuk 3 aan bod gekomen.

In paragraaf 4.1.1 worden de verschillende vragen beschreven bij het beslissingsdiagram in figuur 9. Paragraaf 4.1.2 beschrijft bij elk eindpunt in het diagram een aantal producten die bij de gekozen techniek passen.



Figuur 9: Beslissingdiagram benodigde techniek

4.1.1 Stappenbeschrijving

Deze paragraaf beschrijft de verschillende vraagstappen voor *Figuur 9: Beslissingdiagram benodigde techniek*.

1. Aan hoeveel locaties moet het systeem ondersteuning bieden? Indien het systeem moet kunnen omgaan met de uitval van een volledige locatie (datacenter uitval door stroomuitval, brand, overstroming, etc.), dan zijn er minimaal 2 locaties nodig. Indien geen rekening hoeft te worden gehouden met de uitval van een volledige locatie, kies dan voor 1 locatie.

2. Is het noodzakelijk dat beide locaties tegelijk gebruikt kunnen worden door de applicaties? Hier gaat het om de schrijfacties, indien schrijfacties slechts op één locatie tegelijkertijd worden uitgevoerd, kan gekozen worden voor 'Nee'.

3. Voor het repliceren tussen de locaties kan gekozen worden voor een synchrone of asynchrone replicatie. Indien dataverlies niet acceptabel is, dient gekozen te worden voor een synchrone replicatie oplossing. Hierbij zal de secundaire locatie altijd consistent zijn met de primaire locatie, waardoor geen dataverlies optreedt bij uitval van de primaire locatie. Als enig beperkt dataverlies acceptabel is voor de applicatie, kies dan voor asynchrone replicatie, omdat dit de beste performance van het systeem biedt.

4. De afstand tussen de twee locaties heeft een directe relatie met de latency. Hoe groter de afstand hoe groter de latency op de verbinding tussen deze twee punten. Bij synchrone replicatie levert deze latency een beperking op van de performance die mogelijk is van het database systeem, doordat de transactie wordt vertraagd met de latency van de verbinding. Als deze afstand beperkt is tot een aantal kilometers, waardoor de latency niet groter is dan 50ms, is het gebruik van een synchrone master/slave replicatietechniek nog mogelijk.

5. Indien de afstand groter is, waardoor de latency te groot is, is het afhankelijk van de benodigde performance of een synchroon master/slave replicatietechniek nog mogelijk is. Ga na of de benodigde performance kleiner is dan de aangeboden performance van de oplossing (zie stresstest resultaten voor uitslagen mbt. geteste producten).

6. Als de performance niet voldoende is en synchrone replicatie een eis is, dan zal de locatie van het tweede datacenter dichter bij de eerste gekozen moeten worden. Is dat niet mogelijk dan is synchrone replicatie niet mogelijk. Als alternatief kan dan gekozen worden voor een asynchrone replicatietechniek waarbij in de applicatie voorzieningen getroffen moeten worden voor het eventuele dataverlies dat op zou kunnen treden of een multihop systeem zoals dat is toegepast door Interpay (5.2)

7. Dienen alle systemen gebruikt te kunnen worden om op te schrijven?

8. Is er een shared-storage aanwezig dat gebruikt moet worden door de database?

9. Als niet op alle systemen geschreven hoeft te worden, is het gebruik van alle systemen om van te lezen wel noodzakelijk? Als van meerdere systemen gelezen kan worden, kan het systeem meer leesacties aan dan van een enkel systeem. Wel dient dan een vorm van load balancing ingezet te worden.

10. Als slaves worden gebruikt voor leesacties moet er een keus worden gemaakt of deze leesacties achter mogen lopen op wat er op de master geschreven is voor de gebruikte applicatie. Als dat niet zo is, dan moet een synchrone replicatie worden gebruikt. Kan dat wel, is asynchrone replicatie beter doordat dit de performance ten goede komt.

11. Als slechts op een systeem geschreven en gelezen hoeft te worden, kan gebruik worden gemaakt van een actief/passief systeem. Daarbij is de overschakeling tussen beide belangrijk om te zorgen dat het systeem beschikbaar blijft. Dient deze overschakeling automatisch te gebeuren of kan dit door goede procedures handmatig gebeuren?

4.1.2 Producten bij technieken

Voor alle technieken uit het schema in *Figuur 9: Beslissingdiagram benodigde techniek* zijn in *Tabel 4: Producten bij technieken* de producten opgenomen.

Het is mogelijk dat een product ontbreekt bij een bepaalde techniek of ook toepasbaar is voor andere technieken. Zo kan een master/slave replicatie systeem ook als een actief/passief systeem gebruikt worden, als de gebruiker de slaves niet gebruikt. Om de indeling overzichtelijk te houden, zijn de producten ingedeeld op hun voornaamste functie waarvoor zij ontwikkeld zijn.

Hoofdstuk 3 biedt een beschrijving van de verschillende technieken, het paragraafnummer staat tussen haakjes. De producten worden in paragraaf 4.2 en Appendix A beschreven.

Tabel 4: Producten bij technieken

Nummer	Techniek	Producten
I	Shared-nothing Cluster (3.3.1)	Sequoia, MySQL Cluster, PostgreSQL PgCluster
II	Shared-storage Cluster (3.3.1)	Oracle FailSafe, Oracle RAC, MS SQL n-Way cluster
III	Asynchrone master/slave replicatie (3.2)	MySQL Replicatie, Oracle Data Guard, Microsoft SQL Server Log Shipping, IBM Q-replication, PostgreSQL Slony-I
IV	Synchrone master/slave replicatie (3.2)	Oracle Data Guard, PostgreSQL Postgres-R
V	Actief/Passief zonder monitor (3.5.3)	Microsoft SQL Server Data Mirroring zonder witness, IBM DB2 HADR
VI	Actief/Passief met monitor (3.5.3)	Microsoft SQL Server Data Mirroring met witness
VII	Multi-Master replicatie (3.2)	Sequoia, PostgreSQL Slony-II

De producten in categorie V zonder eigen monitoring kunnen met externe monitorsoftware in categorie VI worden toegepast. Hiervoor dient de oplossing dan met cluster management software te worden uitgebreid.

4.2 Productvergelijking

Appendix B bevat een product overzicht van de product mogelijkheden van de vijf database producenten die in dit project zijn vergeleken. Voor deze vergelijking zijn de grote drie database producten van Oracle, IBM en Microsoft vergeleken met de twee belangrijkste open-source producten van MySQL en PostgreSQL. Volgens LaMonica [94] hadden de grote drie respectievelijk 40, 31 en 12 procent van de databasemarkt in 2004. Alternatieven zoals Interbase/Firebird, Sybase en anderen zijn buiten beschouwing gelaten door hun kleine aandeel op de markt.

De producten van de grote drie database producenten, Oracle, IBM en Microsoft bieden ieder hun eigen implementatie van de verschillende technieken die in hoofdstuk drie besproken zijn. Oracle biedt met hun Data Guard een master/slave techniek aan die zowel synchroon als asynchroon kan werken. IBM biedt dezelfde techniek aan onder de naam High Availability Disaster Recovery en Microsoft noemt het Database Mirroring. Opvallend verschil tussen deze producten is de mogelijkheid bij Oracle om de slave te gebruiken om leesacties op uit te voeren. Met de Oracle Flashback tool kan de slave zelfs gebruikt worden om naar toe te schrijven, bijvoorbeeld voor het aanmaken van rapporten. Zowel bij Microsoft als IBM is het niet mogelijk om de slave te gebruiken, tenzij de replicatiemodus wordt onderbroken. Oracle heeft hier dus een belangrijk voordeel op de overige twee. Zowel bij PostgreSQL Slony-I als MySQL Replicatie is lezen van de slave wel mogelijk bij hun replicatietechnieken, echter beide ondersteunen alleen een asynchrone replicatietechniek. Hier loopt de slave dus achter ten opzichte van de master, waar Oracle dit in de synchrone modus niet heeft. Voor een synchrone replicatie modus waarbij leesacties op de slave moeten kunnen worden uitgevoerd is dus alleen het Oracle Data Guard product geschikt.

Op het gebied van fail-over lopen de producten ook uiteen. Bij zowel Oracle als Microsoft wordt de functionaliteit hiervoor standaard meegeleverd en is een automatische fail-over mogelijk. IBM biedt met zijn HADR oplossing wel mogelijkheden tot automatische fail-over, maar hier is externe monitoring software benodigd die de overschakeling regelt. Het HADR product biedt deze ondersteuning zelf niet. Ook bij MySQL en PostgreSQL is deze overschakeling van slave naar master van externe software afhankelijk. Het inzetten van clustermanagement software zoals Linux-HA, Veritas cluster manager, Lifekeeper en anderen is voor de beschikbaarheid van de oplossing dus noodzakelijk.

Zowel Microsoft als Oracle bieden software aan waarmee van de MSCS functionaliteit van Windows Servers gebruik kan worden gemaakt. Dit maakt het mogelijk om een cluster van enkele Windows machines op te zetten waarop de database draait. Voor deze cluster oplossingen is het gebruik van shared-storage een verplichting. IBM biedt een dergelijke oplossing niet aan en heeft ook geen vergelijkbare optie voor Oracle RAC. Bij IBM kan wel voor een Parallel Sysplex als cluster worden gekozen, maar dan dient ook IBM hardware en het AIX besturingssysteem te worden aangeschaft. Bij de open-source producten van MySQL, PostgreSQL en ook Sequoia wordt alleen een shared-nothing architectuur toegepast. Het gebruik van meerdere systemen is veelal goedkoper

dan een gespecialiseerde shared-storage oplossing, waardoor door de open-source producten voor de shared-nothing architectuur wordt gekozen. Hierdoor bieden de open-source producten niet alleen voordeel op softwarelicenties maar ook op hardware kosten. Bijzonder aan versies 4.1 en 5.0 van MySQL Cluster is de geheugen gebaseerde methode voor data opslag. Hierbij wordt de storage op disk slechts als back-up gebruikt mocht het hele cluster uitvallen door bijvoorbeeld stroomuitval en staat de consistente data in het geheugen van de systemen. Dit betekent wel dat de gebruikte systemen voldoende geheugen moeten bevatten om de dataset aan te kunnen. In versie 5.1 van MySQL Cluster wordt voor alle velden waar geen index voor aanwezig is ook schijfgebaseerde opslag geboden. Velden met een index of indexen zelf zullen wel nog in geheugen blijven. Geen enkel concurrerend product biedt een puur geheugen gebaseerde oplossing aan vergelijkbaar met MySQL Cluster.

Op het gebied van kosten biedt Oracle het duurste pakket aan met de Enterprise Edition voor $40.000, IBM volgt met $33.125 en bij Microsoft kost het slechts $25.000 per processor. Opvallend is dat Oracle RAC een onderdeel is van de Standard editie, maar als optie bij de Enterprise editie gekocht moet worden. De beide open-source producten hebben geen aanschaflicentiekosten, maar bieden wel service of supportcontracten aan. Hoewel dit een stuk goedkoper is dan de aanschafkosten, dient ook met kosten voor beheer en onderhoud, en training van medewerkers rekening te worden gehouden. Het voert hier te ver om een uitspraak te doen over de Total Cost of Ownership van al deze producten. Meer daarover kan op de sites van de fabrikanten en onderzoeksbureaus gevonden worden [99][100][101][102].

Interessant om te zien bij de grote drie producenten is de ontwikkeling van producten en functies ter voorkoming van menselijke fouten. Vooral Oracle loopt hierin voorop met de Flashback tool [95]. Deze tool maakt het mogelijk om verwijderde versies van data terug te halen door middel van een snapshot techniek. Ook het toevoegen van een soort prullenbak techniek met Flashback Drop [95], zoals bekend van Windows, zorgt ervoor dat bij het optreden van menselijke fouten vrij eenvoudig ingegrepen kan worden, zodat data hersteld kan worden.

Microsoft biedt met Log Shipping een functionaliteit waarbij de vertraging van verscheping ingesteld kan worden. Zo kan bijvoorbeeld 5 minuten ongedaan gemaakt worden, omdat de slave 5 minuten data achterloopt op de master. Indien een tabel wordt verwijderd, heeft een beheerder nog 5 minuten om deze vanaf de slave te herstellen. Dit heeft natuurlijk als nadeel dat als de master uitvalt, vijf minuten data verloren gaat omdat deze nog niet op de slave is bijgewerkt. Ook moet de beheerder met zijn neus op het systeem zitten om binnen die tijd de fout ongedaan te maken. Het uitbreiden van de vertraging levert dan een groter dataverlies bij uitval op. Bij de open-source producten richt men zich vooral op technologische uitbreidingen, zoals het MySQL Cluster product en heeft het voorkomen van menselijke fouten (nog) minder aandacht.

Concluderend loopt Oracle voorop in het aantal features op het gebied van High Availability ten opzichte van de andere grote twee, maar bieden de open-source producten hun eigen benadering van het probleem. Voor bepaalde technieken,

zoals synchrone replicatie met toegang tot de slaves, bieden slechts een of twee producten een oplossing, zodat een duidelijke keuze voor een bepaalde techniek belangrijk is om te bepalen welk product gebruikt moet gaan worden. Het kiezen van een product voordat duidelijk is welke situatie opgelost moet worden en welke techniek daarvoor noodzakelijk is, is dus niet verstandig. Het is dan goed mogelijk dat het product niet de juiste techniek of gewenste features brengt die nodig zijn voor het project.

5. PRAKTIJK OPLOSSINGEN

Om te bepalen in welke mate de High Availability theorie ook in de praktijk wordt toegepast, zijn meerdere bedrijven gevraagd voor het geven van een interview met betrekking tot High Availability in de praktijk. Alleen Ilse Media wilde aan een dergelijk interview meewerken, andere organisaties waren niet geïnteresseerd. Een volledig verslag van het Ilse Media interview staat in appendix C.

Dit hoofdstuk beschrijft van een aantal organisaties welke beschikbaarheidmaatregelen ze hebben genomen. Aangezien alle informatie, behalve bij Ilse Media, niet uit directe bronnen voortkomt en deze niet door een interview konden worden gecontroleerd, is het mogelijk dat organisaties inmiddels andere producten gebruiken of dat het beschrevene niet volledig accuraat of correct is.

5.1 Make or Buy

De “make or buy” beslissing voor het invoeren van een hoog beschikbare oplossing is een lastige beslissing. In veel gevallen biedt het kopen van een product geen volledige oplossing. Zo moet in veel gevallen in de applicatie toch rekening worden gehouden met de gekozen oplossing, wat betekent dat deze applicatie ontworpen moet worden om gebruik te maken van de beschikbare technieken.

Veel producten bieden alleen een oplossing voor het eigen domein of schuiven een deel van de verantwoordelijkheid naar een ander domein. Zo biedt het MySQL Cluster wel beschikbaarheid voor de datanodes, maar het bereiken van de API nodes door de clients moet door een load-balancer worden geregeld. Hier wordt de connectie verschoven naar een ander domein. Bij het HADR product van IBM geldt dit voor de monitoring en de overschakeling indien er fouten optreden.

Er bestaat geen enkel product dat voor alle mogelijke situaties een oplossing biedt, er zal dus altijd van verschillende componenten een geheel moeten worden gesmeed dat goed samenwerkt. Hoewel je dus als organisatie producten kan kopen, ontkom je er als organisatie niet aan om zelf de producten en applicatie te integreren in een complete oplossing, tenzij je dit ook uitbesteedt. Voor het hoog beschikbaar maken van een service of applicatie is dus meer nodig dan alleen een product aanschaf, zowel de ontwikkeling als het beheer dient goed geregeld te worden.

Zowel de Rabobank als Interpay maken voor het verkrijgen van een dergelijke oplossing gebruik van organisaties als IBM en HP, die zowel hardware, software als services leveren om een High Availability systeem te ontwerpen en ontwikkelen. Op kleinere schaal zoals bij Ilse wordt veelal gewerkt met consultants die specifieke kennis van dergelijke oplossingen hebben en kunnen adviseren bij het ontwerpen en ontwikkelen van een systeem. Doordat de kennis vrij specialistisch is, is het voor veel organisaties niet aantrekkelijk om de kennis zelf in huis te halen of er ervaring mee op te doen, omdat het niet tot hun core

business behoort. Daardoor wordt meestal het hele traject uitbesteed aan een specialistische organisatie of een leverancier van totaaloplossingen als IBM.

5.2 Interpay

Interpay verzorgt een groot deel van het Nederlandse betalingsverkeer in zowel binnen- en buitenland. Hierbij gaat het om pin, chip en creditcard betalingen en betalingen over internet. In 2001 is door Interpay besloten een nieuw uitwijksysteem te gaan toepassen. Voor dit systeem was er sprake van een dataverlies van 24 uur indien het datacentrum uit zou vallen. Door middel van tape back-ups werd de data van de primaire locatie naar een back-up locatie gebracht, waarbij deze back-ups slechts een keer per dag gemaakt werden. Omdat slechts een keer per dag met De Nederlandse Bank (DNB) verhandeld hoefde te worden, kon bij verlies van de transacties de banken om nieuwe gegevens worden gevraagd. Doordat vanaf 2001 de verhandeling elk half uur moest plaatsvinden en ook het dataverlies tot maximaal 4 uur beperkt moest worden van het DNB, was een nieuwe oplossing benodigd [96].

Het nieuwe systeem is ontwikkeld door IBM en is op basis van een multihop strategie ontworpen. Doordat IBM in 2000 slechts 50km kon afleggen met de toen beschikbare glasvezel technologie, kon de afstand tussen beide datacenters niet direct overbrugd worden. Een oplossing met een versterkend station was te risicovol in verband met uitval van dit station. Als oplossing is een synchrone replicatie bedacht tussen twee datacenters op slechts enkele kilometers afstand. Vanaf een van deze locaties wordt dan asynchroon naar het secundaire datacentrum gerepliceerd. Dit betekent dat uitval van één datacentrum opgevangen kan worden door het tweede synchroon lopende datacentrum. Als beide uitvallen door bijvoorbeeld een overstroming is altijd op de secundaire locatie nog een asynchrone datastroom aanwezig, waaruit dan hooguit enkele minuten ontbreken [96]. Aangezien een dataverlies van 4 uur is geëist door DNB is dit meer dan acceptabel, zeker als een groot deel van Nederland overstroomd is.

Indien dus de afstand tussen beide locaties te groot is, kan geopteerd worden voor een multihop systeem waarbij de voordelen van een synchrone en asynchrone techniek worden gecombineerd, zodat bij uitval altijd nog een optimale oplossing overblijft. Hiervoor is dan wel een extra datacenter locatie benodigd die binnen enkele kilometers van het primaire datacentrum ligt. Het voordeel van deze extra locatie ten opzichte van plaatsing binnen één locatie is de tolerantie voor uitval van deze locatie door bijvoorbeeld stroomuitval.

5.3 Rabobank

Voor de Rabobank is het beschikbaar zijn van de website waarmee internetbankieren wordt afgehandeld zeer belangrijk. De imagoschade voor de Rabobank en financiële problemen voor klanten als internet bankieren niet beschikbaar is, zijn aanzienlijk. Het internetbankieren systeem draait op het HP NonStop systeem dat gebaseerd is op Unix [36]. Door middel van een Remote Data Facility kan de data op meerdere plekken worden opgeslagen, zodat het systeem bestand is tegen uitval van een datacenter.

Een tweede belangrijke applicatie bij de Rabobank is de OnLine Integraal (OLI) applicatie. Door middel van de OLI applicatie kunnen alle werknemers bij de informatie van klanten komen als dat nodig is. Dit maakt het mogelijk om vanuit elke locatie een klant te kunnen bedienen, ook al is de klant op een andere locatie ingeschreven. Ook dit systeem dient beschikbaar te zijn, bij uitval kunnen klanten niet meer worden geholpen met hun bankzaken en vallen de werkzaamheden in de bankfilialen van Rabobank stil.

De OLI applicatie krijgt 9 miljoen hits per dag en wordt verwerkt op één 16 processors tellende NonStop S86000 Server systeem van HP. Dit systeem is uitgerust met een drievoudige modulaire redundantie. Meerdere keren per dag worden transacties uitgewisseld met het IBM mainframe, dat de geldzaken van de bank regelt [36].

Omdat de beschikbaarheid voor een bank als de Rabobank zo belangrijk is, kan geïnvesteerd worden in een systeem met drievoudige redundantie. Deze systemen zijn voor de gewenste beschikbaarheid van applicaties voor een groot aantal organisaties veelal te duur, maar bij Rabobank wegen ze op tegen de kosten van het niet beschikbaar zijn.

5.4 eBay/ Marktplaats

eBay gebruikt voor hun veilingsysteem een veelvoud aan computers. Het zoeksysteem waarmee items en veilingen gevonden kunnen worden bestaat uit 40 Solaris systemen die door middel van Veritas cluster management software van beschikbaarheid worden voorzien. eBay is in alle producten gestandaardiseerd op het gebruik van Veritas, welk probleem ze ook moeten oplossen [97]. Als database wordt Oracle gebruikt, waarbij van Veritas de Database Edition voor Oracle wordt gebruikt, zodat zowel back-ups als overschakeling door middel van Veritas geregeld kan worden.

Marktplaats dat door eBay is opgekocht in 2004 heeft door Stone-IT een eigen oplossing laten ontwikkelen met behulp van de LAMP stack [98]. Deze stack maakt gebruik van vier open-source componenten, genaamd Linux, Apache, MySQL en PHP dat een zeer schaalbare oplossing kan leveren. Dit systeem maakt gebruik van partitie van data en het repliceren tussen verschillende MySQL servers. Door middel van Linux Virtual Server wordt een softwarematige load balancing ingezet. Marktplaats verwerkt met een cluster van meer dan 50 systemen [98] meer dan 3,3 miljoen hits per dag [103].

eBay gebruikt dus als best-practice het standaardiseren op een of twee software producten en daar alle oplossingen omheen bouwen. Bij marktplaats is voor het zelf ontwikkelen van een oplossing gekozen, hoogstwaarschijnlijk omdat ten tijde van het opstarten van marktplaats in 1999 oplossingen van bijvoorbeeld Oracle of partijen als IBM en HP te duur waren of onvoldoende schaalbaar en door middel van open-source deze kosten drastisch beperkt konden worden.

5.5 Ilse Media

Ilse Media heeft ongeveer 40 websites onder haar beheer. Een aantal van deze websites bieden informatie aan een aantal miljoen gebruikers per dag, zoals de nieuwssite nu.nl en de linkspagina startpagina.nl. Om te zorgen dat al deze gebruikers continu de hele dag de websites kunnen bezoeken, heeft Ilse een aantal maatregelen getroffen om de websites beschikbaar te maken en houden.

Ilse Media gebruikt vier verschillende niveaus om de beschikbaarheid van hun websites te behalen. Niveau 1 is het laagste niveau waar op basis van best-effort de beschikbaarheid van de website wordt geregeld. Niveau 4 geeft een 99.99% garantie op beschikbaarheid. Alle beschikbaarheidcijfers worden berekend per maand. Dit betekent dat de maximale continue downtime beperkt wordt ten opzichte van een grotere periode zoals een jaar. Deze niveaus zijn vergelijkbaar met de niveaus die besproken zijn in 2.2.4. Een website of applicatie wordt ingedeeld in een van de vier niveaus aan de hand van de soort applicatie, de mogelijke imagoschade die optreedt bij uitval en het beschikbare budget. Hoewel de inkomstenbron uit advertenties voor Ilse belangrijk is, is imagoschade de grootste drijfveer om haar websites van een bepaald beschikbaarheidniveau te willen voorzien. De niveau-indeling gebeurt op basis van belangrijkheid van de site en of de kosten van het niveau opwegen tegen de kosten van uitval. De belangrijkste sites nu.nl en startpagina.nl draaien op niveau 4, de Ilse zoekmachine draait door de grote behoefte aan hardware op niveau 3. Het eventuele verlies van de zoekmachinefunctionaliteit indien een volledig datacenter uitvalt, is acceptabel voor Ilse gezien de kleine kans op het optreden van een dergelijke volledig datacenter uitval.

Alle geserveerde pagina’s worden statisch aangeleverd door het Content Management Systeem (CMS) dat in het back-office draait. Omdat bijna alles statisch is, is het gebruik van databases beperkt. De databases worden gerepliceerd in een Master-Slave set-up, waarbij de overschakeling van de slave, indien de master uitvalt, handmatig gebeurt. Omdat automatische overschakeling risico’s meebrengt voor de data en de monitoring van de database goed op orde is, is het handmatig overschakelen een goede optie voor Ilse. De data die in de databases wordt bewaard hoeft slechts een beperkte tijd te worden bewaard. Veelal is een termijn van vier weken voldoende. Daardoor is opslagcapaciteit en het maken van back-ups geen probleem.

Ilse Media maakt gebruik van een hostingprovider met veel overcapaciteit. Deze overcapaciteit wordt ingezet voor het vervangen van kapotte systemen en het bijplaatsen van systemen als er capaciteitstekorten ontstaan. Er wordt slechts één hostingpartij gebruikt, zodat er één verantwoordelijke is en één aanspreekpunt. Het verdelen over meerdere hostingpartijen werkt niet goed door de lastige communicatie en het afschuiven van verantwoordelijkheden. De hostingpartij is verantwoordelijk voor de hardware en het OS, en dient dat ook zelf te monitoren. Ilse neemt het applicatieniveau voor haar rekening.

Om de vele bezoekers te kunnen hanteren, wordt door Ilse gebruik gemaakt van hardware load-balancers en caching machines. Deze zorgen ervoor dat de verbindingen over de verschillende webservers worden verdeeld en dat content, die niet veranderd is, gelijk vanuit de cache geserveerd kan worden. Op deze manier wordt de druk van de webservers afgehaald. Indien alles uit zou vallen, is er nog een fall-back mogelijk naar het KPN Cache systeem. Dit systeem cachet de gegevens van de websites van Ilse en levert deze uit, als de systemen van Ilse zelf dit niet meer kunnen. Dit biedt dan verouderde informatie, maar er treedt minder imagoschade op dan als de volledige website uit de lucht zou zijn.

Ilse heeft ervaring met het uitvallen van meerdere componenten, het optreden van storingen en het testen van de opstelling en heeft in het afgelopen jaar nog geen moment gehad waarop de websites niet beschikbaar waren voor de bezoekers.

Een overzicht van de best-practices van Ilse samen met het volledige interview staat in Appendix C.

6. TESTOPSTELLING

Dit hoofdstuk beschrijft de testopstelling die gebruikt is om een aantal producten te kunnen testen op bruikbaarheid bij het komen tot een zo hoog mogelijke beschikbaarheid van een systeem en toepasbaarheid binnen projecten van Technolution.

De verschillende doelen van de testopstelling staan beschreven in paragraaf 6.1. De achtergrond van het project waarnaar de testopstelling gemodelleerd wordt, staat beschreven in paragraaf 6.2. Paragrafen 6.3 en 6.4 geven de eisen en aannames waar de opstelling aan moet voldoen. De producten die getest zullen gaan worden, staan beschreven in paragraaf 6.5. In paragraaf 6.6 staat een beschrijving van de stresstest die uitgevoerd zal worden om te bepalen welke invloed de verschillende HA features op de performance van het systeem hebben. Om de afstand tussen de twee locaties te simuleren, wordt gebruik gemaakt van een WAN link simulator, die beschreven staat in paragraaf 6.7 In paragraaf 6.8 wordt tenslotte een argumentatie gegeven voor de verschillende test- en onderzoekscases die zijn uitgevoerd op de gekozen producten. Deze test- en onderzoekscases zijn opgenomen in Appendix D.

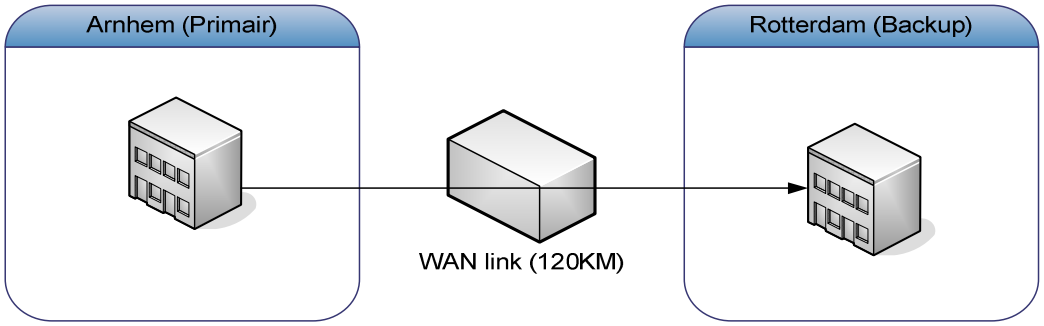
6.1 Doel

Deze testopstelling heeft als doel om aan te tonen of de geselecteerde producten geschikt zijn om een systeem door te laten draaien, zelfs als een van beide locaties uitvalt door een bom. Belangrijk bij het testen is het opdoen van ervaring met de producten en de consequenties van het gebruik van deze producten in kaart te kunnen brengen. De gebruiksvriendelijkheid en beheersbaarheid van de producten zal worden onderzocht. Uiteindelijk zal een advies kunnen worden uitgebracht of de producten geschikt zijn voor gebruik in dergelijke projecten.

6.2 Context

De testopstelling wordt gemodelleerd naar een bestaand project dat gebruikt wordt om alarmen via een meetnetwerk op te merken en te verwerken. Omdat uitval van één locatie geen gevolgen mag hebben voor de werking van het systeem is door de klant besloten dat twee locaties gebruikt moeten worden. Het beleid van de klant schrijft voor dat in deze gevallen minimaal twee locaties benodigd zijn. De hoofdlocatie bevindt zich in Arnhem, de back-up locatie in Rotterdam. De afstand tussen deze twee locaties is ongeveer 120 Km. *Figuur 10: Situatie schets* geeft de situatie weer.

Behalve de afstand tussen beide locaties, zijn de specifieke locaties verder niet relevant voor de testopstelling. In de rest van de opstelling zal verder worden gesproken over locaties A en B. Locatie A is de hoofdlocatie en locatie B fungeert als back-up. Als het systeem correct functioneert, zal deze op locatie A operationeel zijn.



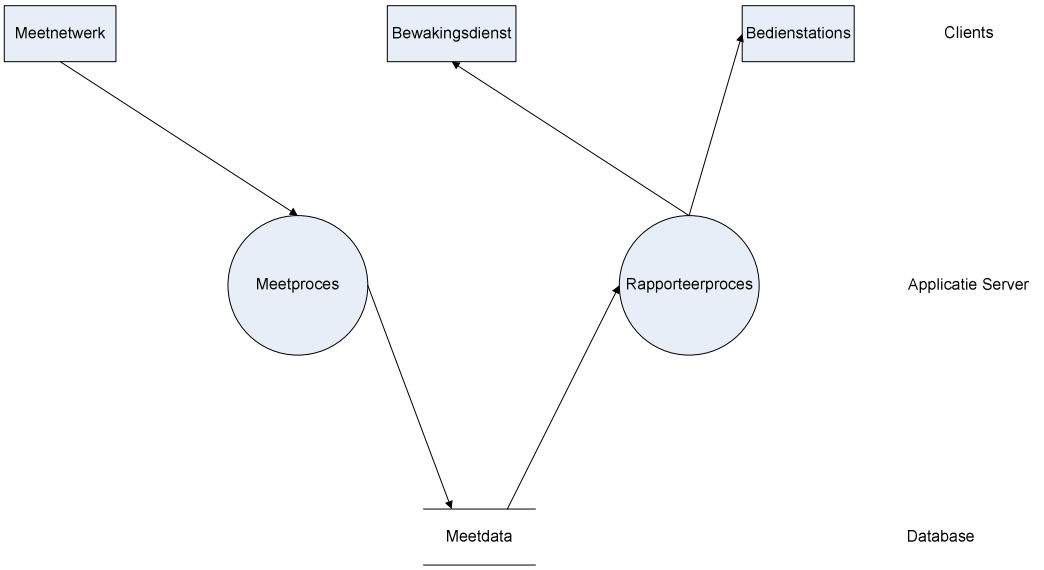
Figuur 10: Situatie schets

Het systeem wordt door een extern meetnetwerk van meetwaardes voorzien. Deze worden in het meetproces verwerkt en opgeslagen als meetdata. De meetdata wordt door een rapporteerproces verwerkt en gerapporteerd. Deze rapportage kan op de bedienstations worden bekeken. Een externe bewakingsdienst krijgt de rapporten doorgestuurd en kan daarop actie ondernemen. *Figuur 11: Meetsysteem* geeft het systeem weer. Beide locaties bevatten de processen en de dataopslag en hebben een verbinding met de clients.

Het systeem heeft dus twee verschillende clients, meetclients en rapportageclients.

De volgende eigenschappen van het systeem worden aangenomen:

- 200.000 metingen per maand via de meetclients;
- 13 meetclients die de metingen doorsturen;
- Elke minuut wordt een rapport opgevraagd door de bedienstations (43200 per maand);
- Het aantal bedienstations is niet gespecificeerd.



Figuur 11: Meetsysteem

6.3 Eisen

De te implementeren oplossing voor dit project moet aan de volgende eisen voldoen:

Eis 1: Het uitvallen van een enkel systeem verantwoordelijk voor de database of de applicatie server, heeft geen gevolgen voor het doordraaien van de applicaties. Zowel de alarmverwerking als de rapportering moet blijven functioneren.

Eis 2: Het uitvallen van de WAN link tussen locatie A en locatie B mag geen gevolgen hebben voor het doordraaien van de applicaties. Alarmen kunnen worden verwerkt en gerapporteerd en de secundaire locatie moet worden gesynchroniseerd bij herstel van de link.

Eis 3: Bij het uitvallen van de primaire locatie mag geen data verloren gaan. Metingen die door het systeem aangenomen zijn en verwerkt worden of al afgehandeld zijn, dienen bewaard te blijven. Nog niet aangenomen metingen dienen door de secundaire locatie te worden aangenomen bij uitval van de primaire locatie.
(Recovery Point Objective (RPO) = 0)

Eis 4: Alleen bij het uitvallen van de volledige primaire locatie mag de client handmatig worden omgezet naar het secundaire systeem.

Eis 5: Bij het uitvallen van een enkel systeem op de primaire locatie moet de client zonder problemen door kunnen werken zonder dat de client hiervoor actie moet ondernemen.

Eis 6: Bij het uitvallen van de primaire locatie is het acceptabel als de secundaire locatie binnen 4 uur online is. (Recovery Time Objective (RTO) < 4 uur). Minder dan 30 minuten is wenselijk.

Eis 7: Een component dat uitgevallen is, dient binnen 8 uur gerepareerd te kunnen worden. Omdat het systeem slechts één failure tegelijkertijd aankan, is het na het uitvallen van één component noodzakelijk om deze component zo snel mogelijk te repareren. Het optreden van een tweede failure voordat de eerste is opgelost, zal slechts in beperkte gevallen niet tot downtime leiden en dient voorkomen te worden. Procedures en processen dienen op deze tijden te worden afgestemd door middel van service contracten of reserve componenten.

De RPO en RTO zoals genoemd in eisen 3 en 6 staan weergegeven in *Figuur 1: Recovery Time en Point Objectives* in paragraaf 2.2.1. Een beschrijving van deze begrippen wordt gegeven in paragraaf 2.2.1

6.4 Aannames

Voor de testopstelling zijn de volgende aannames gedaan:

Aanname 1: Er treedt slechts één fout tegelijkertijd op. Dit kan een enkel systeem zijn, een enkele netwerkverbinding of een enkele locatie.

Aanname 2: Voor een afstand van 120km is 50ms latency realistisch, zie 6.7.

Aanname 3: Meetclients zijn in staat hun metingen te bufferen, zodat een bepaalde tijd geen metingen verwerkt hoeven te worden.

Aanname 4: Er worden 5 bedienstations gebruikt, voor een totaal van 5 x 43200 = 216000 rapport opvragen per maand.

Aanname 5: Het verwerken van een meting is het toevoegen van een enkele rij aan de database. Het opvragen van een rapport is het aantal metingen van het laatste uur.

Aanname 6: Beide locaties hebben toegang tot het meetnetwerk en een connectie met de bewakingsdienst. Het meetnetwerk is niet afhankelijk van een van beide locaties en valt dus niet uit bij uitval van een locatie.

6.5 Gekozen producten

Voor de testopstelling is gekozen voor drie verschillende producten. Er wordt gebruik gemaakt van een middlewareoplossing, een clusteroplossing en een replicatieoplossing. Als het zelf ontwikkelen van een oplossing buiten beschouwing wordt gelaten en niet gekeken wordt naar oplossingen op lagere niveaus als OS of SAN replicatie, zijn dit de drie opties waaruit gekozen kan worden bij een High Availability opzet.

Het testen van een replicatieproduct op OS niveau of van SAN replicatie ligt buiten de scope van dit project. Het aanschaffen van een SAN storage systeem en replicatiesoftware puur en alleen voor deze testopstelling is niet verantwoord door de hoge kosten. Binnen de industrie worden beiden ook niet gebruikt voor de replicatie van databases, maar vooral voor file replicatie. Aangezien file replicatie geen rol speelt in deze testopstelling, valt het testen van deze technieken buiten de scope.

Er is gekozen voor de volgende drie producten, IBM DB2 met HADR, MySQL Cluster en Sequoia middleware. Dit zijn respectievelijk een replicatie oplossing, een cluster oplossing en een middleware oplossing.

IBM DB2 High Availability Disaster Recovery (HADR) biedt een oplossing voor het gebruik van meerdere locaties en is daar specifiek voor ontwikkeld. Zowel synchrone als asynchrone replicatie is mogelijk met HADR. Daarmee is het een goede kandidaat voor deze testopstelling en kan ook een goede vergelijking worden gemaakt tussen een asynchrone en synchrone replicatie. Zowel Oracle als Microsoft bieden vergelijkbare opties, Microsoft werkt echter niet onder

Linux, het gekozen OS in deze opstelling en de evaluatieversie van Oracle is beperkter in tijd, waardoor IBM een betere optie is voor deze testopstelling. Het testen van beide is zeker interessant in de toekomst om te bepalen of de features echt vergelijkbaar zijn of dat één er bovenuit steekt.

MySQL biedt met MySQL cluster een oplossing die 99,999% beschikbaarheid zou moeten halen. In versie 5.1 van MySQL kan ook tussen twee clusters worden gerepliceerd, waarmee meerdere locaties mogelijk zijn. Ook is support voor disk-based clustering toegevoegd, waarmee het een interessante optie lijkt te worden voor cluster oplossingen. Versie 5.0 biedt alleen in-memory clustering, waardoor dit voor grote datasets geen goedkope optie is door de hoeveelheid geheugen die benodigd is. Hoewel versie 5.1 van MySQL nog in beta is, kan dit al een goede indruk geven van de bruikbaarheid van deze software.

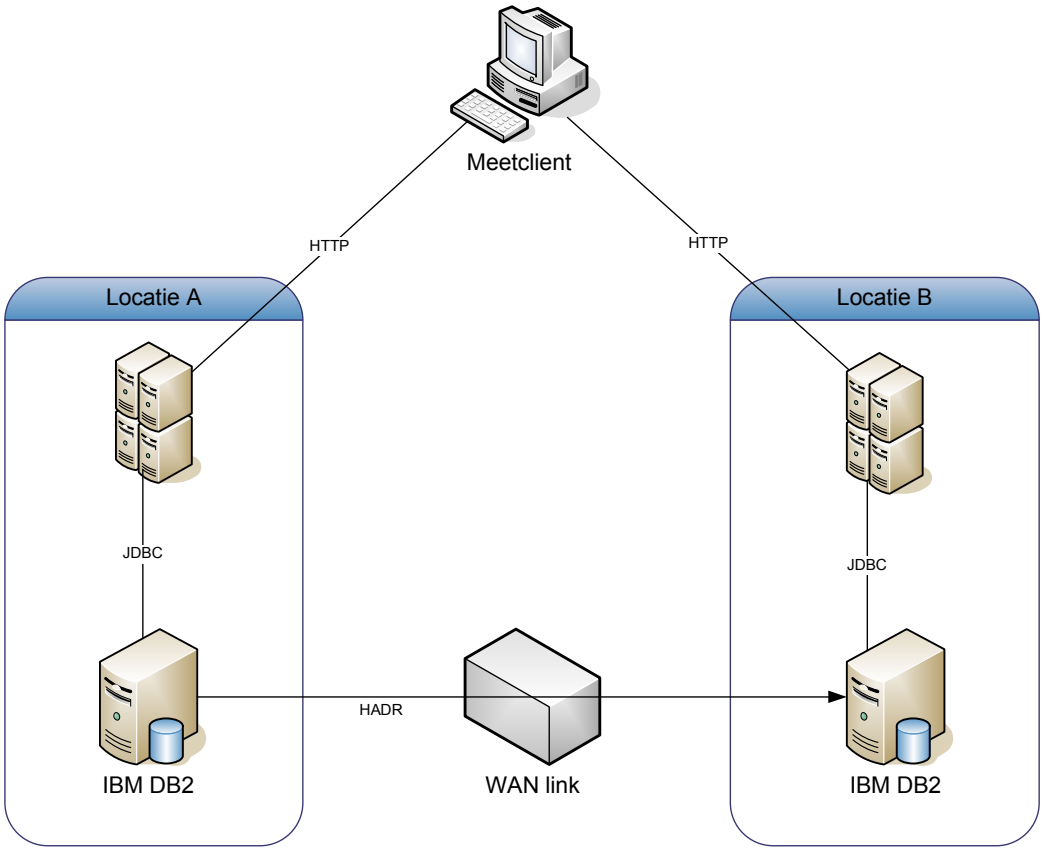
Sequoia is een middleware oplossing die universeel toepasbaar is op verschillende databases, omdat het op JDBC niveau werkt. Elke database met een JDBC driver kan door Sequoia gebruikt worden. Sequoia is open-source, maar er is ook een commerciële versie van beschikbaar via Continuent. Andere middleware systemen richten zich vooral op transactie afhandeling en niet op het High Availability aspect. Deze transactie systemen zijn meestal gebaseerd op een two-phase commit protocol, dat wel consistente data opslag biedt, maar bij uitval van een van de componenten niet meer beschikbaar is voor de applicatie. Daardoor zijn deze systemen niet interessant voor het te testen scenario. Sequoia probeert dit op te lossen door een groep communicatie systeem te gebruiken, waarbij uitval van een van de componenten betekent dat de groep met een partij minder communiceert.

Per product wordt in de volgende paragrafen de architectuur behandeld die van belang is voor de testopstelling, 6.5.1 behandelt IBM DB2 HADR, 6.5.2 Sequoia en MySQL Cluster wordt beschreven in 6.5.3.

6.5.1 Product 1 – IBM DB2 HADR

IBM DB2 HADR heeft op beide locaties één systeem nodig, waarop een IBM DB2 database draait. *Figuur 12: IBM DB2 HADR testopstelling* geeft de opstelling weer. Door middel van JDBC kan de database worden benaderd vanuit de applicatie server. Het HADR proces verzorgt de synchronisatie van de data tussen beide databases.

Hoewel niet getekend in figuur 12 is het mogelijk om meerdere applicatie servers op één database aan te sluiten, zodat deze geen single point of failure vormen. Bij het uitvallen van de database op locatie A is wel altijd een fail-over naar locatie B noodzakelijk, waardoor de WAN link erg belangrijk is.



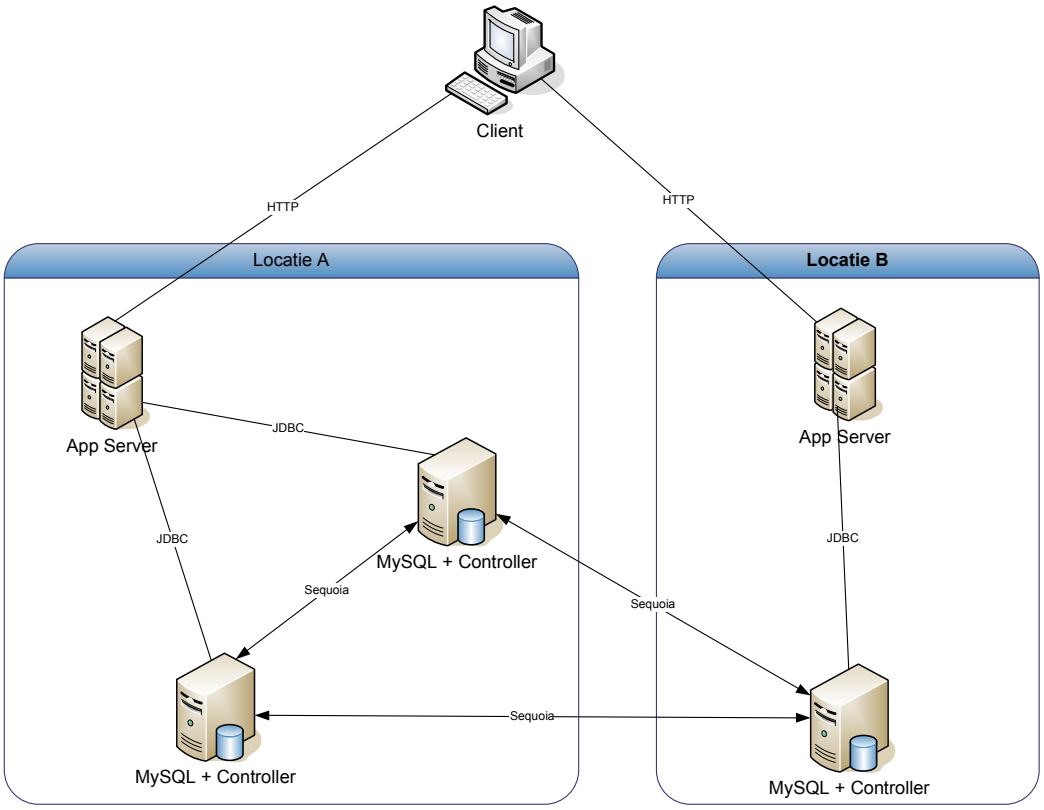
Figuur 12: IBM DB2 HADR testopstelling

6.5.2 Product 2 – Sequoia

Sequoia kan meerdere databases gebruiken als data backend, zolang deze JDBC als connectiemethode ondersteunen. Omdat MySQL gratis beschikbaar is, wordt MySQL in deze opstelling als backend gebruikt. Eventuele performance invloeden van de gekozen database backend zijn nihil vergeleken met de vertraging van de Sequoia software. Bij het inzetten van de Sequoia software in een productieomgeving kan nog een vergelijkende performancetest worden uitgevoerd tussen verschillende database backends.

Om Sequoia te kunnen draaien is een Java SDK nodig. Sequoia kan op meerdere platformen draaien, doordat alleen een Java Virtual Machine (JVM) nodig is. Deze JVM draait op elk besturingssysteem waar Java voor beschikbaar is. Sequoia kan daardoor zowel op Linux als op Windows functioneren.

Figuur 13: Sequoia opstelling toont de testopstelling van het Sequoia product. In deze testopstelling zullen de controller en de bijbehorende MySQL database op dezelfde machine actief zijn, dit is een minimum opstelling. Het gebruik van aparte machines heeft als voordeel dat niet zowel de controller als de database uitvallen bij uitval van een machine. Daarbij heeft de belasting van de controller dan geen invloed op de belasting van de database en omgekeerd. De consequenties van het gebruik van één machine zullen in de opstelling worden getest. Ook zal getest worden of beide locaties tegelijk gebruikt kunnen worden.



Figuur 13: Sequoia opstelling

6.5.3 Product 3 – MySQL Cluster

MySQL Cluster maakt gebruik van 3 verschillende nodes, die zijn weergegeven in *Figuur 14: MySQL Cluster opstelling.*

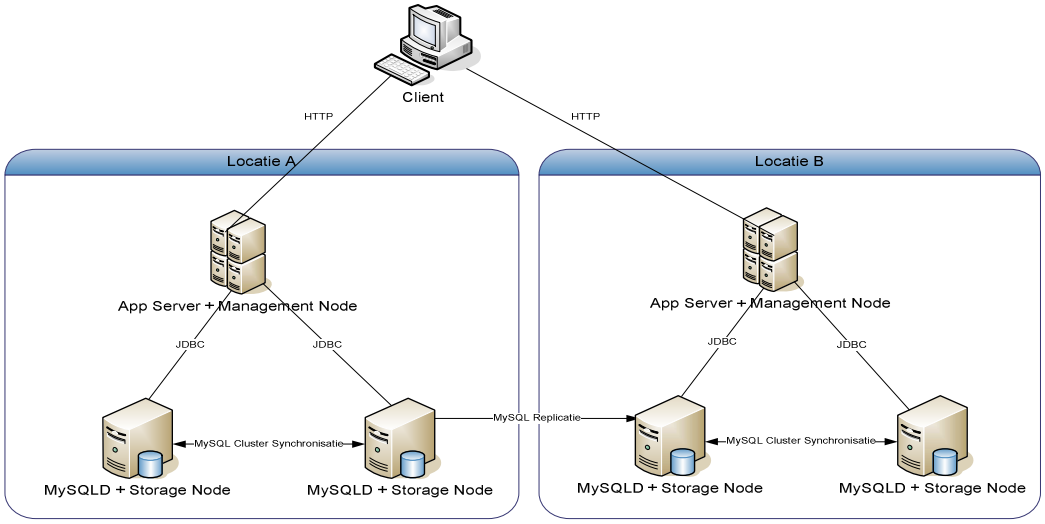
Een management node coördineert het opstarten van het cluster en speelt een rol bij falende nodes om het cluster te herconfigureren.

MySQLD nodes zijn normale server processen die in verbinding staan met de storage nodes. De MySQLD nodes kunnen ook data bevatten die niet in het cluster is ondergebracht, bijvoorbeeld statische data of data die specifieke functies nodig hebben die niet door de cluster engine ondersteund worden. Voor redundantie van deze data kan replicatie gebruikt worden.

De storage nodes verzorgen de opslag van de data en de onderlinge synchronisatie, zodat de data consistent blijft. Een minimale opstelling gebruikt 3 nodes, 1 management node en 2 storage nodes, waar op deze nodes ook de MySQLD processen draaien. Het aantal storage nodes hoeft niet hetzelfde te zijn als het aantal MySQLD nodes, waardoor onafhankelijk van elkaar geschaald kan worden als dat nodig is.

In de testopstelling draait de management node samen met de applicatieserver op een machine, zodat een machine uitgespaard kan worden. Afhankelijk van de benodigde performance kunnen de verschillende functies op verschillende machines worden gedraaid. In de testopstelling is voor de minimale opstelling gekozen.

De cluster naar cluster replicatie is een Master-Slave replicatie, waardoor slechts op een locatie geschreven kan worden door de client. Het cluster in locatie B fungeert als slave van het cluster in locatie A, totdat deze locatie of het cluster uitvalt. De client zal op locatie A schrijven, totdat een switch nodig is.



Figuur 14: MySQL Cluster opstelling

6.6 Stresstest

Om een uitspraak te kunnen doen over de robuustheid en de werking van de oplossingen bij uitval, terwijl er een zware belasting op het systeem is, zal een stresstest worden uitgevoerd. Er zijn verschillende mogelijkheden om een systeem op deze wijze te testen, door middel van een gesimuleerde of echte productieomgeving waarin het systeem gebruikt wordt of het gebruik van een benchmark suite. Omdat de tijd ontbreekt om een simulatie van de productieomgeving te bouwen en de productieomgeving zelf niet beschikbaar is, blijft het gebruiken van een benchmark over.

Een benchmark suite bevat verschillende simulaties die relevant kunnen zijn voor het gebruik van de software die getest wordt. Bij databases wordt veelal een onderscheid gemaakt tussen On-Line Transaction Processing (OLTP) en Decision Support Systems (DSS). Het eerste wordt gebruikt voor het verwerken van zoveel mogelijk transacties die real-time plaatsvinden en zijn zowel lees als schrijfintensief. De DSS systemen zijn veelal databases met zeer veel data waarop rapportages worden uitgevoerd, waarbij leesacties het grootste deel van de acties uitmaken. Belangrijk bij de benchmark suite is het kiezen van de correcte testcase die gemaakt is voor een van beide functies. Combinaties komen uiteraard ook voor tussen beide functies.

Hoewel een benchmark een inzicht kan geven in de robuustheid en performance van een database, hoeft de uitgevoerde testcase geen goed beeld te geven als deze niet overeenkomt met de werkelijke productieomgeving. Daarom is het belangrijk niet alleen resultaten van een benchmark te gebruiken, maar ook een test van de productieomgeving uit te voeren als overgegaan wordt tot het in gebruik nemen van het product.

De benchmark waar alle database producenten van gebruik maken om klanten te overtuigen van hun goede database systemen zijn de TPC-C en TPC-H benchmarks van de Transaction Processing Performance Counsil [104]. De TPC-C biedt een benchmark om OLTP prestaties te meten [105], de TPC-H is voor ad-hoc Decision Support prestatiemetingen [106]. Op welke manier deze benchmarks werken staat beschreven in documentatie die beschikbaar is op [104], maar er is geen kant-en-klaar pakket gemakkelijk beschikbaar dat gebruikt kan worden om een willekeurige database te testen. Wel beschikbaar zijn de verschillende Database Tests van het Open Source Development Labs [107]. Deze database tests van de OSDL zijn een open-source implementatie van de TPC-C en TPC-H benchmarks. Hoewel het implementaties zijn van beide benchmarks zijn ze niet goed vergelijkbaar, omdat niet aan alle eisen van de TPC kan worden voldaan. Helaas zijn deze implementaties alleen beschikbaar voor de twee open-source database producten PostgreSQL en MySQL en niet universeel toepasbaar, zodat gebruik voor de andere testproducten, Sequoia en DB2, niet mogelijk is. Dit betekent dat deze testsuite niet gebruikt kan worden voor deze stresstest.

Een test suite die wel universeel toepasbaar is door het gebruik van JDBC als database connectie is de Database Open-Source Test Suite (DOTS) [109], een onderdeel van de Linux Test Project [108]. Deze test suite is ontwikkeld door een aantal grote organisaties die de ontwikkeling van Linux wilden steunen.

Door de Linux Test Project test suite kunnen wijzigingen in de Linux kernel getest worden op verbetering of verslechtering van zowel performance als stabiliteit en robuustheid. De DOTS kan door middel van JDBC op elke database worden aangesloten die JDBC ondersteunt, wel dient het pakket op Linux te draaien om de belasting van het systeem te kunnen meten. Omdat in de testopstelling voor Linux als OS is gekozen en de drie oplossingen JDBC ondersteunen, is de DOTS goed geschikt om de producten aan een test te onderwerpen.

De twee interessantste testcases van de database open-source test suite zijn de ATCJ1 en ATCJ2. Beide zijn volgens de DOTS geavanceerde testcases en simuleren een bepaald scenario, in tegenstelling tot de base cases die slechts eenvoudige operaties uitvoeren. ATCJ1 simuleert een gebruiker registratie en authenticatie systeem, waarbij gebruikers kunnen worden toegevoegd en kunnen worden opgezocht. Het aanpassen van gebruikers wordt ook continue uitgevoerd. De ATCJ2 is een simulatie van een online veiling, waarbij continu producten worden toegevoegd en op producten wordt geboden. Ook het zoeken naar producten wordt hiermee gesimuleerd. Beide testcases leveren tabellen op waarvan de inhoud continu toeneemt, waardoor deze testcases als stresstest gebruikt kunnen worden. Omdat de ATCJ2 gebruik maakt van de gebruikerstabel die door ATCJ1 wordt gebruikt, zullen beide bij tegelijkertijd draaien invloed op elkaar uitoefenen, waardoor de resultaten niet goed vergelijkbaar zullen zijn.

De ATCJ2 benchmark is erg cliëntintensief, doordat veel verschillende berekeningen nodig zijn voordat een transactie gedaan kan worden. Hierdoor treedt de client al vrij snel als bottleneck op, waardoor resultaten niet goed meer vergelijkbaar zijn. De maximaal te behalen resultaten worden dan niet door de database server bepaald maar door de gebruikte client hardware. De belasting van het client systeem is bij de ATCJ1 benchmark zeer beperkt, waardoor de client niet als bottleneck optreedt. Hierdoor kunnen resultaten tussen producten beter worden vergeleken. De ATCJ1 benchmark zal dan ook worden gebruikt als stresstest.

De ATCJ1 benchmark voert drie verschillende acties uit, namelijk INSERT, UPDATE en QUERY. Door middel van een INSERT wordt een nieuwe gebruiker toegevoegd, met UPDATE worden gegevens van een gebruiker gewijzigd en de QUERY wordt gebruikt om gebruikersgegevens te controleren, zodat ingelogd kan worden. De transacties worden in willekeurige volgorde uitgevoerd, maar komen ongeveer evenveel voor. De ATCJ1 benchmark werkt slechts met één tabel, er wordt dus niet gekeken naar de performance van joins of stored procedures. Daarover kan dan ook geen uitspraak worden gedaan met behulp van de testresultaten. Omdat met de stresstest vooral gekeken wordt naar de invloed van de HA opties op de performance is dat minder relevant.

Om te bepalen welke invloed de verschillende High Availability opties hebben en welke invloed de WAN latency heeft, zullen deze door middel van een aantal 30 minuten testruns worden vergeleken. Het aantal transacties wat binnen 30 minuten kan worden uitgevoerd kan dan worden vergeleken, zodat een beeld

wordt verkregen van de relatieve performance van de opties binnen één product.

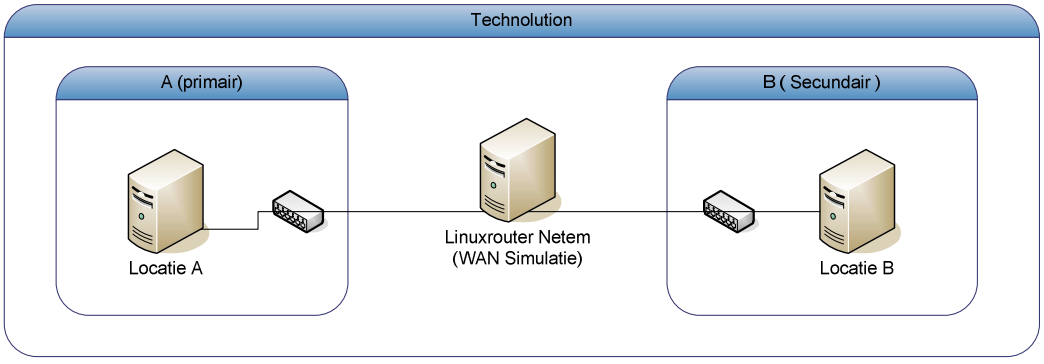
6.7 WAN Simulatie

De testopstelling moet een simulatie vormen van een systeem waarbij gebruik wordt gemaakt van twee locaties. Bij het gebruik van twee locaties die meerdere kilometers uit elkaar liggen, is een Wide Area Network benodigd (WAN). De eigenschappen van het WAN dienen te worden nagebootst om te zien welke invloed dit heeft op de verschillende producten. Omdat een echte WAN opstelling lastig te realiseren is, zal deze gesimuleerd moeten worden.

Voor deze testopstelling wordt de aanname gedaan dat de twee locaties niet verder dan 200km uit elkaar liggen. Het vaststellen van de latency die bij deze afstand hoort is erg lastig, omdat dat erg afhankelijk is van de gebruikte technologie en de vraag of een rechtstreekse verbinding mogelijk is. Over het algemeen is een directe verbinding over dergelijke afstanden niet mogelijk of niet rendabel genoeg. Als aanname wordt gedaan dat een latency van 50ms een redelijke weergave is van de latency op de WAN link. Ter vergelijking, een trans-Atlantische link vergt gemiddeld 100ms Round Trip Time. Om de invloed van latency te achterhalen zal ook een latency van 300ms worden gebruikt om te testen. Deze latency's zijn vooral relevant als data naar de andere kant van de wereld gerepliceerd moet worden.

De WAN link wordt gesimuleerd met een delay van 50 ms. De LAN latency wordt buiten beschouwing gelaten, omdat deze minder dan 1/50 van de gesimuleerde delay is (< 1ms).

Voor het simuleren van de verbinding kan een machine voorzien van een Linux besturingssysteem gebruikt worden. Deze machine krijgt een speciale kernel module waarmee bepaalde WAN eigenschappen ingesteld kunnen worden. Zo is het mogelijk om enkele of alle pakketten een vertraging te laten ondergaan, bepaalde pakketten zoek te laten raken en de bandbreedte van de verbinding in te stellen. Ook duplicatie van pakketten en het veranderen van de volgorde is mogelijk. Voor deze simulatie wordt gebruik gemaakt van de Netem functionaliteit in de 2.6 kernel [110]. *Figuur 15: WAN link opzet* geeft een overzicht van de WAN opstelling.



Figuur 15: WAN link opzet

6.8 Testargumentatie

Bij elk product dat in deze testopstelling getest zal worden is gekeken naar de opzet van de oplossing. Bij elk apart onderdeel binnen deze oplossing, zoals een controller bij Sequoia of een database node bij MySQL, is een testcase opgesteld die test welke gevolgen het uitvallen van dit onderdeel heeft. Tevens zijn beweringen van fabrikanten over het kunnen opvangen van bepaalde uitvalsituaties meegenomen in de testcases om te bepalen of deze beweringen correct zijn. Hoe complexer de oplossing, hoe meer componenten de oplossing bevat, waardoor meer testcases nodig zijn om zaken aan te kunnen tonen. Bij een complexe oplossing als Sequoia waarbij meerdere componenten gebruikt worden, zal dat betekenen dat er meerdere testcases nodig zijn in vergelijking met de relatief simpelere IBM oplossing.

In dit testplan zullen zowel testcases als onderzoekscases aan bod komen. Testcases zijn vragen waarop een duidelijk ja of nee geantwoord kan worden, bijvoorbeeld de vraag of het systeem succesvol blijft functioneren bij uitval van een van de componenten. Onderzoekscases worden gebruikt om zaken te onderzoeken en antwoorden op vragen te vinden die niet met een ja of nee beantwoord kunnen worden. Hierbij gaat het bijvoorbeeld om metingen van de invloed van de WAN latency en wat er nodig is om een systeem te herstellen als een bepaalde fout is opgetreden.

Het volledig testen van elke mogelijke situatie die in een productieopstelling voor zou kunnen komen is niet mogelijk, wel zijn de meest voorkomende problemen bij High Availability oplossingen door de testcases afgedekt. Bij deze problemen gaat het vooral om single point of failures, zowel op netwerk als op hardware en software gebied. Elk component wordt door een testcase aan een test onderworpen. De onderzoekscases behandelen de belangrijkste vragen met betrekking tot de producten, zoals de benodigde stappen bij herstel van uitval en de robuustheid van de producten. De verschillende metingen kunnen worden uitgebreid naar andere vlakken indien dat relevant is voor de productieomgeving waarin het product uiteindelijk komt te draaien.

Alle test- en onderzoekscases zijn opgenomen in Appendix D.

7. BESPREKING VAN TESTRESULTATEN

Een volledig overzicht van alle testcases met hun resultaten staat in Appendix E. Dit hoofdstuk vat de belangrijkste resultaten samen en maakt een vergelijking tussen de drie geteste producten in paragraaf 7.1. Paragraaf 7.2 beschrijft de resultaten van de uitgevoerde stresstest op de drie producten met hun verschillende HA features aan- en uitgeschakeld om hun invloed op de prestaties te bepalen. In paragraaf 7.3 wordt teruggekomen op de eisen die in paragraaf 6.3 aan de producten in de testopstelling werden gesteld. In deze paragraaf wordt antwoord gegeven op de vraag of de producten voldoen voor het project waar de testopstelling naar gesimuleerd is.

7.1 Testresultaten

Het IBM HADR systeem is een actief-passief master/slave replicatie systeem, waarbij het slave systeem niet gebruikt kan worden door clients. De replicatie biedt alleen redundantie van data aan, maar biedt verder geen verbetering van performance, doordat het slave systeem niet gebruikt kan worden. Er zijn dus geen schaalvoordelen met betrekking tot lees en schrijfacties, zoals deze er wel mogelijk zijn bij Sequoia en MySQL.

7.1.1 Automatische overschakeling

Hoewel de overschakeling van clients bij IBM automatisch kan worden afgehandeld door Automatic Client Reroute (ACR) dat in combinatie met HADR gebruikt en geconfigureerd kan worden, dient hiervoor wel op een specifieke manier een connectie vanuit de clientapplicatie te worden opgezet. Evenals bij Sequoia legt het gebruik van ACR restricties op aan de applicatie. ACR is vooral praktisch bij het gebruik van een zelf geïnitieerde take-over voor het uitvoeren van onderhoud aan een van de twee machines, doordat clients dan automatisch overgeschakeld kunnen worden. Sequoia biedt de functionaliteit van automatische client overschakeling wel standaard aan in hun JDBC driver, waarbij alleen de verschillende controllers in de connectiestring opgenomen hoeven worden. Bij MySQL Cluster is er wel automatische overschakeling tussen de SQL en data nodes, maar niet tussen SQL nodes onderling. Om vanaf de applicatie bij de SQL nodes te komen, is een extra load-balancer of virtueel IP-adres oplossing benodigd om te zorgen dat het cluster beschikbaar blijft, ondanks uitval van een dergelijke node. Het overschakelen tussen twee locaties werkt bij Sequoia automatisch door de multimaster opzet van de oplossing. Bij MySQL moet er een eigen oplossing worden gebouwd om de overschakeling naar een tweede locatie mogelijk te maken. Door middel van scripts of aanpassing van de applicatie kan het schakelen naar de backup locatie wel mogelijk worden gemaakt.

Het HADR systeem heeft geen monitoring functie in zich, waardoor er bij het uitvallen van de primaire database er geen automatische overschakeling plaatsvind naar de stand-by database. Volgens IBM is dit wel mogelijk met externe monitor software of behulp van zelfgemaakte scripts. In combinatie met een monitoring pakket en automatische overschakeling kan het ACR systeem een goede oplossing zijn voor een Bomb-Proof Server. Zo kan de tweede

locatie doordraaien als de eerste uitvalt, zonder dat gebruikers hier enig zicht op krijgen. Wel dient de applicatie dan ondersteuning te hebben voor het opnieuw uitvoeren van transacties en moet er bij de overschakeling naar de secundaire locatie bij uitval van de primaire database geen probleem optreden door bijvoorbeeld het uitvallen van de WAN link.

MySQL biedt met MySQL cluster een geclusterde database aan die 99.999% beschikbaarheid moet garanderen. Hoewel dit voor het cluster zelf misschien mogelijk is, betekent het helaas niet dat de volledige oplossing ook deze beschikbaarheid haalt. De overname van uitgevallen data nodes werkt zonder problemen, dit is volledig automatisch. De uitval van een SQL node, waarmee de applicatie of client een connectie maakt, levert echter wel uitval van functionaliteit op. De data is dan nog wel beschikbaar in het cluster, maar niet via de gebruikte SQL node. Een verbinding naar deze SQL node wordt niet automatisch overgeschakeld naar een nog actieve node. Dit betekent dat een load-balancer of een virtueel ip adres zal moeten worden toegevoegd om ervoor te zorgen dat connecties worden overgezet naar een andere SQL node. Hierin voorziet de MySQL Cluster oplossing niet, daarvoor is dus software of hardware van een derde partij noodzakelijk. Een tweede mogelijkheid is het inbouwen van deze functionaliteit in de applicatie, zodat deze automatisch overschakelt bij problemen, zoals dat door Sequoia of IBM's ACR wordt toegepast.

7.1.2 Synchronisatie

Bij HADR wordt automatische gesynchroniseerd nadat de WAN link is uitgevallen en weer hersteld is. Bij het uit en aanzetten van de standby database wordt echter niet automatische gesynchroniseerd. Het is mogelijk dat de time-out van de HADR synchronisatie een rol speelt of dat de database op een bepaalde manier weer aangezet moet worden, zodat dit wel automatisch gebeurt. Het is in ieder geval noodzakelijk te controleren of hersynchronisatie weer gestart is na herstel van het slave systeem. Synchronisatie treedt bij MySQL replicatie automatisch op als de verbinding herstelt is, bij uitval van de slave moet deze met een "Start slave" commando herstart worden. Bij Sequoia moet een handmatige actie worden uitgevoerd als beide locaties transacties verwerken, bij één locatie kan er in een aantal gevallen wel automatisch gesynchroniseerd worden.

7.1.3 Installatie en configuratie

De installatie en configuratie van de IBM DB2 database is door middel van een grafische schil vrij eenvoudig in te richten. Wel zijn de error messages niet altijd even duidelijk en is lastig na te gaan door welke setting een bepaalde error optreedt en of het opnieuw configureren een ander resultaat oplevert. Een groot nadeel van de grafische schil is de traagheid waarmee deze reageert op input en de grote wachttijden tijdens het uitvoeren van database commando's. Zo duurt het instellen van HADR, nadat tien schermen zijn doorgelopen, vaak meer dan 1,5 a 2 minuten. Sequoia biedt bij installatie geen grafische mogelijkheden en ook bij MySQL is het instellen prima mogelijk vanaf de commandline. Beide oplossingen zijn beduidend sneller te installeren en configureren dan de DB2 database. Ook de foutmeldingen zijn minder cryptisch en de handleiding is van goede kwaliteit. Vooral MySQL biedt een zeer goede handleiding aan op hun website, bij Sequoia wordt de documentatie bij elke release beter en bij IBM is

vooral het opzoeken problematisch door de grote hoeveelheid documentatie. Qua installatie bieden beide open-source projecten een duidelijk voordeel boven IBM doordat deze veel minder complex zijn. Wel biedt de DB2 database zeer veel opties voor het optimaliseren en tunen van de database, waar dan helaas wel bepaalde expertise voor nodig is.

7.1.4 Toepassing in productieomgeving

De geteste disk data optie in de beta 5.1.11 versie van MySQL Cluster werkt niet anders dan de in memory variant, wel is de 5.1 versie nog duidelijk in ontwikkeling. Er treden regelmatig fouten op waardoor het cluster wordt afgesloten of een van de nodes zichzelf afsluit. Versie 5.1 is op dit moment daarom nog niet geschikt voor het gebruik in een productieomgeving. Ook bij Sequoia volgen de ontwikkelingen zich in snel tempo op, in februari 2006 was er versie 2.5 en half juli is versie 2.9 uitgekomen. Hoewel er in elke versie meerdere fouten worden opgelost, betekent het dat er nog geen stabiele volledig goed werkende versie beschikbaar is. Ook is in versie 2.9 de stap gemaakt om de JGroups communicatie te gaan vervangen door Appia. Hoewel dit volgens Sequoia een verbetering van de performance moet geven, is het wachten op de resultaten. Waarschijnlijk zal vanaf versie 3.0 de software stabiliseren en goed bruikbaar worden in een productieomgeving.

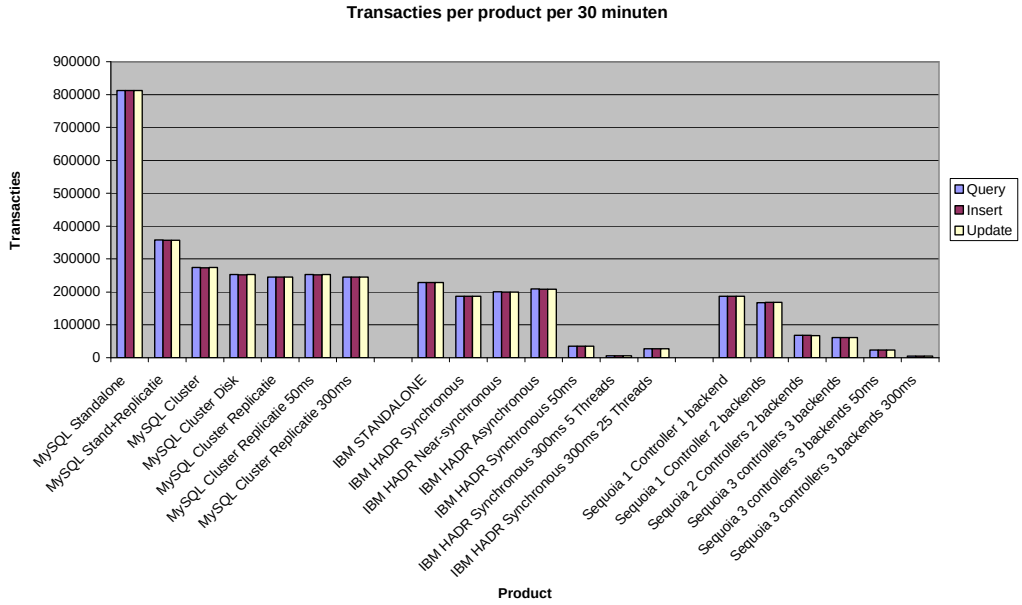
Een voordeel van de oplossing van MySQL boven die van IBM is de grotere redundantie. Bij IBM zal bij een hardware uitval op de primaire locatie altijd naar de secundaire locatie geschakeld moeten worden. Hierbij is de beschikbaarheid van de WAN link erg belangrijk, zodat deze overschakeling mogelijk is bij uitval en geen dataverlies optreedt. Bij MySQL kan de primaire locatie blijven functioneren bij uitval van één component, en is pas bij volledige clusteruitval een overschakeling naar de secundaire locatie nodig. In dat geval kan wel dataverlies optreden, doordat er tussen twee MySQL Clusters alleen asynchroon wordt gerepliceerd. Dit nadeel heeft IBM niet, zolang de afstand en benodigde performance synchrone replicatie toestaat.

Sequoia biedt als enige van de drie een Multi-Master systeem waarmee op beide locaties tegelijkertijd gewerkt kan worden. Dit heeft als grote voordeel dat geen overschakeling hoeft plaats te vinden zoals bij IBM of MySQL. De secundaire locatie kan zonder problemen worden gebruikt als de primaire uitvalt, daar is geen extra stap voor nodig. Wel brengt dit een groot risico met zich mee, namelijk de uitval van de verbinding tussen de twee locaties. Indien die verbinding uitvalt, blijven beide locaties apart van elkaar doordraaien, het zogenaamde “split-brain scenario”. Dat resulteert dan in een systeem dat niet meer consistent is als beide transacties blijven verwerken. Indien een van beide geen transacties krijgt te verwerken bij WAN uitval, is het synchroniseren geen probleem, omdat geen samenvoeging gemaakt hoeft te worden. Door Sequoia wordt een zogenaamde netwerk partitie en reconciliatie niet automatisch ondersteunt, hier dient een beheerder zelf handmatig een oplossing voor te bedenken. Bij het gebruik van Sequoia is de monitoring en beschikbaarheid van de WAN link dus erg belangrijk.

7.2 Stresstest

Voor de stresstest zijn een aantal verschillende productconfiguraties getest. Resultaat van deze testen staan weergegeven in *Figuur 16: Transacties per product per 30 minuten* en *Tabel 5: Stresstest resultaten*. In deze figuur zijn het aantal transacties per 30 minuten uitgezet tegen de verschillende producten met hun eigen features. Een transactie bevat ofwel een query, ofwel een insert of update. Alle testen zijn drie keer uitgevoerd waarna het gemiddelde is berekend.

Bij alle drie de producten zijn de standaardwaardes gehanteerd zoals ze na installatie zijn ingesteld. De producten zijn dus niet geoptimaliseerd of getuned voor een bepaalde taak. Dit kan en zal betekenen dat de producten bij optimalisaties beter kunnen presteren dan hier is weergegeven. Omdat het hoofddoel van de test het bepalen van de invloed van features binnen een product op de performance is en niet het vergelijken van de producten met elkaar, is deze werkwijze acceptabel.



Figuur 16: Transacties per product per 30 minuten

Tabel 5: Stresstest resultaten

	Query	Insert	Update
MySQL Standalone	812.049	811.996	812.275
MySQL Stand+Replicatie	357.705	357.470	356.974
MySQL Cluster	273.935	273.718	274.632
MySQL Cluster Disk	252.625	251.997	252.786
MySQL Cluster Replicatie	245.428	245.117	245.356
MySQL Cluster Replicatie 50ms	252.796	252.420	253.102
MySQL Cluster Replicatie 300ms	245.395	245.140	245.010

IBM STANDALONE	228.880	229.143	228.616
IBM HADR Synchronous	186.923	186.665	187.250
IBM HADR Near-synchronous	200.127	199.884	199.868
IBM HADR Asynchronous	209.198	208.569	208.848
IBM HADR Synchronous 50ms	34.862	34.939	35.237
IBM HADR Synchronous 300ms 5 Threads	6.288	6.133	6.214
IBM HADR Synchronous 300ms 25 Threads	27.279	27.169	27.409
Sequoia 1 Controller 1 backend	186.907	186.831	187.016
Sequoia 1 Controller 2 backends	167.563	168.257	168.053
Sequoia 2 Controllers 2 backends	6.7860	67.625	67.268
Sequoia 3 controllers 3 backends	60.997	60.706	60.787
Sequoia 3 controllers 3 backends 50ms	23.150	23.019	23.258
Sequoia 3 controllers 3 backends 300ms	4.889	4.970	5.024

Bij zowel Sequoia als IBM DB2 leidt het gebruik van een synchrone techniek tot een afname van het aantal uit te kunnen voeren transacties per 30 minuten. Omdat MySQL geen synchrone replicatie heeft, ontbreekt deze dip bij MySQL. Wel is een duidelijk verschil te zien tussen de standalone MySQL server en het gebruik van MySQL Cluster. Dit is ook te verklaren, omdat het gebruik van het netwerk een langere response tijd oplevert dan als lokaal data moet worden opgehaald of weggeschreven. Omdat de stresstest van deze response tijd afhankelijk is, kunnen er minder transacties worden uitgevoerd in de 30 minuten periode.

Opvallend is het verschil tussen MySQL Standalone met en zonder replicatie, hier gaat het om een afname van 450.000 transacties per 30 minuten, ongeveer 55%. Het verschil tussen MySQL Cluster met en zonder replicatie is beduidend kleiner. Hoewel de MySQL replicatie asynchroon verloopt, is het opslaan van de transactie in de binaire log dusdanig vertragend dat minder dan de helft van de standalone transacties uitgevoerd kan worden. Bij IBM is een dergelijk verschil tussen Standalone en Asynchroon slechts 20.000, ongeveer 10% van het totaal aantal transacties dat in standalone mode gehaald kan worden. Bij de synchrone modus scheelt het 45.000, ongeveer 25%. Dat asynchrone replicatie dus geen invloed uitoefent op de performance is niet waar, het is alleen minder dan de synchrone modus en het ondervindt geen invloed van hogere latency's, zoals dat bij MysQL cluster replicatie met 50 en 300 ms te zien is. Deze leveren dezelfde performance als over een lokaal netwerk met maximaal 1 ms vertraging, waarmee is aangetoond dat de latency bij het gebruik van asynchrone replicatie niet uitmaakt voor het aantal te kunnen verwerken transacties.

De invloed van zowel 50 als 300ms op een synchrone replicatie is duidelijk te zien aan zowel IBM als Sequoia. Bij IBM kost een latency van 50ms op een WAN link 85% (35.000 tov 230.000) van het aantal uit te voeren transacties. De 6000 transacties bij 300ms is zelfs een reductie van 98%. Om erachter te komen of met parallelisme dit opgehoogd kan worden, is met 25 threads gemeten. Daarmee komt het aantal transacties op het niveau van 50ms latency met 5 threads. Voor applicaties waarbij de transacties niet serieel afgehandeld hoeven te worden is een grotere latency dus minder beperkend, dan als op elke

transactie gewacht moet worden. Bij dergelijke latency's is het optimaliseren van parallel gedrag dus zeer de moeite waard, waarbij dan wel de bandbreedte van de verbinding toereikend moet zijn voor de parallel lopende processen.

Sequoia biedt een iets lagere performance vergeleken met IBM. Dit komt doordat de middleware JDBC laag een extra vertraging toevoegt en geen performance optimalisaties van de gebruikte MySQL database kan toepassen. Ook hier is duidelijk de synchrone techniek te zien aan het verschil tussen 1 en 2 controllers en tussen 3 controllers met en zonder latency's. Bij het gebruik van slechts 1 controller is het gebruik van het groep communicatie protocol niet nodig, zodat de netwerk latency's en het protocol geen vertraging opleveren. Het verschil tussen 1 controller met 2 backends en 2 controllers met elk een backend geeft een vermindering van 100.000 transacties. Helaas levert een oplossing met 1 controller geen volledige beschikbaarheid op, omdat als de controller uitvalt het volledige systeem niet meer beschikbaar is. Het minimum van 2 controllers is daarvoor wel noodzakelijk.

Concluderend betekent het gebruik van een synchrone techniek dat ofwel de afstand tussen beide datacentra zeer beperkt moet zijn of dat de applicatie met een relatief beperkte performance correct moet kunnen functioneren. De 200.000 transacties *per maand* uit het project waarnaar de testopstelling gemodelleerd is, kunnen met de 6000 transacties per 30 minuten op een 300ms latency nog prima uit de voeten indien deze vrij constant over de maand verspreid zijn. Echter een applicatie met meer dan 200.000 transacties *per dag* loopt dan al vrij snel tegen de maximale transacties aan van 288.000 (6000x2x24) per dag bij gebruik van Sequoia met een 300ms latency.

7.3 Productevaluatie

Terugkomend op de 7 gestelde eisen in paragraaf 6.3 van de testopstelling blijken alle drie de producten tekort te komen. Per eis wordt hier elk product behandeld.

Eis 1: Het uitvallen van een enkel systeem verantwoordelijk voor de database of de applicatie server, heeft geen gevolgen voor het doordraaien van de applicaties. Zowel de alarmverwerking als de rapportering moet blijven functioneren.
Op dit gebied doet alleen Sequoia het volledig goed. Hierbij heeft de uitval van een van de componenten geen gevolgen. Bij MySQL dient voor de opvang van de uitval van de SQL node een extra load balancer te worden ingezet. Bij IBM is een externe monitor nodig die de HADR database om kan schakelen als deze uitvalt. Doordat IBM geen redundantie op de primaire locatie heeft, is de WAN verbinding hier heel belangrijk.

Eis 2: Het uitvallen van de WAN link tussen locatie A en locatie B mag geen gevolgen hebben voor het doordraaien van de applicaties. Alarmen kunnen worden verwerkt en gerapporteerd en de secundaire locatie moet worden gesynchroniseerd bij herstel van de link.
Bij alle drie de producten blijft de primaire locatie doordraaien als de WAN link uitvalt. Zowel MySQL als IBM kunnen automatisch hersynchroniseren bij herstel van deze link. Bij Sequoia dient dit handmatig te gebeuren, omdat beide locaties doordraaien. Hierdoor kan een inconsistente state tussen beide optreden, zoals al aangegeven in paragraaf 7.1.

Eis 3: Bij het uitvallen van de primaire locatie mag geen data verloren gaan. Metingen die door het systeem aangenomen zijn en verwerkt worden of al afgehandeld zijn, dienen bewaard te blijven. Nog niet aangenomen metingen dienen door de secundaire locatie te worden aangenomen bij uitval van de primaire locatie.
(Recovery Point Objective (RPO) = 0)
Zowel IBM als Sequoia bieden door de synchrone replicatie hier een oplossing dat geen dataverlies tot gevolg heeft. Wel moet dan bij IBM voor de Synchronous mode gekozen worden en dient de Standby server overgeschakeld te worden naar primary voordat de applicaties deze weer kunnen gebruiken. Sequoia opereert in multimaster setting en heeft dus geen overschakeling nodig. MySQL biedt door zijn asynchrone replicatie geen garantie op geen dataverlies en kan dus niet aan deze eis voldoen.

Eis 4: Alleen bij het uitvallen van de volledige primaire locatie mag de client handmatig worden omgezet naar het secundaire systeem.
Bij Sequoia is deze eis niet nodig, hier wordt automatisch overgeschakeld. BIj IBM kan met ACR ook automatisch worden overgeschakeld, mits op goede wijze geconfigureerd in de applicatie. Bij MySQL dient dit handmatig te gebeuren of door middel van een zelf geschreven script of applicatieuitbreiding.

Eis 5: Bij het uitvallen van een enkel systeem op de primaire locatie moet de client zonder problemen door kunnen werken zonder dat de client hiervoor actie moet ondernemen.
Aan deze eis voldoen alle drie de systemen, mits bij HADR het ACR goed is ingesteld. Zowel MySQL Cluster als Sequoia bieden standaard overschakeling aan op dezelfde locatie, waarbij Cluster een load balancer nodig heeft als het om een SQL node gaat.

Eis 6: Bij het uitvallen van de primaire locatie is het acceptabel als de secundaire locatie binnen 4 uur online is. (Recovery Time Objective (RTO) < 4 uur). Minder dan 30 minuten is wenselijk.
Sequoia is door de multimaster setting standaard online, en hier is binnen 4 uur dus geen probleem. Bij MySQL Cluster hoeft ook geen overschakeling plaats te vinden op het secundaire cluster, zodat ook deze aan de eis voldoet. Bij IBM is de overschakeling afhankelijk van menselijke handelen (of een externe monitor), hier dient dan binnen 4 uur een persoon de overschakeling te initiëren.

Eis 7: Een component dat uitgevallen is, dient binnen 8 uur gerepareerd te kunnen worden. Omdat het systeem slechts één failure tegelijkertijd aankan, is het na het uitvallen van één component noodzakelijk om deze component zo snel mogelijk te repareren. Het optreden van een tweede failure voordat de eerste is opgelost, zal slechts in beperkte gevallen niet tot downtime leiden en dient voorkomen te worden. Procedures en processen dienen op deze tijden te worden afgestemd door middel van service contracten of reserve componenten.
Voor deze eis brengen alle drie de producten hetzelfde, hier is het de procedures en processen die goed op elkaar afgestemd moeten worden, zodat de beheerders correct kunnen handelen en de systemen zo snel mogelijk weer in de lucht kunnen krijgen. Het opnieuw installeren zal bij zowel Sequoia als MySQL sneller gebeurd zijn dan bij IBM, door de complexiteit en traagheid bij installatie en configuratie van de laatste.

Concluderend biedt Sequoia de beste papieren door de synchrone replicatie en de redundantie op de primaire locatie. Wel is het van belang de WAN link goed te monitoren, zodat het uit elkaar lopen van de primaire en secundaire locatie bij uitvallen van deze link voorkomen kan worden. Indien de performance meer moet zijn dan Sequoia aanbiedt is het noodzakelijk de latency en dus de afstand tussen beide locaties te verkleinen. Als dataverlies bij uitval van een volledige locatie geen harde eis van nul transacties is, dan biedt MySQL met haar cluster oplossing en asynchrone replicatie een goede beschikbaarheidsoplossing met voldoende performance. Ook hoge latency's kunnen dan prima worden doorstaan, waardoor lange afstanden tussen de datacenter locaties mogelijk worden. IBM biedt door het ontbreken van redundantie op de primaire locatie eigenlijk geen goede oplossing voor dit project, doordat uitval van de master altijd een overschakeling naar de secundaire locatie betekent. Dat houdt in dat voor deze overschakeling de WAN link wel altijd beschikbaar moet zijn als het systeem uitvalt. Dat is bij de andere twee alleen van belang als de volledige eerste locatie uitvalt, waardoor deze een betere oplossing bieden.

8. CONCLUSIES EN AANBEVELINGEN

Binnen de High Availability markt is er helaas niet één techniek met een “one solution fits all” oplossing. Bij elk project zal een afweging moeten worden gemaakt van het belang van consistentie en dataverlies versus het belang van performance. Synchrone replicatie brengt tenslotte altijd performance vermindering met zich mee in ruil voor consistentie aan beide zijden van het replicatiekanaal.

Projecten waarbij het risico van een compleet datacenter uitval genomen kan worden, hebben de keuzemogelijkheid tussen performance en consistentie, bij projecten waar minimaal twee datacenters nodig zijn, wordt deze keuze beperkt door de afstand tussen beide locaties. Hoe groter de afstand, hoe hoger de latency en hoe minder performance er overblijft bij gebruik van een synchrone techniek.

Een tweede probleem waar bij het gebruik van twee locaties tegenaan wordt gelopen is het split brain scenario, zoals ook bij Sequoia optreedt indien de WAN link uitvalt. Beide issues maken het lastig om een bomb-proof opstelling op te zetten, maar niet onmogelijk. Met extra monitoring, het inbouwen van extra checks in de applicatie, het paralliseren van transacties en het beperken van de afstand tussen beide locaties is voor veel projecten synchrone replicatie een mogelijkheid. Projecten waarbij performance erg belangrijk is, zullen genoegen moeten nemen met een asynchrone replicatietechniek, eventueel in combinatie met een multihop oplossing zoals door Interpay is toegepast.

Van de geteste producten biedt elk product zijn eigen voor en nadelen waardoor het voor het ene probleem beter geschikt is dan voor de ander. De gewenste beschikbaarheid en de geeiste voorwaarden bepalen dan ook in grote mate welk product het beste geschikt is.

Een probleem dat door veel producten niet goed opgelost kan worden is het overschakelen van client verbindingen naar andere locaties. Bij een verbinding op eenzelfde locatie is dat meestal nog mogelijk, maar indien die hele locatie uitvalt, zijn geen goede technieken voor handen om de clients automatisch de secundaire locatie te laten gebruiken. Veel producten regelen de beschikbaarheid dan ook alleen voor hun eigen domein waarvoor ze verantwoordelijk zijn, bijvoorbeeld alleen het database niveau en laten de rest aan externe soft- of hardware over. Dit betekent dat je geen kant-en-klaar product kan aanschaffen, maar als organisatie zelf de verschillende componenten moet integreren in een oplossing.

Bij het opzetten van een High Availability oplossing is het erg belangrijk te bepalen welke componenten een onderdeel vormen van het systeem en wat de gevolgen zijn als ze uitvallen. De oplossing moet voor alle niveau’s gelden voor zover technisch en financieel mogelijk. Het installeren van een cluster oplossing waarbij alles op één netwerk switch zit, levert een complete uitval op indien de switch het begeeft. De beschikbaarheid dient dus volledig te worden ingericht. Hetzelfde geldt voor het gebruik van twee locaties: indien naar de tweede locatie wordt overgeschakeld, doordat de eerste uitvalt, dient de tweede locatie

ook een backup te hebben in een derde locatie of de garantie dat de eerste binnen een bepaalde tijd weer online is. Anders valt de oplossing alsnog als ook de tweede locatie het begeeft.

Concluderend betekent dit dat de organisatie zelf verantwoordelijk is voor de aanschaf en integratie van producten en deze op alle niveau's van de keten moet inzetten. Een High Availability oplossing alleen in software of hardware lost het probleem niet op, door optredende fouten in de communicatiesystemen of het uitvallen van de elektriciteit in het datacentrum. Behalve de inrichting van alle niveau's voor beschikbaarheid is ook het beheren en monitoren van deze niveau's heel belangrijk, zodat zo tijdig mogelijk bij fouten kan worden ingegrepen. Hoewel heel handig bieden automatische systemen helaas geen garantie dat het altijd correct werkt en is menselijk handelen in veel gevallen nog een noodzaak. Om menselijke fouten zo veel mogelijk uit te sluiten zijn goed afgesproken procedures en processen, evenals het testen en trainen van medewerkers zeer belangrijk en een aanbeveling voor toekomstige HA projecten.

9. REFERENTIES

[1] Roosenboom, C., Altijd tot uw dienst,
Manieren om beschikbaarheid van systemen en diensten te verhogen,
Computable 26 april 2002,
http://www.computable.nl/artikels/archief2/d17ra2ol.htm,
Bekeken op 21 feb 2006

[2] Chao,T en Philips,J., High Availability Overview: Supplement Feature,
DM Review June 2002
http://www.dmreview.com/article_sub.cfm?articleId=5299
Bekeken op 21 feb 2006

[3] Muijzer, B., Inleiding High Availability, Sun Microsystems Nederland,
http://www.nluug.nl/events/vj01/papers/muyzer.pdf
UNIX en High Availability, *Voorjaarsconferentie 2001 NLUUG*
Bekeken op 2 feb 2006

[4] Kim, W., Highly Available Systems for Database Applications,
IBM Research Laboratory, p76
Computing Surveys, Vol 16, No1, March 1984 ACM

[5] Gray,J. , Helland P., O'Neil,P., Dennis, S., The Dangers of Replication and a Solution, SIGMOD Conf. 1996: pp.173-182 MSR-TR-96-17

[6] Otey, M., Otey D., Choosing a database for High Availability: An analysis of SQL Server and Oracle, April 2005
http://www.microsoft.com/sql/prodinfo/compare/oracle/dbavailability.mspx

[7] Otey,M., Otey,D. , Comparing the High Availability Features of DB2 UDB 8.2 and SQL Server 2005, April 2005,
http://download.microsoft.com/download/6/2/b/62b99ab9-c951-4fe9-a01d-3f6788412485/ComparingHighAvailabilityDB2%20UDB%208.2andSS2005.doc http://www.microsoft.com/sql/prodinfo/compare/ibm/default.mspx

[8] IBM Software Group, Toronto Laboratory, Database Availability Comparison, December 2005
ftp://ftp.software.ibm.com/software/data/pubs/papers/dbcomparison.pdf

[9] Dell Power Solutions, Exploring High-Availability Features in Microsoft SQL Server 2005, Februari 2006
http://www.dell.com/downloads/global/power/ps1q06.pdf

[10] IBM Software Group, Toronto Laboratory, Technical Comparison of DB2 HADR and Oracle Data Guard, Maart 2005
ftp://ftp.software.ibm.com/software/data/pubs/papers/hadr-comp.pdf

[11] MySQL Cluster Reference Manual,
http://dev.mysql.com/doc/refman/5.0/en/ndbcluster.html,

MySQL Cluster documentatie
http://www.mysql.com/products/database/cluster/, Bezocht op 17 juli 2006

[12] MySQL Cluster Frequently Asked Questions
http://dev.mysql.com/doc/refman/5.0/en/mysql-cluster-faq.html, Bezocht op 17 juli 2006

[13] MySQL 5.1 Development Roadmap for MySQL Cluster
http://dev.mysql.com/doc/refman/5.0/en/mysql-5-1-cluster-roadmap.html, Bezocht op 17 juli 2006

[14] *IEEE Task Force on Cluster Computing. High Availability*
http://www.ieeetfcc.org/high-availability.html, Bezocht op 17 juli 2006

[15] WhatIs Definition van High Availability
http://searchcio.techtarget.com/sDefinition/0,,sid19_gci761219,00.html, Bezocht op 17 juli 2006

[16] Bitpipe High Availability Definitie
http://www.bitpipe.com/tlist/High-Availability.html, Bezocht op 17 juli 2006

[17] Scheffel P.,Johann F., Hoed u voor betekenisloze metrics, SLA's: aanbieders hebben nog veel te leren. it service magazine nr4, mei 2003
http://www.vka.nl/files/SLA-metrics.pdf

[18] Dhondy, N., Petersen, D., Raften, D. , GDPS: The e-business Availability Solution, IBM Whitepaper
ftp://ftp.software.ibm.com/common/ssi/rep_wh/n/ZSW01781USEN/ZSW01781USEN.PDF

[19] Patterson, D. , A Simple Way to Estimate the Cost of Downtime, Computer Science Division, University of California at Berkeley, Extended abstract for LISA Conference 2002,
http://roc.cs.berkeley.edu/papers/Cost_Downtime_LISA.pdf

[20] Ronstrom, M. , Thalmann, L. , MySQL Cluster Architecture Overview, High Availability features of MySQL Cluster, MySQL white paper, april 2004 ,
http://www.mysql.com/why-mysql/white-papers/cluster.php

[21] Stratus Technologies Fault Tolerant Server providing 99.999% + uptime guarantee, http://www.stratus.com/products/index.htm,
Bezocht 6 april 2006

[22] Marathon Continuous Availability , Marathon FTvirtual server
http://www.marathontechnologies.com/Continuous_Avail.html, Bezocht op 17 juli 2006

[23] Kembel, R., "Fibre Channel, A Comprehensive Introduction", Northwest Learning Associates Inc., 2000, ISBN 0-931836-840.

[24] PGCluster project, multi-master synchronous replication for PostgreSQL, http://pgcluster.projects.postgresql.org/index.html, Bezocht op 7 april 2006

[25] Sherry, G. , Slony-II: An Innovative Approach to Multimaster Replication for PostgreSQL, http://www.slony2.org/slony2_opendbcon.pdf, Bezocht op 7 april 2006

[26] Database Replication projects, Bettina Kemme, University of McGill, Canada http://www.cs.mcgill.ca/%7Ekemme/disl/replication.html, Bezocht op 7 april 2006

[27] Kemme, B. Alonso, G., Don't be lazy, be consistent: Postgres-R, a new way to implement Database Replication,
Proc. of the 26th International Conference on Very Large Databases (VLDB), Cairo, Egypt, September 2000.
http://www.cs.mcgill.ca/~kemme/papers/vldb00.pdf

[28] Sequoia, JDBC middleware, http://sequoia.continuent.org/HomePage , Bezocht op 17 juli 2006

[29] HA-JDBC, JDBC middleware, http://ha-jdbc.sourceforge.net/, Bezocht op 17 juli 2006

[30] Continuent.com, m/cluster middleware voor MySQL, http://www.continuent.com/index.php?option=com_content&task=view&id=211&Itemid=168, Bezocht op 17 juli 2006

[31] Microsoft SQL Server 2005: SQL Server 2005 Features Comparison, 7 November 2005, http://www.microsoft.com/sql/prodinfo/features/compare-features.mspx

[32] Microsoft SQL Server 2005: How to Buy , 7 November 2005, http://www.microsoft.com/sql/howtobuy/default.mspx#EUH, Bezocht op 17 juli 2006

[33] Clark, L. , Technical Introduction to Oracle Fail Safe , A guide to Concepts and Terminology, October 2001,
http://www.oracle.com/technology/tech/windows/failsafe/pdf/fisc32.pdf

[34] Kumar, S. Oracle Database 10g Release 2 , High Availability, Mei 2005, http://www.oracle.com/technology/deploy/availability/pdf/TWP_HA_10gR2_HA_Overview.pdf

[35] Ray A., Meeks J., Oracle Data Guard in Oracle Database 10g Release 2, Business Continuity for the Enterprise, Mei 2005,
http://www.oracle.com/technology/deploy/availability/pdf/TWP_DataGuard_10gR2.pdf

[36] Rabobank Relies on HP NonStop systems for Internet Banking, Success Story, 2004
http://h71028.www7.hp.com/ERC/downloads/uploads_casestudy_Rabobank.pdf

[37] Coda File System, Carnegie Mellon University,
http://www.coda.cs.cmu.edu/

[38] GFS: Global File System, Red Hat,
http://www.redhat.com/software/rha/gfs/

[39] LinBit, DRBD support organisatie, http://www.linbit.com/en/drbd/drbd/00/

[40] DRBD Open Source, http://www.drbd.org

[41] Amir, Y., Danilov,C. Amir, M. e.a , On the Performance of Wide-Area Synchronous Database Replication, November 18, 2002

[42] Continuent m/cluster, High Availability, Scalability and manageability for MySQL, Overview Whitepaper,
http://www.continuent.com/index.php?option=com_content&task=view&id=34&Itemid=55, Bezocht op 19 juli 2006

[43] Continuent, http://www.continuent.com

[44] Sequoia project, http://sequoia.continuent.org/HomePage

[45] C-JDBC, de voorganger van Sequoia, http://c-jdbc.objectweb.org/

[46] Daly, K., Mean Time Between Flareups, er, Failures, 15 juni 2004
http://www.faqs.org/faqs/arch-storage/part2/section-151.html, Bezocht op 17 juli 2006

[47] Two Phase Commit protocol, Wikipedia
http://en.wikipedia.org/wiki/Two-phase_commit,
Bezocht 2 juni 2006

[48] Gray, J. Paxos Commit, a consensus protocol
http://research.microsoft.com/~Gray/talks/PaxosCommit_Techfest.ppt
Bezocht 2 juni 2006

[49] Multi phase commit protocols,
http://www.cse.ogi.edu/class/cse515/lectures/11%20-%202PC%20and%203PC.pdf
Bezocht 2 juni 2006

[50] Keating, B. , DB Specialists, Challenges involved in Multi-master Replication, 2001
http://www.dbspecialists.com/presentations/mm_replication.html
Bezocht 2 juni 2006

[51] Burleson, D. , dba-oracle.com, A Four-phase Approach to Procedural Multi-master Replication
http://www.dba-oracle.com/art_dbazine_mm_repl.htm
Bezocht 2 juni 2006

[52] Oracle Transparent Application Failover
http://www.oracle.com/technology/deploy/availability/htdocs/taf.html
Bezocht 6 juni 2006

[53] Transparent Application Failover, an Oracle White Paper, Maart 2006
http://www.oracle.com/technology/tech/oci/pdf/taf_10.2.pdf

[54] Zikopoulos, P. , The IBM DB2 Version 8.2 Automatic Client Reroute Facility, 22 december 2005 http://www-128.ibm.com/developerworks/db2/library/techarticle/dm-0512zikopoulos/, Bezocht op 17 juli 2006

[55] Windows Server 2003 : Microsoft Clustering Services, http://www.microsoft.com/windowsserver2003/technologies/clustering/default.mspx , bezocht 8 juni 2006

[56] Oracle Real Application Clusters
http://www.oracle.com/database/rac_home.html, Bezocht op 17 juli 2006

[57] Oracle Real Application Clusters 10g, An Oracle Technical Whitepaper, May 2005,
http://www.oracle.com/technology/products/database/clustering/pdf/TWP_RAC10gR2.pdf

[58] Talmage, R., Log Shipping in SQL Server 2000, part 1, 11 april 2002
http://www.microsoft.com/technet/prodtechnol/sql/2000/maintain/logship1.mspx

[59] PostgreSQL: The world's most advanced open source database
http://www.postgresql.org/

[60] Slony-I, http://gborg.postgresql.org/project/slony1/projdisplay.php

[61] JGroups, A toolkit for reliable multicast communication,
http://www.jgroups.org

[62] Appia, Group communication toolkit, http://appia.di.fc.ul.pt/

[63] Linux Virtual Server, http://www.linuxvirtualserver.org/

[64] Linux-HA, The High Availability Linux project, http://www.linux-ha.org/

[65] Linux-HA, Success Stories, http://www.linux-ha.org/SuccessStories, Bezocht op 17 juli 2006

[66] Microsoft Windows Server 2003, Cluster Service http://www.microsoft.com/windowsserver2003/techinfo/overview/bdmtdm/default.mspx, Bezocht op 17 juli 2006

[67] Red Hat Cluster Suite, http://www.redhat.com/software/rha/cluster/, Bezocht op 17 juli 2006

[68] Delevering High Availability Solutions with Red Hat Cluster Suite, Sept 2003, http://www.redhat.com/whitepapers/rha/RHA_ClusterSuiteWPPDF.pdf

[69] Veritas: Evolution of High Availability Technologies, 30 april 2003, http://www.veritas.com/van/articles/2943.jsp#2, Bezocht op 17 juli 2006

[70] HP NonStop Computing, http://h20223.www2.hp.com/NonStopComputing/cache/76385-0-0-0-121.html, bezocht op 5 juli 2006

[71] Wikipedia : Reliability, Availability and Servicability http://en.wikipedia.org/wiki/Reliability,_Availability_and_Serviceability, bezocht op 5 juli 2006

[72] Veritas : Basics of Failover Management in Application Clustering Packages, 7 juli 2003, http://www.veritas.com/van/articles/3245.jsp, bezocht op 5 juli 2006

[73] Aspen Systems, The history behind Beowulf Clusters, The Era Of Supercomputing, http://www.aspsys.com/clusters/history/, Bezocht op 5 juli 2006

[74] JAVA SE Technologies- Java DataBase Connectivity (JDBC) http://java.sun.com/javase/technologies/database.jsp, Bezocht op 6 juli 2006

[75] Java Transaction API Specification, http://java.sun.com/products/jta/, Bezocht op 6 juli 2006

[76] Buyya, R. , Architecture Alternatives for Single System Image Clusters, 1999, http://www.buyya.com/papers/ssiArch.html , Bezocht op 6 juli 2006

[77] The Globus Alliance, http://www.globus.org/, Bezocht op 6 juli 2006

[78] Foster, I., What is the Grid? A three point checklist. 20 juli 2002, http://www-fp.mcs.anl.gov/~foster/Articles/WhatIsTheGrid.pdf

[79] Wikipedia : Gird Computing, http://en.wikipedia.org/wiki/Grid_computing, Bezocht op 6 juli 2006

[80] Broadwell, P. , Component Failure Prediction using Supervised Naïve Bayes Classification, 16 december 2002, http://www.cs.berkeley.edu/~pbwell/papers/bayesian.pdf

[81] Vargas, E., High Availability Fundamentals, Sun Blueprints Online, November 2000 http://www.sun.com/blueprints/1100/HAFund.pdf

[82] Kuball,S. ,May, J. , Hughes, G. , Building a system failure rate estimator by identifying component failure rates, 1999, p32-41 in Software Reliability Engineering, 1999. Proceedings. 10th International Symposium on, http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=809308

[83] Dot Hill, SAN Tutorial, http://www.dothill.com/tutorial/, bezocht op 10 juli 2006

[84] Wikipedia, Storage Area network, http://en.wikipedia.org/wiki/Storage_area_network, bezocht op 10 juli 2006

[85] Tate, J. , Kanth, R., Telles, A., Introduction to Storage Area Networks, IBM Redbook, April 2005, http://www.redbooks.ibm.com/redbooks/pdfs/sg245470.pdf

[86] Novell Cluster Services, Troubleshooting Novell Cluster Services for netware 5 and netware 6 http://www.novell.com/documentation/ncs6p/pdfdoc/cluster_tswizard.pdf, bezocht op 10 juli 2006

[87] Linux-HA STONITH, Shoot The Other Node In The Head, http://www.linux-ha.org/STONITH, Bezocht op 17 juli 2006

[88] Oracle Data Guard Overview, http://www.oracle.com/technology/deploy/availability/htdocs/DataGuardOverview.html, Bezocht op 10 juli 2006

[89] Oracle Prijzen, SE Editie: http://www.oracle.com/technology/products/database/oracle10g/pdf/DS_General_Oracle_Database10gR2_SE_0605.pdf en EE editie: http://www.oracle.com/technology/products/database/oracle10g/pdf/DS_General_Oracle_Database10gR2_EE_0605.pdf, Bezocht op 10 juli 2006

[90] MySQL Replication Reference Manual, http://dev.mysql.com/doc/refman/5.1/en/replication.html, Bezocht op 10 juli 2006

[91] IBM DB2 Universal Database - High Availability Disaster Recovery (HADR) http://www-306.ibm.com/software/data/db2/udb/hadr.html, Bezocht op 10 juli 2006

[92] IBM DB2 Information Integrator: Introduction To Replication And Event Publishing ftp://ftp.software.ibm.com/ps/products/db2/info/vr82/pdf/en_US/db2gpe80.pdf, bezocht op 10 juli 2006

[93] Mienes, P. , OS/390… Waarom toch? Achtergronden van de betrouwbaarheid en beschikbaarheid van OS/390 Mainframes, 2001

http://www.isaca.org/Template.cfm?Section=Home&CONTENTID=16609&TEMPLATE=/ContentManagement/ContentDisplay.cfm, Bezocht op 10 juli 2006

[94] LaMonica, M, Market Research gives Oracle the Top Spot, 7 juni 2004, http://news.zdnet.com/2100-9592_22-5227856.html, Bezocht op 11 juli 2006

[95] Oracle 10g Flashback Technology overview http://www.oracle.com/technology/deploy/availability/htdocs/Flashback_Overview.htm, Bezocht 11 juli 2006

[96] Bakker, J. Computable, Bancaire uitwijkmanoeuvre, 13 december 2002, http://www.computable.nl/artikels/archief2/d50ra2gf.htm, bezocht op 11 juli 2006

[97] Succes stories in High Availability using VERITAS software - eBay , September 2001, http://www.avcomeast.com/eBay.pdf, bezocht op 11 juli 2006

[98] Stone-IT, Linux for Business, Klanten : Marktplaats http://www.stone-it.com/klanten.php?refsid=1, bezocht op 11 juli 2006

[99] MySQL AB: A guide to lower Database TCO, 31 januari 2005, http://www.mysql.com/why-mysql/white-papers/tco.php, bezocht op 12 juli 2006

[100] Microsoft SQL Server, Total Cost of Ownership http://www.microsoft.com/sql/prodinfo/compare/tco.mspx, bezocht op 12 juli 2006

[101] Vaas, L. , eWEEK, Sybase Takes on Database TCO, 6 october 2003, http://www.eweek.com/article2/0,1895,1499066,00.asp, bezocht op 12 juli 2006

[102] Standish Group, Battle of the Databases, Focus on TCO, 2003, http://www.sun.com/aboutsun/media/presskits/solarisx86/Sybase_Standish_Group_TCO.pdf, Bezocht op 12 juli 2006

[103] Marktplaats, Over marktplaats statistieken, http://statisch.marktplaats.nl/html/about_us.html, Bezocht op 12 juli 2006

[104] Transaction Processing Performance Council, www.tpc.org, Bezocht op 12 juli 2006

[105] Transaction Procession Performance Council, TPC-C, www.tpc.org/tpcc, Bezocht op 12 juli 2006

[106] Transaction Procession Performance Council, TPC-H, www.tpc.org/tpch, Bezocht op 12 juli 2006

[107] Open Source Development Labs, OSDL Database Testsuite, http://www.osdl.org/lab_activities/kernel_testing/osdl_database_test_suite/ , Bezocht op 12 juli 2006

[108] Linux Test Project test suite, http://ltp.sourceforge.net/, Bezocht op 12 juli 2006

[109] Barrera, D., Database Opensource Test suite User's guide, http://ltp.sourceforge.net/dotshowto.php, bezocht op 12 juli 2006

[110] Netem – LinuxNet, http://linux-net.osdl.org/index.php/Netem , bezocht op 12 juli 2006

[111] Ilse Media websites, http://www.Ilsemedia.nl/nl-web-Sites.php, Bezocht op 12 juli 2006

[112] Veritas, Solving the Business Problem of Downtime: Succesful Business Continuity Management and planning with Veritas Consulting Services, 2004, http://www4.symantec.com/Vrt/offer?a_id=10210, Bezocht op 17 juli 2006

[113] VMware ESX Server, http://www.vmware.com/products/vi/esx/, Bezocht op 18 juli 2006

[114] Boosting MySQL Cluster with Dolphin Sockets, http://www.dolphinics.com/products/software/mysqlcluster_booster.html, Bezocht op 20 juli 2006

[115] Veritas, http://www.symantec.com/enterprise/veritas/index.jsp, Bezocht op 20 juli 2006

[116] Sun Cluster, http://www.sun.com/software/cluster/, Bezocht op 20 juli 2006

[117] HeartbeatProgram, Linux HA, http://www.linux-ha.org/HeartbeatProgram, Bezocht op 24 juli 2006

10. WOORDENLIJST

ACR..... Automatic Client Reroute, techniek van IBM om clients van systeem naar systeem te routeren, als fouten of overnames optreden. Onderdeel van de DB2 database.

ANP..... Algemeen Nederlands Persbureau, verzorgt nieuwsberichten voor meerdere media

API ..... Application Programming Interface, is een interface tussen verschillende programma's, zodat programma's met elkaar kunnen communiceren. Met een API kan een programmeur verschillende functies achter de API in zijn eigen applicaties gebruiken.

Continuous

Availability..... Continue Beschikbaarheid. Een system dat continu beschikbaar is en downtime van componenten maskeert tov zijn gebruikers.

CA ..... Zie Continuous Availability

CIFS ..... Common Internet File System, file systeem ontwikkeld door Microsoft

CMS ..... Content Management System, wordt gebruikt om content mee vast te leggen en te organiseren.

CODA ..... Gedistribueerd file system

COTS..... Commercial/Common Off The Shelf, producten die gemakkelijk of goedkoop beschikbaar zijn in winkels

DB ..... DataBase, opslag van data, veelal in een relationeel formaat

Disaster Recovery .... Meestal een set van procedures waarin staat vastgelegd op welke manier het systeem weer in de lucht kan worden gebracht, nadat een disaster het datacenter heeft verwoest. DR-systemen hebben meestal speciale voorzieningen om op meerdere locaties te kunnen functioneren of binnen zeer korte tijd op een secundaire locatie te herstellen zijn.

DNB..... De Nederlandse Bank, verantwoordelijk voor alle banken in Nederland, en het Nederlandse beleid op gebied van bankzaken

DNS	Domain Name System, maakt het mogelijk om domeinnamen om te zetten in ip adressen, zodat machines bereikt kunnen worden via een naam ipv een nummer.
DOTS	Database Opensource Test Suite, test suite onderdeel van de linux test project, met benchmarks op JDBC basis.
DR	Zie Disaster Recovery
DRBD	Block device gebaseerde replicatie oplossing om tussen twee systemen dezelfde inhoud op disk te krijgen
DSS	Decision Suport System, rapportages van gegevens ter ondersteuning van het nemen van beslissingen (ad-hoc leesacties op grote hoeveelheden data)
Fail-over	Overschakeling van een service naar een secundair systeem bij het optreden van een fout in het primaire systeem. Zie ook Take-over.
Fault-Tolerant	Term wordt veelal gebruikt voor systemen die blijven functioneren als fouten optreden of componenten uitvallen. Vindt zijn oorsprong in FT-systems .
FMS	Fail-over Management System, applicatie waarmee grafisch een cluster beheerd kan worden en eventueel voor een automatische failover zorgt.
FT-system	Fault Tolerant system, een systeem waarbij alle hardware componenten dubbel zijn uitgevoerd, zodat de uitval van een enkel component getolereerd kan worden.
FT	Zie Fault-Tolerant
GFS	Global File System, een gedistribueerd cluster file system ontwikkeld door Red hat
GRID	Aan elkaar gekoppeld netwerk van computers die samen kunnen werken. Vergelijkbaar met clusters, maar dan anders georganiseerd.
High-Availability	Hoge Beschikbaarheid, een systeem dat meer dan 99,9 % van de tijd service kan verlenen aan de gebruiker.
HA	Zie High Availability.
HACMP	High Availability Cluster Multi-Processing systeem ontwikkeld door IBM

HADR	High Availability Disaster Recovery, onderdeel van de IBM DB2 database.
HAGEO	HACMP met een geografische spreiding.
JDBC	Java DataBase Connectivity, database toegang vanuit Java applicaties.
LAMP	Linux, Apache, MySQL, PHP/Perl/Python, een benaming voor een combinatie van software waarmee vooral webapplicaties worden ontwikkeld en gedraaid.
LAN	Local Area Network, benaming voor het netwerk binnen panden
LB	Zie Load Balancer
LDAP	Lightweight Directory Access Protocol, wordt gebruikt voor directory's met gegevens, vooral voor authenticatie van personen in grote omgevingen
Load Balancer	Een apparaat of software die verbindingen verdeeld over een aantal achterliggende machines, meestal met intelligente monitoring om belasting te kunnen meten.
LVS	Linux Virtual Server, een project waarmee een SSI gecreerd kan worden van meerdere systemen.
MDAC	Microsoft Data Access Components, vergelijkbaar met JDBC maar dan voor Microsoft programmeertalen.
MSCS	Microsft Server Clustering Services, Cluster software van Microsoft
MTBF	Mean Time Between Failure, de gemiddelde tijd van functioneren van een groep van componenten
MTTR	Mean Time To Repair, de gemiddelde tijd om een component te kunnen repareren.
NAT	Network Adress Translation, techniek om private ip netwerken via een publiek ip te laten communiceren
NLB	Network Load Balancing, zie Load Balancing
OLTP	On-line Transaction Processing, het online verwerken van transacties (zowel veel lees als schrijfacties ter verwerking van transacties)
OS	Operating System, besturingssyteem zoals Linux of Windows

OSDL	Open Source Development Labs, organisatie die zich inzet voor open source ontwikkeling
PHP	Scripttaal voor webapplicaties
RAC	Real Application Clusters, clustertechniek van Oracle
RAID	Redundant Array of Independent/Inexpensive Disks, meerdere harde schijven in combinatie, waarmee beschikbaarheid of snelheid kan worden verhoogd
RAS	Reliability, Availability and Serviceability, betrouwbaarheid, beschikbaarheid en onderhoudbaarheid.
RPO	Recovery Point Objective, een gesteld doel voor de hoeveelheid dataverlies die mag optreden bij een crash van het systeem. Een RPO van 1 uur betekent dat maximaal 1 uur dataverlies mag optreden om de doelstelling te halen. Alle data tot een uur voor de crash kan hersteld worden.
RTO	Recovery Time Objective, een gesteld doel voor de tijd waarin het systeem weer beschikbaar moet zijn. Een RTO van 4 uur betekent dat het systeem binnen 4 uur weer online moet zijn na uitval om de doelstelling te halen.
SAN	Storage Area Network, een opslagsysteem dat door meerdere systemen gebruikt kan worden via glasvezel, Scsi of IP netwerken.
SCI	Scalable Coherent Interface, high speed point-to-point packet protocol, extreem lage latency's mogelijk, toepasbaar in het MySQL Cluster product
SCSI	Technologie om verschillende devices op een PC aan te sluiten, in deze context vooral toegepast op storage.
Shared-storage	Gedeelde opslag van data. Meestal wordt gebruik gemaakt van Storage Area Networks. Elke node kan bij de data op de gedeelde opslag.
Shared-nothing	Elke host heeft zijn eigen dataopslag en deze wordt niet met andere nodes gedeeld.
SLA	Service Level Agreement, een set afspraken op papier, waarin de verwachte service overeen wordt gekomen.
SPOF	Single Point Of Failure, een component dat bij uitval een volledig systeem onbruikbaar maakt.

Split-brain	Twee aparte delen van het cluster die blijven functioneren zonder dat ze met elkaar kunnen communiceren, mogelijk met inconsistent gedrag en data tot gevolg
SQL	Structured Query Language, een taal om databases mee te bevragen.
Single System Image	Een aantal verschillende systemen dat zich naar buiten toe gedraagt als een systeem.
Site	Datacenter locatie, wordt meestal gebruikt in discussies over disaster recovery
SSI	Zie Single System Image.
STONITH	Shoot The Other Node In The Head, een methode om te zorgen dat bij een mogelijk split-brain scenario slechts een node overblijft die de service uitvoert.
Take-over	Overschakeling van een service naar een secundair systeem, geïnitieerd door een persoon, meestal bij plegen van onderhoud aan primair systeem. Zie ook Fail-over.
TCR	Transparent Client Reroute, Oracle's client reroute techniek
TPC	Transaction Processing Performance Counsil, Organisatie die zich richt op het benchmarken van database systemen.
WAN	Wide Area Network, een netwerk tussen meerdere locaties, meestal meer dan een km uit elkaar

APPENDIX A. PRODUCTEN

Deze appendix beschrijft verschillende producten die op het moment van opstellen beschikbaar zijn op de markt of zeer binnenkort op de markt zullen verschijnen. Producten die na eind april 2006 op de markt zijn verschenen kunnen nieuwe en/of gewijzigde functies bevatten ten opzichte van de hier beschreven functionaliteit en features.

Paragrafen A.1 tot en met A.3 beschrijven de grote drie proprietaire database producenten, respectievelijk Oracle, Microsoft en IBM. Paragrafen A.4 en A.5 beschrijven de open-source aanbieders MySQL en PostgreSQL. Paragraaf A.6 beschrijft de verschillende database middleware oplossingen die op de markt verkrijgbaar zijn. De software waarmee cluster worden beheerd en georganiseerd wordt in A.7 beschreven.

A.1. Oracle

Oracle biedt een aantal verschillende producten aan voor niet alleen beschikbaarheid, maar ook voor het doen van onderhoud, het maken van backups en het herstellen van fouten. Hier worden alleen de producten beschreven die direct invloed uitoefenen op beschikbaarheid of replicatie. Tools zoals Oracle Flashback waarmee snapshots hersteld kunnen worden, komen niet aan de orde.

Oracle heeft drie verschillende producten op HA gebied, namelijk Oracle Fail Safe, Real Application Clusters en Data Guard. Oracle Fail Safe is een uitbreiding op Microsoft Cluster Server en biedt fail-over clustering. Real Application Clusters is een echte cluster oplossing, waarmee tot 100 nodes worden ondersteunt. Oracle Data Guard kan worden gebruikt voor het repliceren van data en fail-over.

A.1.1. Oracle Fail Safe

Oracle Fail Safe [33] is High Availability software die de fail-over van Oracle databases verzorgt in een Microsoft Cluster Server (MSCS) [55]. MSCS kan gebruikt worden om taken van het primaire systeem in een cluster over te zetten naar een secundair systeem. MSCS heeft echter geen kennis van Oracle databases en applicaties. Fail Safe integreert met MSCS om de Oracle databases en applicaties van fail-over functionaliteit te voorzien die is afgestemd op de Oracle producten.

Het gebruik van shared-storage in een MSCS cluster is noodzakelijk, zodat het secundaire systeem bij de data kan als een fail-over plaats moet vinden. Bij een fail-over wordt de data afgekoppeld van het primaire systeem en toegewezen aan het secundaire systeem, zodat deze de data kan bewerken.

Door middel van Fail Safe kunnen zogenaamde rolling cluster upgrades worden uitgevoerd. Dit betekent dat systemen in verschillende fasen van een nieuwe versie kunnen worden voorzien. Fail Safe houdt de database in de gaten en kan

een fail-over initiëren als de database faalt. Zonder het gebruik van Fail Safe leiden database problemen niet automatisch tot een fail-over, tenzij een ander component,bijvoorbeeld een harde schijf, wat door het MSCS wordt bewaakt ook faalt. In dat geval worden alle services naar een secundair systeem overgeheveld door het MSCS. De Fail Safe software biedt specifieke bewaking van de database en hevelt deze over als er voor de rest geen problemen zijn die voor het MSCS tot actie zouden leiden.

Fail Safe is alleen mogelijk op Windows systemen die de MSCS technologie ondersteunen.

A.1.2. Oracle Data Guard

Oracle Data Guard [35] is een standaard onderdeel van de Oracle Database Enterprise Edition en is vergelijkbaar met Microsoft SQL Server Database Mirroring. Data Guard biedt een replicatie van database transacties door middel van transactie logs. Data Guard houdt een transactioneel consistente kopie van de productie database bij op tot negen standby databases.

Data Guard kan zowel met een redo apply als een sql apply werken. In Redo apply wordt de redo data van de productie database op de standby database gesynchroniseerd. De standby is een physical standby van de productie database. De stand-by kan gebruikt worden voor read-only requests, zoals rapportages en het maken van back-ups. In SQL apply worden de redo logs omgezet naar sql statements, waarna deze op de stand-by worden uitgevoerd.

Bij het gebruik van SQL apply wordt de stand-by database een logische stand-by genoemd. Dit betekent dat de database niet afhankelijk is van de primaire database en ook op zichzelf kan functioneren. Er worden slechts SQL statements vanaf een andere machine uitgevoerd, maar voor de database is er geen onderscheid tussen een replicatie statement of een sql statement van een gewone client. Hierdoor is het mogelijk dat deze database ook voor schrijfacties van clients gebruikt kan worden. Er wordt echter niet teruggesynchroniseerd naar de primaire database, zodat de ingevoerde gegevens op deze machine verloren gaan als deze uitvalt. Hierdoor is SQL apply vooral geschikt als een decision support system, waarop rapportages uitgevoerd kunnen worden en eventuele resultaten opgeslagen kunnen worden.

Data Guard heeft drie verschillende modi, namelijk Maximum Protection, Maximum Availability en Maximum Performance. De eerste twee bieden synchrone replicatie met en zonder blocking, de laatste asynchrone replicatie.

Maximum Protection biedt synchrone replicatie van de primaire naar de standby databases. Transacties worden pas gecommit als redo data beschikbaar is op minimaal één van de stand-by servers. Als er geen stand-by servers beschikbaar zijn, dan stopt het verwerken van de transactie en kan er geen data meer weggeschreven worden. De transacties worden door de Oracle database geblokkeerd, omdat niet de garantie gegeven kan worden dat de data op minimaal twee plekken weggeschreven kan worden. Data Guard Maximum Protection garandeert dus dat transacties die verwerkt zijn op minimaal twee

plekken zijn opgeslagen, zodat de uitval van een van de systemen geen dataverlies met zich meebrengt. Oracle raadt bij het gebruik van Maximum Protection aan om minimaal twee stand-by servers te gebruiken, zodat de uitval van een stand-by geen stop van transacties betekent. Door het gebruik van synchrone replicatie is er een negatieve impact mogelijk op de reponse tijden van de database. Het gebruik van hoge bandbreedte met lage latency's wordt daarom aangeraden. [35]

Maximum Availability biedt hetzelfde als Maximum Protection met als verschil dat het ontbreken van een stand-by server geen onderbreking oplevert van de verwerking van transacties. Op deze manier heeft het optreden van een stand-by server crash of netwerk partitie geen gevolgen voor de beschikbaarheid van de data. Mogelijk dataverlies kan optreden als de productie database uitvalt, voordat de stand-by of het netwerk gerepareerd is. Als het probleem herstelt is, zal de stand-by gesynchroniseerd worden tot de laatste transactie en treedt geen dataverlies op indien de productie dan uitvalt.

Maximum Performance tenslotte maakt gebruik van een asynchrone replicatietechniek, waarbij de replicatie van de data geen invloed uitoefent op het aantal transacties wat het systeem kan verwerken. Maximum Performance is de standaard techniek van Data Guard en is vooral aan te raden bij applicaties die hoge performance nodig hebben of replicatie over WAN links met hoge latency. Deze vorm van replicatie loopt wel het risico dat dataverlies optreedt bij het optreden van een crash van de primaire server, doordat de stand-by niet volledig up-to-date is.

Data Guard ondersteunt automatische failover, zodat een standby database de taken van de productie database over kan nemen. Voor het gebruik van Data Guard is het noodzakelijk hetzelfde platform te gebruik voor zowel de productie als de standby database. Dat betekent overal het zelfde operating system en dezelfde database versie, tenzij logical standby's worden gebruikt in rolling update mode [35].

A.1.3. Oracle Real Application Clusters (RAC)

Oracle Real Application Clusters [56], afgekort tot RAC, is de database cluster oplossing van Oracle. Met Oracle RAC in versie 10g R2 kunnen maximaal 100 machines, afhankelijk van het gebruikte operating system, worden samengebracht om als een grote database te opereren [57].

In het RAC systeem is een scheiding gemaakt tussen de processen die de data benaderen, de zogenaamde Oracle Instances en de data opslag, de Oracle Database. Bij een standalone database bevinden beide zich op hetzelfde systeem, in RAC bevindt de Oracle Database zich op de storage laag en de Oracle Instances op de verschillende cluster machines. RAC bestaat dus uit twee verschillende lagen, die met elkaar moeten communiceren. Om alle instances toegang te geven tot de Oracle Databases is het gebruik van shared-storage noodzakelijk [57]. Om te voorkomen dat de shared-storage een single point of failure wordt, is het noodzakelijk deze redundant uit te voeren.

Voor het gebruik van meerdere locaties biedt Oracle het RAC on Extended Distance Clusters aan. Hierbij wordt een oplossing geboden voor het volledig uitvallen van een datacenter. Door middel van het extended distance cluster kan op twee datacenters het RAC cluster gedraaid worden, waarbij de afstand beperkt moet worden tot maximaal 100km. Grotere afstanden zijn door de dan steeds hoger wordende latency's niet aan te raden [57]. De extreem hoge kosten van een aanbevolen directe kabelverbinding tussen beide datacenters zal deze oplossing slechts voor een beperkt aantal organisaties haalbaar maken. Bij uitvalsscenario's waarbij de oorzaak slechts een beperkte regio betreft zal dit product een oplossing bieden, bij scenario's waarbij grote regio's getroffen worden zoals aardbevingen en overstromingen voldoet ook de Extended distance cluster niet. In die gevallen wordt het gebruik van RAC in combinatie met Data Guard aangeraden, zodat grotere afstanden overbrugd kunnen worden. Daarbij is de configuratie en installatie van RAC met Data Guard een stuk eenvoudiger ten opzichte van de Extended Distance Clusters variant [57].

A.2. Microsoft

Microsoft biedt bij haar database SQL Server 2005 een aantal verschillende functies die de beschikbaarheid van de data kunnen verhogen. Database Mirroring, vergelijkbaar met Oracle Data Guard wordt beschreven in A.2.1. Log Shipping, waarbij de transactie logs worden verstuurd naar het standby systeem wordt beschreven in A.2.2. De Replication techniek waarmee een publiceer/inschrijf model, vergelijkbaar met IBM's Q-Replication, wordt gehanteerd staat beschreven in A.2.3. De cluster oplossing van Microsoft genaamd N-Way Clustering volgt in A.2.4.

Niet al deze opties zijn op elke database versie van Microsoft beschikbaar, een overzicht van de beschikbaarheid staat in [31].

A.2.1. Microsoft SQL Server Database Mirroring

Microsoft SQL Server 2005 bevat een technologie genaamd Database Mirroring [7],[6]. Database mirroring werkt op het database niveau en voorziet in het spiegelen van de database door middel van replicatie. Deze technologie bevat twee modi, asynchrone en synchrone mirroring.

Bij synchrone mirroring wordt de transactie log overgestuurd naar de mirror, waarna op een antwoord wordt gewacht. De primaire node commit de transactie pas nadat een antwoord van de mirror is ontvangen en geeft een response terug aan de client. In asynchrone modus wordt de transactie gelijk gecommit en wordt daarna de transactie naar de mirror gestuurd, waardoor de mirror enige transacties achter kan lopen. Er wordt direct een antwoord naar de client teruggestuurd.

Bij het gebruik van de synchrone modus kan door middel van een zogenaamde witness automatische failover bereikt worden. Deze witness is een onafhankelijk derde systeem die de primaire en backup node in de gaten houdt. Bij een probleem op de primaire node zorgt de witness ervoor dat de backup de databasetaken overneemt van de primaire node. In het geval van asynchrone modus of het ontbreken van een witness is alleen een manuele fail-over mogelijk.

Het secundaire systeem kan niet gebruikt worden voor leesacties op de gemirrorde database, deze database is continu in recovery mode. Door het gebruik van databasesnapshots is het mogelijk om een snapshot te gebruiken voor bijvoorbeeld rapportages. Een databasesnapshot geeft een momentopname weer van de database, die niet wordt gewijzigd terwijl er nieuwe data binnenkomt. Een snapshot is daarom niet te gebruiken voor applicaties die real-time data moeten verwerken, maar alleen voor applicaties die rapportages moeten maken.

Om ervoor te zorgen dat de clients het goede systeem bevragen, kan gebruik worden gemaakt van Transparent Client Redirection dat geïntegreerd is in Microsoft Data Access Components (MDAC) [7] . Door de TCR techniek kunnen

clients naadloos switchen van primaire naar backup server. Het adres van de backupserver wordt automatisch opgehaald als met de primaire server wordt verbonden. Als een retry op de primaire node niet werkt, wordt automatische de backup server gebruikt. Door het gebruik van MDAC zijn aanpassingen niet nodig in de client. Deze techniek is vergelijkbaar met de Automatic Client Reroute van IBM.

A.2.2. Microsoft SQL Server Log Shipping

Log Shipping [58] is een vergelijkbare techniek met Database Mirroring, maar dan meer asynchroon. Log Shipping werkt door op gezette tijden de gewijzigde data door middel van de transactie logs door te sturen naar de secundaire servers. Deze techniek ondersteunt meerdere secundaire machines en deze machines kunnen gebruikt worden om van te lezen. Er is in tegenstelling tot Database Mirroring geen automatische failover mogelijk.

Door middel van een time-delay kunnen fouten van gebruikers worden hersteld, voordat deze worden gerepliceerd naar de stand-by databases. Op deze manier is het binnen de tijd vertraging nog mogelijk een fout van een gebruiker of beheerder te herstellen vanaf de standby locatie, zonder dat een oude backup teruggezet moet worden. Het nadeel van deze tijd vertraging is het grotere dataverlies in geval van crash van de primaire server. Microsoft positioneert log shipping als oplossing voor het uitvallen van locaties en lokale server uitval [6] Indien geen dataverlies op mag treden is het niet geschikt en zal naar Database Mirroring gekeken moeten worden.

A.2.3. Microsoft SQL Server Replication

Microsoft SQL Server Replication is vergelijkbaar met IBM's Q-replication in het gebruik van een publish-subscribe model. De primaire database publiceert de data die gerepliceerd moet worden, waarna de data door een distributeur wordt gedistribueerd naar de verschillende subscribers.

Grote voordeel van Replication is het beschikbaar zijn van de subscriber database voor rapportages. Replication is niet gebouwd voor High Availability en daarom is er geen automatische failover mogelijk. Wel kan door middel van Windows Cluster Services een subscriber gepromoveerd worden naar primary server. Naar een eenmaal gefaalde primary kan niet makkelijk terug worden geswitcht, deze dient dan een volledige database restore te ondergaan. Replication is vooral bruikbaar als backup oplossing van data. [6]

A.2.4. Microsoft SQL Server N-way Clustering

Microsoft SQL Server N-Way Clustering maakt gebruik van de Windows Cluster Service support die door het Windows systeem geboden wordt. Het is vergelijkbaar met de Oracle Fail Safe oplossing. Afhankelijk van de gebruikte Windows versie kunnen 2, 4 of 8 nodes in een cluster gebruikt worden. Voor dit cluster is een shared-storage systeem nodig, met ofwel Fibre Channel of SCSI support.

Windows Clustering biedt support voor automatische failover, virtuele ip adressen en een failover tijd van ongeveer 30 seconden. Door middel van een heartbeat protocol wordt bijgehouden of de nodes nog actief zijn. Windows Clustering is ook beschikbaar voor Oracle en IBM databases en dus niet specifiek voor Microsoft SQL server [6].

A.3. IBM

IBM biedt verschillende oplossingen aan zowel voor data replicatie als voor cluster oplossingen. Onder de naam High Availability Disaster Recovery (HADR) van de IBM DB2 Universal Database wordt een oplossing aangeboden waarmee een database high available gemaakt kan worden over meerdere locaties. SQL en Q-replication zijn asynchrone replicatietechnieken. Verder biedt IBM nog het Geographically Distributed Parallel Sysplex systeem aan, een cluster oplossing over meerdere locaties.

A.3.1. IBM DB2 UDB High Availability Disaster Recovery

De IBM High Availability Disaster Recovery (HADR) [91] oplossing biedt volgens IBM een High Availability oplossing, zelfs bij het gedeeltelijk of compleet uitvallen van een locatie. Door middel van replicatie wordt dataverlies voorkomen door data te repliceren van een primaire database naar een stand-by database [10].

HADR ondersteunt drie verschillende replicatie modes, namelijk synchroon, bijna synchroon en asynchroon. Bij alle modes heeft een netwerk of standby failure geen invloed op de primary, het systeem blijft dus wel beschikbaar voor de clients. Door de client reroute functionaliteit kunnen clients automatisch naar de standby worden gerouteerd als de standby de functionaliteit van de master heeft overgenomen. HADR biedt geen automatische failover van de primary naar de standby, dit kan door bijvoorbeeld HACMP worden geregeld. Voor automatische failover is een cluster manager benodigd, die bij een failure het takeover commando uitvoeren. HADR gebruikt de client reroute functionaliteit voor de automatische failover.

In synchrone modus worden de log files zowel op de primary als op de standby weggeschreven op disk voordat de applicatie een succesvol antwoord krijgt op zijn commit statement. Elke transactie is daardoor gegarandeerd opgeslagen als deze gecommit is. In near-synchrone modus wordt een antwoord aan de applicatie gegeven als de gegevens in de standby in het geheugen staan. Hier treedt dataverlies op als zowel de primary als de standby tegelijkertijd falen en de standby als primary gaat dienen. In asynchrone mode wordt een antwoord aan de applicatie gestuurd op moment dat de pakketten bij de tcp/ip socket zijn aangekomen. Hierbij is het mogelijk dat data verloren gaat, als de primary faalt en de pakketten niet aankomen bij de standby voordat deze een take-over doet van het systeem.

Een nadeel van de HADR oplossing is het niet kunnen lezen van de standby server, terwijl deze aan het repliceren is met de primary. Alleen de primary kan gebruikt worden voor lees en schrijfacties, de standby kan alleen replicatie data ontvangen. Deze oplossing is dus niet schaalbaar door meerdere systemen toe te voegen, maar alleen door betere hardware toe te passen. Een oplossing die wel leesacties op standby nodes laat uitvoeren is Q replication, een asynchrone replicatietechniek.

A.3.2. IBM DB2 UDB SQL Replication

SQL Replication [92] is een asynchrone replicatietechniek die uit de recovery logs de gewijzigde data laadt naar een staging table. De data uit deze table's kunnen dan gebruikt worden om slaves van nieuwe data te voorzien. De gewijzigde data wordt dus door middel van SQL replication overgezet van de source naar de target. SQL replication wordt vooral gebruikt bij data warehouses en datamarts doordat het puur de data kan kopieren.

A.3.3. IBM DB2 UDB Q-Replication

Q Replication [92] maakt gebruik van een publish-subscribe asynchroon replicatie mechanisme. Een source publiceert door middel van Qcapture uit de recovery log een bepaalde tabel naar een Websphere MQ queue. Uit deze queue kunnen een of meerdere subscribers wijzigingen op deze tabel door middel van Qapply repliceren naar hun eigen tables. Q replication ondersteunt éénweg, tweeweg of peer-to-peer replicatie.

Eénweg replicatie komt overeen met een master-slave setting, tweeweg replicatie geeft een master-master setting, en peer-to-peer geeft een Multi-master setting. Doordat asynchrone replicatie wordt gebruikt, dient bij de tweeweg en peer-to-peer replicatie rekening te worden gehouden met optredende conflicten door wijzigingen die op beide systemen zijn uitgevoerd voordat gesynchroniseerd is. Q Replication biedt als conflict resolutie oplossing een preferentie van de tabel, zodat aangegeven kan worden welke tabel als leidend gezien moet worden. Hiermee kan wel data verlies optreden, doordat data van de niet leidende tabel vervalt.

Doordat een asynchrone techniek wordt gebruikt, kan dataverlies optreden als systemen uitvallen. Q-replication biedt ook geen ondersteuning voor automatische fail-over, maar doet alleen de replicatie van data. Q-replication is vooral geschikt voor applicaties waar enig dataverlies geen probleem is en het maken van continue backups die enige tijd achterlopen, maar niet zoveel als een normale tape backup.

A.3.4. IBM DB2 Parallel Sysplex

Bij het gebruik van het s/390 mainframe van IBM bestaat er de mogelijkheid om gebruik te maken van de Parallel Sysplex functionaliteit [93]. Hiermee kunnen door middel van de Coupling Facility (CF) verschillende systemen aan mekaar worden gekoppeld. Omdat voor de parallel sysplex niet alle systemen in dezelfde ruimte te staan, is het mogelijk om een zogenaamd geographically dispersed parallel sysplex op te zetten. De DB2 database draait dan op verschillende locaties en kan uitval van een van de componenten opvangen. De afstand tussen de locaties is beperkt door de replicatie van de data op de gedeelde opslag schijven.

A.4. MySQL

MySQL biedt twee verschillende technologieën aan voor database replicatie. MySQL Cluster biedt directe ondersteuning voor High Availability, terwijl MySQL Replicatie slechts de replicatie van data voor zijn rekening neemt. MySQL cluster biedt een synchrone Multi-Master techniek, die wordt beschreven in A.4.1, de asynchrone replicatie wordt beschreven in A.4.2. Beide technieken zijn een standaard onderdeel van de MySQL Server distributie.

A.4.1. MySQL Cluster

MySQL Cluster is een techniek die sinds versie 4.1 van de MySQL server beschikbaar is. MySQL Cluster maakt gebruik van een synchrone multi-master techniek om data te repliceren over meerdere systemen. Doordat een multi-master techniek wordt gebruikt, kunnen meerdere systemen tegelijkertijd gebruikt worden om transacties op uit te voeren. De master zoals die nodig is in een master-slave architectuur is hierbij niet meer noodzakelijk. Het uitvallen van een enkel systeem veroorzaakt daarom geen uitval van de database, zoals dat wel het geval is als de master in een master-slave configuratie uitvalt. De single point of failure problematiek in een master-slave systeem is hiermee opgeheven in MySQL Cluster. MySQL claimt dat zijn MySQL Cluster techniek een beschikbaarheid van 99.999% behaalt [20][11].

De eerste MySQL Cluster is beschikbaar sinds versie 4.1 van de MySQL Server en werd ontwikkeld als een pure in-memory database. Ook versie 5.0 biedt alleen ondersteuning aan een in-memory database. Dit betekent dat alle data die zich in de database bevindt in het geheugen van de clustermachines moet voorkomen. De hardeschijf wordt slechts gebruikt om kopieën van de data in het geheugen op te slaan voor het geval alle nodes uitvallen door stroomuitval. In dat geval kan het cluster hersteld worden met de data die is opgeslagen, echter kan dan wel dataverlies zijn opgetreden.

Doordat alleen geheugen gebruikt wordt, is het mogelijk om de vertraging van het gebruik van harde schijven te omzeilen en wordt een behoorlijke performance winst geboekt. Het grote nadeel van dit systeem is het feit dat bij grote datavolumes, die richting de honderden gigabytes lopen ook zoveel geheugen nodig is. Omdat geheugen vele malen duurder is dan hardeschijf ruimte betekent dit een hogere uitgave om mee te schalen met de data. Voor een applicatie waarbij weinig schrijfacties nodig zijn, kan dat een eenmalige investering zijn die de moeite waard is. Applicaties die veel schrijfacties hebben en daardoor snel groeien in benodigde datacapaciteit vergen grote investeringen in geheugenuitbreiding, waardoor dit voor deze applicaties een dure oplossing wordt. Versie 5.1 van MySQL Cluster zal ook ondersteuning bieden voor disk-based clustering, waarmee dit grote nadeel van deze cluster techniek deels wordt opgelost. Bij deze versie is een keuze mogelijk tussen het gebruik van hardeschijf of geheugen, zodat de organisatie een keuze kan maken tussen performance en uitbreiding, zolang het data zonder indexes betreft. Data met indexes en index data zelf dienen in deze versie wel nog in geheugen te worden opgeslagen.

Voor MySQL Cluster is een minimum van drie machines nodig. Cluster maakt gebruik van drie verschillende nodes [20]: Storage nodes, Management nodes en SQL of API nodes.

Storage nodes worden gebruikt om de data op te slaan. Deze data wordt continu in geheugen gehouden en op geselecteerde momenten weggeschreven naar disk. Tussen de storage nodes wordt de data continu synchroon gerepliceerd. De storage nodes handelen alle database transacties af.

Een of meer Management Server nodes verzorgen de configuratie van het systeem. Deze nodes zijn alleen benodigd bij start-up en als een node uitvalt of wordt toegevoegd. Aangezien ongeplande uitval van een node kan voorkomen, is het uitzetten van deze management nodes niet verstandig.

MySQL server nodes ookwel API nodes genoemd, fungeren als tussenlaag tussen de applicaties en de storage nodes. Hierdoor is het voor de applicatie niet zichtbaar dat gebruik wordt gemaakt van een cluster oplossing en hoeft de applicatie niet herschreven te worden, omdat dezelfde interface wordt gebruikt als bij een normale MySQL server. De MySQL Server nodes kunnen ook andere data opslaan, omdat het cluster is geïmplementeerd als een aparte storage engine. Hierdoor kan bijvoorbeeld ook statische data in de server nodes worden opgeslagen, waardoor het cluster minder belast wordt.

MySQL Cluster maakt in tegenstelling tot Oracle RAC geen gebruik van een Shared-storage architectuur, maar van een Shared-nothing architectuur. Elke node bevat zijn eigen storage met eigen data. Afhankelijk van de instelling bevat elke node een gedeelte tot een hele kopie van de volledige database. MySQL Cluster biedt de keuze tussen een tot vier replica's van de database. Eén replica biedt geen redundantie, doordat de database slechts één keer wordt opgeslagen. Als een node crasht, is het deel wat op deze node stond niet meer beschikbaar. Standaard worden twee replica's gebruikt.

MySQL Cluster is ontworpen voor minimaal 100Mbps LAN netwerken, waarbij voor optimale performance Gbit Ethernet of speciale SCI hardware wordt aangeraden [114]. Omdat performance een belangrijk aandachtspunt was bij de ontwikkeling, is het product zodanig opgezet dat geen rekening wordt gehouden met hoge latency's of het regelmatig uitvallen van verbindingen. Ook is voor de synchronisatie van de data tussen de nodes een grote bandbreedte capacitieit nodig. Het gebruiken over een Wide Area netwerk wordt daarom afgeraden, waarbij zelfs wordt betwijfeld of het betrouwbaar zou werken, omdat het daar niet voor ontworpen is. Een tweede probleem is het niet geencrypteerd verzenden van data tussen de storage nodes. Omdat uitgegaan wordt van een lokaal secure netwerk is dit geen issue, maar over internet dient hier wel rekening mee te worden gehouden [12].

Deze oplossing is vooral goed bruikbaar bij websites met database backend waar veel leesacties plaatsvinden of systemen met schrijfacties waarbij geen grote historie online in hetzelfde systeem bewaard hoeft te blijven, zodat de data in het geheugen kan blijven en geen diskopslag nodig is.

A.4.2. MySQL Replicatie

MySQL Replicatie [90] is beschikbaar sinds versie 3.23.15 van de MySQL server. Deze replicatietechniek biedt slechts ondersteuning voor replicatie in één richting van master naar slave. Door middel van asynchrone replicatie op basis van statements, die worden opgeslagen in een binaire logfile, wordt de data op de slaves geupdate. Doordat de replicatie asynchroon verloopt, kan er een vertraging optreden. De data op de slave kan hierdoor afwijken van de data op de master, omdat een wijziging op de master nog niet is doorgevoerd op de slave.

Om een MySQL database van High Availability te voorzien, is alleen replicatie niet voldoende. De replicatietechniek biedt geen ondersteuning voor automatische fail-over indien op de master een probleem optreedt. Voor deze fail-over dient externe software (buiten de MySQL database om) te worden gebruikt. Door gebruik te maken van bijvoorbeeld het Linux-HA framework of eigen geschreven scripts kan een fail-over automatisch worden uitgevoerd. Doordat een bepaalde tijd verloren gaat, voordat een probleem gedetecteerd kan worden, biedt deze oplossing geen volledige beschikbaarheid. Afhankelijk van de detectietijd, kan dit een tiental seconden duren. Doordat er geen synchrone replicatie wordt gebruikt, is het mogelijk dat een deel van de data verloren gaat tijdens een fail-over.

De asynchrone techniek heeft als voordeel dat een lage bandbreedte met relatief hoge latency's geen invloed heeft op de performance van het systeem. De techniek is daarmee prima geschikt om grote afstanden te overbruggen, waarbij een klein dataverlies acceptabel is als er problemen optreden op een locatie. Door middel van chaining kunnen er meerdere slaves op een master aangesloten worden. Twee systemen naar elkaar laten repliceren is met wat scripts ook mogelijk, maar daarbij moet wel met een aantal problemen rekening worden gehouden. Multimaster wordt niet aangeraden door MySQL.

A.5. PostgreSQL

PostgreSQL [59] is een open source database waarbij door middel van plugins bepaalde functionaliteit kan worden toegevoegd. Standaard bevat de database geen replicatie oplossing. PostgreSQL is van mening dat er verschillende replicatie mechanismen zijn, waarbij de gebruiker zelf moet kiezen welke oplossing hij nodig heeft. Daardoor zijn er verschillende plugins voor het PostgreSQL systeem die replicatie aanbieden. Voor het gebruik van Master/Slave replicatie kan Slony-I gebruikt worden, voor Multi-Master replicatie is Pgcluster ontwikkeld. Postgres-R is een synchroon master/slave protocol gebaseerd op een groep communicatie systeem. Slony-II is op dit moment in ontwikkeling als opvolger van pgcluster en postgres-R en zal gebaseerd zijn op Multi-master replicatie door middel van een groep communicatie systeem.

A.5.1. PostgreSQL Slony-I

Slony-I [60] voor PostgreSQL is een trigger gebaseerde master naar een of meerdere slaves replicatie mechanisme. Slony-I maakt gebruik van asynchrone replicatie. Dit is vooral bruikbaar voor het repliceren naar externe datacenters of load balancing voor leesintensieve applicaties. Slony verzorgt alleen replicatie en heeft geen support voor automatische fail-over. Bij het uitvallen van de master machine zal niet automatisch door Slony een slave als nieuwe master worden gepromoot. Voor het hoog beschikbaar maken van een systeem met Slony-I dient externe software gebruikt te worden, zoals het Linux-HA project of Veritas Cluster.

A.5.2. PostgreSQL Postgres-R

Postgres-R [27] is een prototype ontwikkeld door Bettina Kemme van de universiteit McGill in Canada om aan te tonen of een master/slave systeem gebaseerd op een groep communicatieprotocol kan werken. Postgres-R is gebaseerd op versie 6.2 van PostgreSQL en is niet geporteerd naar nieuwere versies. Doordat in versie 7 van PostgreSQL gebruik wordt gemaakt van een Multi version concurrency control systeem kon de Postgres-R code niet gemakkelijk worden overgezet.

Het postgres-R maakt gebruik van een groep communicatie systeem om data synchroon te kunnen repliceren. Dit systeem probeert de slechtere schaalbaarheid en deadlocks van het 2 Phase Commit protocol te omzeilen door gebruik te maken van de Reliable en Total Order eigenschappen van het groep communicatie systeem. Doordat alle berichten bij elke node in dezelfde volgorde aankomen, levert dit een serializable volgorde op. Het communicatie systeem zorgt voor een betrouwbare aflevering van elk bericht, tenzij de ontvangende node crasht. Dan wordt deze node uitgesloten van de berichten, totdat deze weer hersteld is.

Toekomstig onderzoek naar Postgres-R zal zich richten op het uitbreiden naar een Multi-master systeem en integratie in de nieuwere versies van PostgreSQL [26].

A.5.3. PostgreSQL PGCluster

PGCluster [24] voor PostgreSQL is een synchrone multi-master replicatietechniek, die behalve replicatie ook High Availability aanbiedt. Het systeem bestaat uit drie verschillende servers, een Load Balancer, een of meer cluster database nodes en een replicatie server. Een database request wordt door de Load Balancer naar een van de cluster database nodes verstuurd. De cluster database node verstuurt een request naar de replicatie server om de taak uit te voeren, als het om een schrijftaak gaat Nadat de replicatie server ervoor gezorgd heeft dat de taak gerepliceerd is naar alle nodes, antwoordt de bevraagde cluster database node via de load balancer de uitkomst van de request aan de client. Indien het om een leesactie gaat, dan wordt direct door de cluster node geantwoord.

Een falende cluster database node wordt door de load balancer niet meer van requests voorzien en door de replicatie server niet meer gebruikt voor replicatie. Nadat de server hersteld is wordt deze automatisch weer van correcte data voorzien en daarna weer opgenomen in het cluster systeem.

Zowel de load balancer als de replicatie server kunnen van een standby worden voorzien, zodat als deze falen, het systeem beschikbaar blijft. Als er geen replicatie server beschikbaar is, draaien de cluster nodes in stand-alone mode. Afhankelijk van de instelling kan er dan alleen nog gelezen worden of ook geschreven. De keuze tussen beide moet gemaakt worden op het feit of data inconsistenties voor mogen komen als er nog geschreven moet worden bij uitval. Indien schrijfacties worden toegestaan, lopen de verschillende cluster nodes uit elkaar en ontstaat inconsistente data. Is onconsistente data onacceptabel, dan moet de schrijfcapaciteit in dat geval worden uitgechakeld.

PGCluster is bruikbaar voor versies 7.3, 7.4, 8.0 en 8.1 van de PostgreSQL database distributie en is evenals de andere plugins gratis beschikbaar.

A.5.4. PostgreSQL Slony-II

Slony-II voor PostgreSQL is op dit moment in ontwikkeling en zal gebruik maken van een groep communicatie protocol voor de verzending van de transacties naar alle nodes, zoals dat ook door Postgres-R is gebruikt. Slony-II dient een Multi-master replicatie protocol voor PostgreSQL te worden dat beter schaalt dan het 2 Phase Commit protocol door het beperken van het aantal berichten dat nodig is om transacties te repliceren over de nodes. Behalve een presentatie over Slony-II [25] is er verder nog weinig bekend over de voortgang van dit project en eventuele bruikbaarheid in de toekomst.

A.6. Middleware

Middleware functioneert als een tussenlaag tussen de database opslag en de applicatie die gebruik maakt van de data. Er zijn zowel producten gebaseerd op JDBC als producten die een database emuleren richting de applicatie, zoals beschreven in 3.4. Zowel Sequoia A.6.1 als HA-JDBC A.6.2 zijn gebaseerd op de JDBC methode. Continuent biedt een commerciële variant aan van Sequoia en heeft met m/cluster 2005 ook een database emulatie product in huis.

A.6.1. Sequoia

Sequoia [44] is een open-source project en is het vervolg van het C-JDBC (Clustered JDBC) project. Het C-JDBC project [45] wordt gehost door het Object-Web consortium. Sequoia wordt gesponsord door Continuent, het vroegere Emic-Networks. Meer over de Continuent producten staat in A.6.3.

Sequoia is een middleware component op JDBC niveau. Het is geschreven in Java en positioneert zich tussen de applicatie en de database. Door middel van een Sequoia JDBC driver kan de applicatie verbinding leggen met een controller. Deze controller ondersteunt door middel van JDBC alle database producten met een JDBC driver. De controller maakt verbinding met de verschillende databases en verzorgt zowel de replicatie als de load balancing en uitschakeling van database bij failures. Er kunnen meerdere databases op een controller worden aangesloten. Een database kan echter slechts één controller hebben.

Omdat het gebruik van één enkele controller een single point of failure oplevert, kunnen meerdere controllers gebruikt worden. De uitwisseling van gegevens, transacties en status updates tussen deze controllers gebeurt door middel van JGroups [61]. JGroups biedt een framework waarmee communicatie tussen een groep mogelijk wordt gemaakt. Er worden een aantal verschillende protocollen ondersteunt, zoals IP-multicast maar ook TCP en UDP. Afhankelijk van de benodigde betrouwbaarheid en de opzet van de groep kunnen verschillende protocollen worden gebruikt waarmee de communicatie wordt verzorgd. Sequoia gebruikt het reliable total order protocol van JGroups, waarmee berichten bij alle deelnemers betrouwbaar en in dezelfde volgorde worden afgeleverd. De total order wordt bereikt door het gebruik van een sequencer of het gebruik van een token. De betrouwbaarheid wordt door middel van het herzenden van berichten bereikt. Dit betekent dat elke deelnemer dezelfde berichten ontvangt, tenzij deze uitvalt of niet meer bereikbaar is. In dat geval wordt de deelnemer uitgesloten van de groep, totdat deze weer verbinding kan maken. Na het herstel worden de berichten gesynchroniseerd of dient een beheerder een aantal stappen te ondernemen, afhankelijk van de toepassing die JGroups gebruikt. Op dit moment wordt Appia [62] als alternatief ontwikkeld voor JGroups om als groep communicatie systeem te gaan dienen voor Sequoia.

Sequoia implementeert een systeem genaamd RAIDb, een variant op het bekende RAID systeem voor harde schijven. RAIDb staat voor Redundant Array of Independent Databases en biedt zowel een stripe als een mirror aan. Een stripe kan in deze worden opgevat als een gepartitioneerde database, een

mirror biedt data redundantie. Voor een High Availability systeem is het toepassen van stripes geen optie, aangezien daarmee de volledige database wegvalt bij het uitvallen van één node uit de stripe. Er is dan namelijk geen sprake van redundante data. Het nesten van controllers is mogelijk bij Sequoia, waardoor verschillende stripes en mirrors gebruikt kunnen worden voor het beter laten schalen van het systeem. Op deze manier kunnen bepaalde gegevens die veel worden gelezen over twee stripes worden verdeeld die door middel van een mirror dan ook redundant worden opgeslagen. Hoewel dit mogelijk is, neemt de complexiteit snel toe bij toepassing van meerdere niveau's in een Sequoia configuratie.

A.6.2. HA-JDBC

HA-JDBC [29] staat voor High-Availability JDBC en biedt een verschillende aanpak ten opzichte van de Sequoia JDBC techniek. Bij HA-JDBC wordt geen tussenliggende controller gebruikt, maar worden JDBC requests gelijk doorgestuurd naar de onderliggende JDBC drivers. Bij HA-JDBC is alleen een directe mirror mogelijk van de onderliggende database. De gebruikte databases dienen gelijk te zijn in tegenstelling tot Sequoia waar verschillende database producten gebruikt kunnen worden. Bij het herstellen van een gefaalde node wordt de database brute force gesynchroniseerd, dat betekent dat de hele database wordt gekopieerd, ook al is een groot deel van de database met dezelfde data gevuld.

Behalve de website van het HA-JDBC project [29] is verder nauwelijks informatie te vinden over gebruikers van deze technologie of eventuele problemen bij het gebruik. Het toepassen van HA-JDBC brengt dan ook onzekere risico's met zich mee.

A.6.3. Continuent m/cluster, p/cluster, uni/cluster

Continuent [43] is de nieuwe naam van Emic Networks, een belangrijke speler in het aanbieden van cluster functionaliteit voor MySQL. p/cluster en uni/cluster zijn commerciële versies van de Sequoia opensource software. p/cluster is geoptimaliseerd voor PostgreSQL, uni/cluster biedt middleware software aan die met verschillende database producten samen kan werken.

m/cluster 2005 [41] is Continuent's product voor MySQL. m/cluster is een stuk middleware wat tussen de database en de applicatie inzit. Tegenover de applicatie doet m/cluster zich voor als de MySQL database, zodat de applicatie niet aangepast hoeft te worden. De databasesoftware wordt zodanig aangepast dat de middleware op de standaard poort van de database mag draaien. Er wordt gebruik gemaakt van een Single Database Image als variant op de Single System Image zoals beschreven in 3.5.1, op basis van een shared ip adress. Elke node in het cluster heeft datzelfde ip adres, zodat het als een MySQL database optreedt. Voor de applicatie is het niet nodig te weten naar welke node geconnect wordt, evenals het aantal nodes dat het cluster bevat.

Bij het binnenkomen van een request van de applicatie wordt deze request gedistribueerd naar de verschillende nodes die zich binnen het cluster bevinden. Hiervoor wordt gebruik gemaakt van een proprietair groep communicatie systeem. Berichten worden allemaal in dezelfde volgorde bij alle nodes afgeleverd of helemaal niet. Het uitvallen van netwerkverbindingen voor korte tijd is geen probleem door het gebruik van queues om transacties te onthouden voor een van de nodes. m/cluster is specifiek voor een LAN ontwikkeld, het toepassen in een WAN omgeving is nog in ontwikkeling. Een WAN wordt op het moment niet ondersteund, waardoor het gebruik van meerdere locaties geen optie is bij m/cluster. Het combineren van m/cluster met MySQL replicatie is wel mogelijk, indien beperkt dataverlies door de asynchroniteit van MySQL replicatie geen probleem is.

Halverwege 2006 moet een nieuwe versie van m/cluster uitkomen, genaamd m/cluster 2006. Deze versie zal gebaseerd zijn op de Sequoia technologie, waarmee Continuent volledig overstapt naar Sequoia gebaseerde oplossingen voor zijn cluster middleware. Het ontwikkelen van specifieke functies en features voor MySQL is met Sequoia ook mogelijk, maar daarvoor hoeft de MySQL server niet meer te worden aangepast, zoals in de oude 2005 versie. Hierdoor kunnen de productreleases onafhankelijk van de MySQL releases gedaan worden en is de ontwikkeling nauwelijks nog afhankelijk van de MySQL ontwikkeling.

.

A.7. Cluster Management

Cluster Management software maakt het mogelijk om services over meerdere systemen te verdelen. Het gebruik van cluster management software biedt de mogelijkheid om meerdere systemen te gebruiken voor een service, zodat de beschikbaarheid verhoogd kan worden. Ook het draaien van services die niet genoeg hebben aan de performance van een systeem wordt daarmee mogelijk. De cluster management software verzorgt het overnemen van services als fouten optreden of onderhoud moet worden gepleegd. Een belangrijke functie van de software is het monitoren van de systemen en services op fouten. Deze software is niet specifiek ontwikkeld voor database software en kan ook worden toegepast voor andere services als webservers, email en applicatieservers. Database software die geen eigen monitoring aanbieden kunnen door middel van cluster management software van monitoring en failover capacitieiten worden voorzien. Voor deze failover dienen dan wel functies te zijn ingebouwd in de database die door de monitor software gebruikt kunnen worden.

Dit hoofdstuk biedt een klein overzicht van de verschillende cluster management software die beschikbaar is op de markt en welke in combinatie gebruikt kunnen worden met de eerder besproken database software.

A.7.1. Linux Virtual Server

Linux Virtual Server [63] (LVS) is een open-source softwarepakket waarmee een aantal verschillende servers beschikbaar worden gemaakt als één grote server door middel van een load-balancer richting de gebruikers.

Virtual Server biedt drie verschillende technieken aan om de verschillende servers via één machine toegankelijk te maken. Dit zijn Network Adress Translation (NAT), IP tunnels of directe routering. Afhankelijk van het aantal servers en of alles lokaal staat of niet, kan een van de technieken worden gekozen. NAT is goed toepasbaar bij kleine clusters die in een privaat netwerk staan. IP tunnels zijn bruikbaar bij WAN links en als de servers niet in hetzelfde netwerk segment zich bevinden. Directe routering is alleen bruikbaar indien de load balancer en servers in hetzelfde netwerk segment staan, maar heeft niet de tunnel overhad van IP Tunneling.

LVS biedt een aantal monitoring systemen waarmee verschillende services in de gaten gehouden kunnen worden. De load balancer kan dan connecties naar andere systemen overzetten indien een van de systemen uitvalt. Omdat de load balancer (LB) een single point of failure is, is het noodzakelijk om een tweede load balancer neer te zetten die als stand-by functioneert. Indien de eerste LB uitvalt, kunnen zo de connecties worden overgenomen. Software uit het Linux-HA project wordt veelal gecombineerd met de software van de Linux Virtual Server. LVS richt zich voornamelijk op het aanbieden van services met meerdere machines in een cluster, waarbij Linux-HA zich richt op de beschikbaarheid van de aangeboden services.

Om LVS samen met een database in cluster opstelling te gebruiken, dient de database wel cluster mogelijkheden te hebben. De database software op de

verschillende machines dient wel over dezelfde informatie te beschikken. Indien meerdere nodes tegelijkertijd gebruikt moeten worden, dan dient de database software dit wel te ondersteunen. Een actief-passief database cluster systeem kan dan ook beter gebruik maken van Linux-HA dan van LVS.

A.7.2. Linux-HA

Linux-HA [64] is het open source Linux High Availability Project. Sinds 1999 vindt het project zichzelf geschikt voor gebruik in mission-critical productie omgevingen. Het project heeft als doel om een High Availability oplossing te implementeren die reliablility, availability en serviceability (RAS) stimuleert in een Linux omgeving.

Linux-HA gebruikt het Heartbeat protocol om te detecteren of nodes nog actief zijn en de communicatie nog in tact is. Indien problemen optreden kan door middel van Heartbeat de gespecificeerde services worden overgeheveld naar een andere node die de services dan kan overnemen. Andere opties in Linux-HA zijn het monitoren van resources en resource fencing. Resource fencing is het afschermen van resources voor bepaalde systemen, bijvoorbeeld storage, als niet zeker is of een systeem goed functioneert. Door middel van STONITH (Shoot The Other Node In The Head) devices kunnen nodes worden uitgeschakeld, indien geen zekerheid kan worden verkregen over de status. Dit is belangrijk indien een netwerk partitie optreedt en niet twee nodes tegelijkertijd dezelfde service mogen aanbieden. Het systeem dat als eerste de STONITH bereikt sluit daarmee het andere systeem af.

Een ander onderdeel van het Linux-HA project is het ondersteunen van gedeelde opslag voor de services. Hier zijn verschillende oplossingen voor, die door externe software wordt geleverd. Voorbeelden zijn DRBD, NFS, CIFS en GFS.

Indien Linux-HA met een database samen wordt gebruikt, dient een monitor te worden gebruikt die de database kan monitoren. Afhankelijk van de gebruikte database kan de data door middel van replicatie of shared-storage op het stand-by systeem beschikbaar worden gemaakt. Indien de primaire database of het systeem uitvalt, kan door de monitor het stand-by systeem worden ingeschakeld.

Linux-HA wordt veelvuldig gebruikt volgens de succes verhalen [65] en wordt gezien als een geduchte concurrent voor commerciële pakketen.

A.7.3. Microsoft Cluster Services

Microsoft Cluster Services (MSCS) [55] biedt de mogelijkheid om meerdere Windows Servers te combineren tot een cluster. MSCS is sinds de Enterprise editie van NT4 beschikbaar en biedt alleen ondersteuning voor Windows. MSCS is een standaard onderdeel van de Windows Server versies en brengt geen extra kosten boven een Windows Datacenter licentie met zich mee.

MSCS biedt zowel een Cluster Server als een Network Load Balancer (NLB). De NLB kan gebruikt worden om webservers en firewalls van High Availability te

voorzien en fungeert daarmee als een software load balancer. De cluster server is bruikbaar voor databases en email servers. In combinatie met SQL Server biedt Microsoft daarmee een volledige oplossing aan. MSCS ondersteunt tussen twee en acht verschillende nodes. Omdat niet alle cluster services op meer dan een node kunnen draaien is het mogelijk verschillende services te combineren zodat de hardware optimaal gebruikt kan worden. Zowel SAN storage als lokale storage met replicatie wordt door het MSCS ondersteund [66]. Ondersteuning van meerdere geografische gescheiden locaties is aanwezig, zodat de uitval van een locatie geen uitval van de volledige voorziening hoeft te betekenen.

A.7.4. Red Hat Cluster Suite

De Red Hat Cluster Suite [67] biedt evenals MSCS van Microsoft zowel een ip load balancer als een Cluster Manager aan. Red Hat Cluster Suite draait op het Red Hat Linux Enterprise edities 3 en 4 en heeft daarmee een zelfde soort integratie als bij Microsoft het geval is. Wel dient voor de Cluster Suite extra betaald te worden.

De Cluster Manager biedt hoge beschikbaarheid door gebruik te maken van applicatie failover, zoals dat ook gebruikt wordt in MSCS, Sun Cluster en HP TruCluster [68]. De Cluster Manager ondersteunt maximaal 8 nodes en heeft een shared-storage array nodig om te functioneren. Alle acht nodes dienen op deze storage array te worden aangesloten, zodat de nodes de data van andere nodes kunnen lezen, indien een failover noodzakelijk is.

A.7.5. Overig

Zowel IBM als HP bieden een eigen implementatie voor respectievelijk hun AIX en HP-UX besturingssystemen van hun cluster management software. Bij IBM heet dit High Availability Cluster Multi-Processing (HACMP). IBM biedt ook een HAGEO oplossing aan, waarmee een geografisch gescheiden systeem mogelijk wordt.

RSF1 is een clustermanagement systeem van high-availability.com waarmee verschillende services van high availabilty kunnen worden voorzien. Het bestaat uit een aantal verschillende monitoring agents die de services monitoren. Als er fouten optreden of nodes utivallen, kunnen de services door de cluster management software naar andere nodes worden overgeheveld en daar worden herstart. Voor MySQL bijvoorbeeld biedt RSF-1 een automatische failover oplossing aan. MySQL zelf biedt alleen replicatie software die de data replicatie op zich neemt. RSF-1 biedt monitor agents voor databases, applicaties en infrastructuur zaken, als webservers, email, LDAP en Radius en is dus goed inzetbaar als de software wel in replicatie van data voorziet maar niet in het automatisch overnemen van taken.

Een andere grote naam is Veritas [115], nu van Symantec, dat gebruikt wordt door eBay. Zij bieden niet alleen cluster management, maar ook replicatie en backup software. Ook Sun biedt een Cluster aan [116].

Het voert hier helaas te ver om al deze producten apart te behandelen, meer informatie is op de respectievelijke productpagina's verkrijgbaar.

APPENDIX B. PRODUCT OVERZICHT

Specificaties / Producten	Oracle FailSafe	Oracle Data Guard	Oracle RAC	IBM DB2 HADR	IBM DB2 SQL Replication	IBM DB2 Q Replication	Microsoft SQL Log Shipping
Onderdeel van	Oracle 10g SE en EE	Oracle 10g EE	Oracle 10g SE en EE (optie)	IBM DB2 8.2 ESE, WSE	IBM DB2 8.2 ESE, WSE	IBM DB2 8.2ESE, WSE	MS SQL Server 2005 Standard en Enterprise
Gericht op	Beschikbaarheid	Beschikbaarheid/ data redundantie	Performance cluster	Beschikbaarheid / data redundantie	Data loading	Data replicatie/backups	Data replicatie/backups
Replicatie	Shared storage	Synchroon/ Asynchroon	Shared storage	Synchroon/ Asynchroon	Asynchroon	Asynchroon	Asynchroon
Technologie	Cluster	Master/Slave	Cluster	Actief/Passief	nvt	Publish/ subscribe	Master/slave
Auto-failover	Ja	Ja	Ja	Ja, met externe monitor software	Nee	Nee	Nee
Aantal ondersteunde/benodigde systemen	Afhankelijk van MSCS	tot 9 standby's	4(SE), 64(EE)	2 (Primary + Standby)	NVT	2 of meer	2 of meer
Aanschafkosten per cpu	SE $15000 , EE $40000	EE $40000	SE $15000 , EE $40000	ESE: $33125 WSE: $7500	ESE: $33125 WSE: $7500	ESE: $33125 WSE: $7500	Standard $6000, Enterprise $25000

Specificaties / Producten	Oracle FailSafe	Oracle Data Guard	Oracle RAC	IBM DB2 HADR	IBM DB2 SQL Replication	IBM DB2 Q Replication	Microsoft SQL Log Shipping
Trialversies	30 dagen	30 dagen	30 dagen	90 dagen	90 dagen	90 dagen	180 dagen
Platform	Windows	Windows, Linux en Unix	Windows, Linux en Unix	Windows, Linux,Unix, AIX, HP-UX, Solaris	Windows, Linux,Unix, AIX, HP-UX, Solaris	Windows, Linux,Unix, AIX, HP-UX, Solaris	Windows
Speciale eisen	Windows Server met MSCS ondersteuning		Shared Storage hardware				
Commentaar		Combinatie met RAC mogelijk					
Bronnen	[89]	[88],[89]	[89]	[10],[91]	[92]	[92]	[6],[7]

Producten / Specificaties	Microsoft SQL Database Mirroring	Microsoft SQL Peer-to-peer replication	Microsoft SQL n-Way Cluster	MySQL Cluster	MySQL Replicatie	PostgreSQL Slony-I	PostgreSQL pgCluster
Onderdeel van	MS SQL Server 2005 Standard en Enterprise	MS SQL Server 2005 Standard en Enterprise	MS SQL Server 2005 Standard en Enterprise	MySQL 5.1	MySQL5.1	PostgreSQL plugin 8.1	PostgreSQL plugin 8.1
Gericht op	Beschikbaarheid/ data redundantie	Data replicatie/backups	Beschikbaarheid	Beschikbaarheid performance cluster	Data replicatie	Data replicatie	Beschikbaarheid
Replicatie	Synchroon/ Asynchroon	Asynchroon	Shared storage	Synchroon	Asynchroon	Asynchroon	Synchroon
Technologie	Actief/Passief	Publish/ subscribe	Cluster	Cluster	Master/Slave	Master/Slave	Multimaster
Auto-failover	Ja, met witness	Nee	Ja	Ja	Ja, met externe monitor software	Ja, met externe monitor software	Ja
Aantal ondersteunde/benodigde systemen	2 zonder witness, 3 met	2 of meer	Afhankelijk van MSCS	3 of meer	2 of meer	2 of meer	2 of meer
Aanschafkosten per cpu	Standard $6000, Enterprise $25000	Standard $6000, Enterprise $25000	Standard $6000, Enterprise $25000	Open-source	Open-source	Open-source	Open-source
Trialversies	180 dagen	180 dagen	180 dagen	onbeperkt	onbeperkt	onbeperkt	onbeperkt

Producten / Specificaties	Microsoft SQL Database Mirroring	Microsoft SQL Peer-to-peer replication	Microsoft SQL n-Way Cluster	MySQL Cluster	MySQL Replicatie	PostgreSQL Slony-I	PostgreSQL pgCluster
Platform	Windows	Windows	Windows	Linux	Windows, Linux	Windows, Linux	Linux
Speciale eisen			Windows Server met MSCS ondersteuning				
Commentaar							
Bronnen	[6],[7]	[6],[7]	[6],[7]	[20],[11]	[90]	[60]	[24]

Specificaties / Producten	Onderdeel van	Gericht op	Replicatie	Technologie	Auto-failover	Aantal ondersteunde/benodigde systemen	Aanschafkosten per cpu
PostgreSQL Postgres-R	PostgreSQL plugin voor 6.2	Data replicatie	Synchroon	Master/Slave	Nee	2 of meer	Open-source
PostgreSQL Slony-II	PostgreSQL plugin	Data replicatie	Synchroon	Multimaster	?	?	Open-source

Specificaties / Producten	Trialversies	Platform	Speciale eisen	Commentaar	Bronnen
PostgreSQL Postgres-R	onbeperkt	Linux			[27]
PostgreSQL Slony-II	onbeperkt	?		Nog geen werkende versie	[25]

APPENDIX C. ILSE MEDIA INTERVIEW

Ilse Media B.V. heeft ongeveer 40 websites onder haar beheer. Een aantal van deze websites bieden informatie aan een aantal miljoen gebruikers per dag, zoals de nieuwssite nu.nl en de linkspagina startpagina.nl. Ook de Ilse zoekmachine wordt veel gebruikt, maar deze wordt door Ilse niet als website beschouwd maar meer als service. Een overzicht van de websites van Ilse en de bezoekersaantallen staat op [111]. Om te zorgen dat al deze gebruikers continu de hele dag de websites kunnen bezoeken, heeft Ilse een aantal maatregelen getroffen om de websites beschikbaar te maken en houden.

Ilse Media gebruikt vier verschillende niveaus om de beschikbaarheid van hun websites te behalen. Niveau 1 is het laagste niveau waar op basis van best effort de beschikbaarheid van de website wordt geregeld. Niveau 4 geeft een 99.99% garantie op beschikbaarheid. Alle beschikbaarheidcijfers worden berekend per maand. Dit betekent dat de maximale continue downtijd beperkt wordt ten opzichte van een grotere periode als een jaar. De tabel geeft een overzicht van de cijfers en de soort maatregelen die bij het niveau worden toegepast.

Beschikbaarheidsniveau	Beschikbaarheidscijfer	Soort maatregelen
Niveau 1	99%, Best Effort	Spare componenten beschikbaar
Niveau 2	99.8%	Belangrijke componenten dubbel uitvoeren
Niveau 3	99.9%	Alles dubbel uitvoeren op één datacenter
Niveau 4	99.99%	Alles dubbel uitvoeren, inclusief dubbel datacenter

Een website of applicatie wordt ingedeeld in een van de vier niveaus aan de hand van de soort applicatie, de mogelijke imagoschade die optreedt bij uitval en het beschikbare budget. Hoewel de inkomstenbron uit advertenties voor Ilse belangrijk is, is imagoschade de grootste drijfveer om haar websites van een bepaald beschikbaarheidniveau te willen voorzien.

Websites die zich in het ontwikkelstadium bevinden en/of waarbij de interesse nog beperkt is, beginnen bij Ilse standaard in niveau 1. Naarmate het aantal bezoekers stijgt en de site dus belangrijker wordt voor Ilse zal deze stijgen in het beschikbaarheidniveau. Dat betekent dat de risico's van uitval worden verkleind. Het is niet zo dat sites in niveau 1 vaker uitvallen dan in een hoger niveau, maar het risico op uitval wordt met de stijging in het niveau steeds kleiner door de grotere mate van redundantie. De belangrijkste sites nu.nl en startpagina.nl draaien op niveau 4, de Ilse zoekmachine draait door de grote behoefte aan hardware op niveau 3. Doordat zeer veel machines nodig zijn voor de zoekmachine is het niet rendabel om deze op twee datacenter locaties

dubbel uit te voeren. Het eventuele verlies van functionaliteit indien een volledig datacenter uitvalt is acceptabel voor Ilse gezien de kleine kans op het optreden van zo'n volledig datacenter uitval. Zowel nu.nl als startpagina.nl kunnen in zo'n geval zonder problemen blijven functioneren, omdat een tweede datacenter alle taken dan overneemt.

Alle geserveerde pagina's worden statisch aangeleverd door het Content Management Systeem (CMS) dat in het back-office draait. Met de grote aantallen bezoekers is het niet mogelijk om een dynamisch CMS systeem als frontend te gebruiken. Daardoor worden alle pagina's vanuit het CMS gegenereerd en daarna vanuit een statische omgeving geserveerd richting de gebruikers. Dit betekent dat als het CMS systeem uitvalt, de website gewoon blijft functioneren. Het toevoegen van nieuwe items is dan niet goed meer mogelijk, maar de bezoeker zal geen problemen zien op de bezochte website.

Omdat bijna alles statisch is, is het gebruik van databases beperkt. De databases worden gerepliceerd in een master-slave setup, waarbij de overschakeling van de slave, indien de master uitvalt, handmatig gebeurt. Omdat automatische overschakeling risico's meebrengt voor de data en de monitoring van de database goed op orde is, is het handmatig overschakelen een goede optie voor Ilse. De data die in de databases wordt bewaard hoeft slechts een beperkte tijd te worden bewaard. Veelal is een termijn van vier weken voldoende. Daardoor is opslagcapaciteit en het maken van backups geen probleem.

Belangrijk voor sites als nu.nl is het binnenhalen van de data die nodig is voor de website en deze lokaal opslaan, zodat als aanbieders van de data uitvallen dit geen gevolgen heeft voor de content op de websites van Ilse. Hierbij gaat het bijvoorbeeld om ANP informatie, file en weerberichten. Doordat de data wordt binnengehaald kan invloed worden uitgeoefend op de beschikbaarheid ervan, beschikbaarheid van externe bronnen is niet te garanderen. Op deze manier is het mogelijk eventueel oude data te tonen in plaats van lege velden in de website, indien de leverancier van de data niet kan leveren.

Het gehele serverpark draait op ongeveer 75 machines waarop door middel van vmware meerdere virtuele images draaien waarin het OS en de applicatielaag hun werk doen. De hosting provider biedt twee opties als leverancier van de machines, zowel Dell als HP worden ondersteund. Bij Ilse is voor HP gekozen. Dit biedt als grote voordeel dat de hosting provider ook makkelijk onderhoud kan plegen als een van deze machines problemen heeft.

Belangrijk bij de keuze voor een hostingprovider was de overcapacitieit die deze provider kan leveren. Indien er problemen optreden is men niet afhankelijk van de leverancier van het product, maar kan een aanwezig reserve onderdeel worden gebruikt. Dit zorgt ervoor dat systemen snel vervangen kunnen worden en dat niet op een leverancier gewacht moet worden, waarvan niet gegarandeerd is dat deze het probleem binnen een bepaalde tijd heeft opgelost. De enige garantie die gegeven wordt is dat deze leverancier binnen een bepaalde tijd onsite aanwezig is. Er is in de gebruikte datacenters van de provider een pool van systemen beschikbaar die ingezet kunnen worden indien

een van de eigen systemen niet meer correct functioneert. Hierdoor is binnen een zeer korte tijd een reparatie mogelijk.

Omdat het lastig is om meerdere providers samen te laten werken, vooral als er problemen optreden is besloten de volledige architectuur bij één hosting provider onder te brengen. De voordelen met betrekking tot beheer, onderhoud en het meedenken over problemen wegen op tegen de afhankelijkheid die het kiezen voor één provider met zich meebrengt. Hoewel het in eigen beheer nemen van de systemen deze afhankelijkheid niet heeft, is de benodigde expertise dusdanig specialistisch dat dit beter over kan worden gelaten aan externe partijen die zich specialiseren op deze gebieden. Waar mogelijk heeft Ilse getracht alternatieven te vinden voor een deel van de afhankelijkheid. Zo zijn de eigen ip ranges behouden, zodat als de hosting provider in financiele problemen raakt, Ilse nog mogelijkheden heeft om het serverpark te verhuizen en de services voort te zetten.

Voordat gekozen werd voor een externe hosting partij is door Ilse ook berekend hoeveel medewerkers nodig zouden zijn om de systemen zelf te beheren en onderhouden. Dit zou neerkomen op minimaal 6 FTE’s voor een 24x7 onderhoudsteam. Omdat dit niet tot de business van Ilse hoort en ook niet wil laten horen, is uitbesteden de beste optie.

Belangrijk bij het samenwerken met een hosting provider is het afspreken wie waarvoor verantwoordelijk is en welke zaken belangrijk genoeg zijn om voor te bellen of wat per email afgehandeld kan worden. De scheiding tussen de verantwoordelijkheden ligt tussen het operating system en de applicatielaag. Zowel de hardware, communicatie als besturingsysteem liggen bij de hosting provider, Ilse is verantwoordelijk voor de applicatielaag. Wel is er enige overlap tussen deze twee lagen, waardoor beide partijen verantwoordelijk zijn. Met betrekking tot communicatie is het aftasten welke problemen ernstig zijn en welke tot gewone werktijden kunnen wachten. Zo zijn niet alle problemen ernstig genoeg om daar s’ nachts mensen voor wakker te bellen, maar in sommige gevallen kan dit wel noodzakelijk zijn. Het onderling afstemmen kan een hoop narigheid voorkomen in verband met het niet melden van grote problemen of de continue melding van kleine probleempjes.

Backup van de data wordt online geregeld door de hosting partij. Deze maakt op bepaalde afgesproken momenten een volledige backup van het systeem, die dan bij problemen terug gezet kunnen worden. Dit backup systeem maakt gebruik van harde schijven, zodat een snelle restore mogelijk is. Software waar speciale acties nodig zijn, zoals databases worden door middel van een database dump gebackupd, wat als extra service wordt aangeboden bovenop de volledige system backup. Afhankelijk van de belangrijkheid van de data kan besloten worden vaker als 1x per 24 uur een backup te maken. Dit brengt dan extra kosten met zich mee voor de extra benodigde schijfruimte die de backup inneemt, maar betaalt zich terug als een restore moet worden uitgevoerd. Hoe vaker een backup wordt uitgevoerd, hoe minder dataverlies optreedt als een restore nodig is.

Om de vele bezoekers te kunnen hanteren, wordt door Ilse gebruik gemaakt van hardware load balancers en caching machines. Deze zorgen ervoor dat de verbindingen over de verschillende webservers worden verdeeld en dat content die niet veranderd is gelijk vanuit de cache geserveerd kan worden. Op deze manier wordt de druk van de webservers afgehaald. Indien alles uit zou vallen, is er nog een fall-back mogelijk naar het KPN Cache systeem. Dit systeem cached de gegevens van de websites van Ilse en levert deze uit, als de systemen van Ilse zelf dit niet meer kunnen. Dit biedt dan weliswaar verouderde informatie, maar er treedt minder imagoschade op dan als de volledige website uit de lucht zou zijn.

Om alle systemen in de gaten te houden wordt gebruik gemaakt van het monitoring systeem van de hosting partij. Aangezien deze ook verantwoordelijk is voor de hardware en het OS, dient de partij deze zelf te monitoren zodat problemen in een zo vroeg mogelijk stadium verholpen kunnen worden en niet pas nadat er problemen door Ilse zijn gemeld.

Ilse heeft ervaring met het uitvallen van meerdere componenten, het optreden van storingen en het testen van de opstelling en heeft in het afgelopen jaar nog geen moment gehad waarop de websites niet beschikbaar waren voor de bezoekers.

Statistieken:
Nu.nl : 1 miljoen bezoekers per dag, 7 miljoen pageviews
Startpagina.nl: 2 miljoen bezoekers per dag, 6 miljoen pageviews
Gebruikt gemiddeld 250mbit/s bandbreedte op 1gbit capaciteit richting AMS-IX voor websites, exclusief streaming content

Best practices volgens Ilse :

- Geef de infrastructuur in handen van één organisatie, waarmee goede afspraken te maken zijn
- Zorg voor een zo kort mogelijke periode waarover de SLA berekend wordt. Een maximale downtijd berekend per maand geeft meer zekerheid dan berekend per jaar.
- Indien de expertise niet bij je eigen business hoort, besteedt het dan uit aan specialisten. De extra kosten wegen op tegen de problemen die worden voorkomen.
- Door op een provider met overcapaciteit te vertrouwen, hoeft niet te worden gewacht op levertijden en garantieafhandeling van de leverancier van de producten.
- Kijk niet alleen naar gederfde inkomsten voor een inschatting van de kosten, maar ook naar de imagoschade die bij uitval optreedt. Weeg het verkleinen van de risico’s goed af tegen de toenemende kosten.
- Voor alle data die van externe partijen komt en belangrijk is voor de website omdat deze zichtbaar is, zoals file informatie en weerberichten, is het verstandig deze direct lokaal op te slaan, zodat de beschikbaarheid ervan niet afhankelijk is van derden, maar daar zelf invloed op uitgeoefend kan worden. Zo wordt voorkomen dat lege gaten op de website ontstaan.

- Redundantie van alle onderdelen op alle lagen (omgeving, netwerk, hardware, software en applicatie) geeft de beste garantie voor de beschikbaarheid, alleen nooit volledige garantie.
- Bezuinigen op het aantal backups brengt veelal meer kosten met zich mee dan de besparing oplevert, doordat data verloren gaat bij disk problemen.

APPENDIX D. TESTOPSTELLING

Dit appendix bevat de verschillende testcases voor de producten IBM HADR (D.1), Sequoia (D.2) en MySQL Cluster (D.3). In D.4, 5 en 6 staat een beschrijving van de cliëntsoftware, applicatie en database server. De benodigde hard- en software staat in D.7 De snapshots waarnaar verwezen wordt in de cases staat in D.8.

D.1. Product 1 – IBM HADR

Omdat het IBM DB2 HADR systeem slechts een beperkt aantal componenten bevat, slechts één stuk databasesoftware per locatie, is het aantal componenten wat uit kan vallen ook beperkt. Hierdoor zijn slechts een beperkt aantal testcases benodigd.

D.1.1. Testcases

Testcase IBM-01	
Doel van de test:	Aantonen dat beide locaties tegelijk gebruikt kunnen worden.
Wat wordt er getest :	1. De uitgangssituatie wordt ingesteld. 2. De lees en schrijfapplicaties worden op locatie A aangezet 3. De lees en schrijfapplicaties worden op locatie B aangezet 4. Gecontroleerd moet worden of beide kunnen functioneren
Uitgangssituatie	Het systeem bevindt zich in snapshot 1 en is operationeel
Motivatie	De database op locatie b bevindt zich als standby database altijd in roll-forward mode van de logs die binnenkomen van de primary database. Dit betekent dat deze database niet beschikbaar is voor clients om acties op uit te voeren. Voordat deze gebruikt kan worden dient deze eerst een fail- of takeover uit te voeren of in standaard (niet HADR) modus gebracht te worden. Deze test toont aan dat de database op locatie B niet gebruikt kan worden.
Verwachte resultaten	Locatie B zal niet functioneren op de database in B.

Testcase IBM-02	
Doel van de test:	Aantonen dat de slave de rol van de master overneemt als deze uitvalt. De applicatie kan op de slave worden voortgezet
Wat wordt er getest :	1. De uitgangssituatie wordt ingesteld. 2. De lees en schrijfapplicaties worden op locatie A aangezet 3. De master database node wordt uitgezet 4. Wat gebeurt er, wordt deze automatisch overgenomen door de slave?
Uitgangssituatie	Het systeem bevindt zich in snapshot 1 en is operationeel
Motivatie	Deze test test de basisfunctionaliteit van een Master Slave systeem die het HADR concept aan zou moeten bieden. In het optimale geval neemt de slave het automatisch van de master over als het master systeem uitvalt. Indien dit niet automatisch kan is goede monitoring van een externe applicatie en het overschakelen door middel van scripting of een externe applicatie nodig om niet afhankelijk te zijn van mensen voor de overschakeling. Door deze test wordt aangetoond of makkelijk overgeschakeld kan worden en of zaken automatisch worden opgelost.

Verwachte resultaten	Het slave systeem neemt de database rol van de master over.

Testcase IBM-03	
Doel van de test:	Aantonen dat de uitval van de slave database geen gevolgen heeft voor de master en de applicatie.
Wat wordt er getest :	1. De uitgangssituatie wordt ingesteld. 2. De lees en schrijfapplicaties worden op locatie A aangezet 3. De slave database node wordt uitgezet 4. Blijft de master doordraaien? 5. De slave wordt na 5 min weer ingeschakeld, wordt deze automatisch gesynchroniseerd?
Uitgangssituatie	Het systeem bevindt zich in snapshot 1 en is operationeel
Motivatie	Deze test toont aan of de uitval van de slave database gevolgen heeft voor het master systeem. Indien het master systeem niet kan blijven functioneren bij uitval van de slave, dan heeft dit grote risico's voor de beschikbaarheid van het systeem. Dat zal dan betekenen dat de slave altijd online moet zijn en dat er geen onderhoud aan het systeem gepleegd kan worden.
Verwachte resultaten	De master ondervindt geen gevolgen van het uitvallen van de slave en de slave wordt na inschakeling automatisch gesynchroniseerd.

Testcase IBM-04	
Doel van de test:	Aantonen dat de uitval van de WAN link geen gevolgen heeft voor de applicatie of de master, onafhankelijk welke synchronisatie mode gebruikt wordt.
Wat wordt er getest :	1. De uitgangssituatie wordt ingesteld. 2. De lees en schrijfapplicaties worden op locatie A aangezet 3. Een van de synchronisatiemodes (Synchronous, near synchronous of Asynchronous) wordt ingesteld. 4. De WAN link wordt uitgeschakeld. 5. Blijft de master functioneren? 6. Herhalen met een andere synchronisatiemode
Uitgangssituatie	Het systeem bevindt zich in snapshot 1 en is operationeel
Motivatie	Het uitvallen van de WAN link is vergelijkbaar met het uitvallen van de slave database. Als internet als WAN link gebruikt wordt, dient het systeem er rekening mee te houden dat de verbinding niet altijd beschikbaar is. Als de master niet kan blijven functioneren bij uitval van de WAN link, wordt de beschikbaarheid van het systeem direct afhankelijk van de beschikbaarheid van de WAN link, die over het algemeen lager ligt dan die van systemen.
Verwachte resultaten	Het master systeem blijft functioneren, en de taken van de clients kunnen gewoon worden afgehandeld.

Testcase IBM-05	
Doel van de test:	Aantonen dat de data automatisch wordt gesynchroniseerd als de WAN link na een failure weer wordt hersteld.
Wat wordt er getest :	1. De uitgangssituatie wordt ingesteld. 2. De lees en schrijfapplicaties worden op locatie A aangezet 3. Een van de synchronisatiemodes wordt ingesteld. 4. De WAN link wordt uitgeschakeld. 5. De WAN link wordt na vijf minuten weer ingeschakeld. 6. Wordt de data automatisch gesynchroniseerd? Controleren door middel van logfiles. 6. Herhalen met een andere synchronisatiemode (als dat nuttig is, zie IBM-04)
Uitgangssituatie	Het systeem bevindt zich in snapshot 1 en is operationeel
Motivatie	Deze test toont aan of de data automatisch wordt gesynchroniseerd na een uitval van de WAN link. Indien geen automatische synchronisatie plaatsvindt, betekent dit dat na elke WAN uitval de database beheerder de standby zal moeten herstellen. Hoewel dit geen invloed hoeft te hebben op de beschikbaarheid, is dat niet positief voor het beheersbaarheidaspect. Het vaak uitvallen van de WAN link zal dan veel tijd van de beheerder kosten.
Verwachte resultaten	Het master systeem blijft functioneren, en de taken van de clients kunnen gewoon worden afgehandeld. Het slave systeem wordt automatisch gesynchroniseerd bij herstel.

Testcase IBM-06	
Doel van de test:	Aantonen dat een client automatisch wordt overgeschakeld naar de standby indien de standby een takeover uitvoert op de master
Wat wordt er getest :	1. De uitgangssituatie wordt ingesteld. 2. De lees en schrijfapplicaties worden op locatie A aangezet 3. De standby voert een takeover uit. 4. Wordt de client overgezet naar het standby systeem?
Uitgangssituatie	Het systeem bevindt zich in snapshot 1 en is operationeel
Motivatie	Indien de standby het master systeem wil overnemen, omdat op het master systeem onderhoud uitgevoerd moet worden, is het noodzakelijk voor de client om automatisch overgezet te worden indien geen zichtbare gevolgen hiervan ondervonden mogen worden. IBM heeft hiervoor Automatic Client Reroute bedacht, dat de gegevens van de standby ophaalt bij de connectie naar de master en deze gegevens gebruikt indien de master niet meer bereikbaar is. Deze test toont aan of deze functionaliteit standaard geactiveerd is of dat hiervoor extra stappen genomen moeten worden.
Verwachte resultaten	Het master systeem blijft functioneren, en de taken van de clients kunnen gewoon worden afgehandeld. Het slave systeem wordt automatisch gesynchroniseerd bij herstel.

D.1.2. Onderzoekscases

Onderzoekscase IBM-07	
Doel van de test:	Onderzoeken wat de invloed is van de HA features en latency op de performance van het DB2 systeem
Wat wordt er getest :	1. De uitgangssituatie wordt ingesteld. 2. Het systeem wordt ingesteld met een Standalone database, een HA database zonder WAN latency, HA met 50ms, HA met 300ms en HA met 1000ms 3. Voor elke setting worden 3 x 30 min DOTS benchmark gedraaid, waarna een gemiddelde over de drie keer wordt bepaald. 4. De HA wordt zowel ingezet op synchronous, near synchronous en asynchronous, zodat deze vergeleken kunnen worden en gezien kan worden of er invloed is op het aantal te verwerken transacties. 5. Hierna kunnen de verschillende settings vergeleken worden.
Uitgangssituatie	Het systeem bevindt zich in snapshot 1 en is operationeel
Motivatie	Door middel van de DOTS benchmark suite wordt bepaald wat de invloed is van de HA features en de latency die wordt geïntroduceerd met een WAN link. Op deze manier is vast te stellen of de performance verslechtert als gebruik wordt gemaakt van de High Availability features en in welke mate een afstand overbrugd kan worden die de latency veroorzaakt. Uit deze case komen verschillende getallen voor het aantal uitgevoerde query's, updates en inserts. Deze kunnen voor de verschillende opties vergeleken worden.
Verwachte resultaten	Resultaat van een uitgevoerde transactie duurt langer bij een latency van 300ms vergeleken met 50 ms of geen latency. Transactie response duurt bij een synchrone mode langer dan bij asynchrone mode.

D.2. Product 2 – Sequoia

Voor alle testcases geldt dat zowel met als zonder WAN-link simulatie getest wordt. De eerste test zal zonder simulatie worden uitgevoerd, waarbij na een succesvolle test de WAN simulatie zal worden toegevoegd. Het testen van de WAN simulatie bij een test waarbij de test op het LAN al niet succesvol blijkt, is niet zinvol en zal niet worden uitgevoerd. Bij testcases waar beide tests niet van toepassing zijn, staat dit er expliciet bij.

D.2.1. Testcases

Testcase SEQ-1	
Doel van de test:	Aantonen dat de uitval van één sequoia controller op de primaire locatie geen invloed heeft op de applicatie.
Wat wordt er getest :	1. De uitgangsituatie wordt ingesteld 2. De lees en schrijfapplicaties worden aangezet op de primaire locatie. 3. Een van de controllers op de primaire locatie wordt uitgeschakeld. 4. De resultaten van de schrijfapplicatie in de logfile wordt vergeleken met de data in de database. 5. Van de leesapplicatie wordt bekeken of geen foutmeldingen zijn opgetreden.
Uitgangssituatie	Het systeem bevindt zich in snapshot 2 en is operationeel.
Motivatie	Bij het gebruik van een enkele controller in de sequoia oplossing, is het systeem niet meer bruikbaar als deze controller uitvalt. Om dit probleem op te lossen, maakt Sequoia gebruik van zogenaamde “horizontale schaalbaarheid”, waardoor meerdere controllers gebruikt kunnen worden. Tussen deze controllers wordt informatie uitgewisseld om de databases synchroon te houden en uitval van een controller of backend op te vangen. Elke controller bevat een of meer eigen database backends. Deze test kijkt of die oplossing correct werkt.
Verwachte resultaten	De schrijfapplicatie ondervindt geen gevolgen van de controller uitval en kan zijn taak vervolgen. De leesapplicatie ondervindt geen gevolgen en kan rapporten opvragen.

Testcase SEQ-2	
Doel van de test:	Aantonen dat het systeem op beide locaties tegelijkertijd gebruikt kan worden
Wat wordt er getest :	1. De uitgangssituatie wordt ingesteld. 2. Op de primaire locatie (A) worden de lees- en schrijfapplicaties aangezet. 3. Op de secundaire locatie (B) worden de lees- en schrijfapplicaties aangezet. 4. Na 5 minuten worden de logfiles van beide vergeleken met de data in de database. De database dient nu van beide applicaties de data te bevatten.
Uitgangssituatie	Het systeem bevindt zich in snapshot 3 en is operationeel.
Motivatie	Deze test toont aan of beide locaties tegelijk gebruikt kunnen worden. Als dat zo is, dan hoeft er niet te worden omgeschakeld als een van de locaties uitvalt, waardoor geen interactie van een beheerder nodig is. Wel kan bij uitval van de connectie tussen beide locaties dit betekenen dat de systemen uit mekaar lopen. Dit wordt in Seq-7 getest.
Verwachte resultaten	De operaties worden succesvol verwerkt.

Testcase SEQ-3	
Doel van de test:	Aantonen dat de uitval van de WAN-link geen gevolgen heeft voor de applicatie op de primaire locatie.
Wat wordt er getest :	1. De uitgangssituatie wordt ingesteld. 2. De lees en schrijfapplicaties worden aangezet op de primaire locatie (A). 3. Na een paar minuten wordt de WAN-link uitgeschakeld. 4. Er moet gecontroleerd worden of de schrijfapplicatie kan blijven schrijven en of de leesapplicatie rapporten op kan vragen. 5. Wat gebeurt er met het systeem op de secundaire locatie? Kan deze gebruikt blijven worden of wordt deze uitgeschakeld?
Uitgangssituatie	Het systeem bevindt zich in snapshot 3 en is operationeel.
Motivatie	Deze test toont aan of het uitvallen van de WAN link gevolgen heeft voor de applicatie. Indien de WAN link uitvalt is het wel gewenst dat de applicatie netjes door kan werken.
Verwachte resultaten	De 2e locatie wordt uitgeschakeld, 1e locatie draait door.

Testcase SEQ-4	
Doel van de test:	Aantonen dat bij herstel van de WAN link de cluster configuratie automatisch hersteld wordt
Wat wordt er getest :	1. De uitgangssituatie wordt ingesteld. 2. De lees en schrijfapplicaties worden aangezet op de primaire locatie (A). 3. Na een paar minuten wordt de WAN-link uitgeschakeld. 4. Er moet gecontroleerd worden of de schrijfapplicatie kan blijven schrijven en of de leesapplicatie rapporten op kan vragen. 5. De WAN link wordt weer ingeschakeld. 6. Wordt het systeem automatisch gesynchroniseerd en zijn de gegevens die na de uitschakeling in stap3 zijn geschreven ook beschikbaar op de secundaire locatie? Controleren door middel van de logs.
Uitgangssituatie	Het systeem bevindt zich in snapshot 3 en is operationeel.
Motivatie	Indien de WAN link is uitgevallen is het nodig om bij herstel de cluster configuratie weer te herstellen, zodat nieuwe fouten gehanteerd kunnen worden. Deze test toont aan of dit automatisch gebeurt of dat een handmatige actie benodigd is om dit voor elkaar te krijgen.
Verwachte resultaten	Systeem herstelt de controller configuratie en synchroniseert de gegevens automatisch

Testcase SEQ-5	
Doel van de test:	Aantonen dat de uitval van één databasenode geen gevolgen heeft voor de applicatie.
Wat wordt er getest :	1. De uitgangssituatie wordt ingesteld. 2. De lees en schrijfapplicaties worden aangezet op de primaire locatie (A) 3. Een van de database nodes op de primaire locatie wordt uitgeschakeld. 4. Er moet gecontroleerd worden of de schrijfapplicatie kan blijven schrijven en of de leesapplicatie rapporten op kan vragen.
Uitgangssituatie	Het systeem bevindt zich in snapshot 2 en is operationeel.
Motivatie	Deze test toont aan of het uitvallen van een database backend gevolgen heeft voor de applicatie. Indien de uitval van een node de applicatie niet meer verder laat werken, is de opstelling niet high available.
Verwachte resultaten	De applicatie ondervindt geen gevolgen van de controller uitval en kan zijn taak vervolgen.

Testcase SEQ-6	
Doel van de test:	Aantonen dat het uit en aanzetten van de applicatie server geen gevolgen heeft voor de cluster configuratie
Wat wordt er getest :	1. De uitgangssituatie wordt ingesteld. 2. De lees en schrijfapplicaties worden aangezet op de primaire locatie (A). 3. Na 5 minuten wordt de applicatieserver op de primaire locatie uitgeschakeld. 4. Controle moet plaatsvinden of alle verwerkte transacties zich in de database bevinden. 5. Na het aanzetten van de applicatieserver kunnen de applicaties herstart worden en functioneert het systeem zonder problemen verder.
Uitgangssituatie	Het systeem bevindt zich in snapshot 2 en is operationeel
Motivatie	Deze test toont aan dat de oplossing onafhankelijk is van de gebruikte client. Indien het systeem afhankelijk is van de applicatieserver, kan niet eenvoudig een ander systeem worden toegevoegd of onderhoud worden gepleegd.
Verwachte resultaten	De applicatie kan na het herstarten van de applicatieserver zonder problemen verder werken. Er hoeven geen cluster aanpassingen te worden gedaan.

Testcase SEQ-7	
Doel van de test:	Aantonen dat beide locaties nog bruikbaar zijn als de WAN-link niet meer beschikbaar is
Wat wordt er getest :	1. De uitgangssituatie wordt ingesteld 2. De lees en schrijfapplicaties worden aangezet op de primaire locatie (A). 3. De lees en schrijfapplicaties worden aangezet op de secundaire locatie (B). 4. Na 5 minuten wordt de WAN link uitgeschakeld 5. Gecontroleerd moet worden of beide applicaties nog functioneren op beide locaties.
Uitgangssituatie	Het systeem bevindt zich in snapshot 3 en is operationeel
Motivatie	Deze test laat zien wat er gebeurt als de verbinding tussen beide locaties uitvalt en of beide locaties dan nog bruikbaar zijn. Indien dat zo is, dan dienen eventuele conflicten opgelost moeten worden als de verbinding weer hersteld wordt tussen beide locaties.
Verwachte resultaten	Een van beide locaties wordt uitgeschakeld.

Testcase SEQ-8	
Doel van de test:	Aantonen dat het tegelijkertijd uitvallen van een database en controller node in dezelfde groep, waarbij de controller beheer voert over de database node, geen gevolgen heeft voor de applicatie.
Wat wordt er getest :	1. De uitgangssituatie wordt ingesteld. 2. De lees en schrijfapplicaties worden aangezet op de primaire locatie (A) 3. Een van de database nodes op de primaire locatie wordt uitgeschakeld. 4. De controller van deze database wordt uitgeschakeld. 4. Er moet gecontroleerd worden of de schrijfapplicatie kan blijven schrijven en of de leesapplicatie rapporten op kan vragen.
Uitgangssituatie	Het systeem bevindt zich in snapshot 3 en is operationeel
Motivatie	In deze testcase vallen twee componenten uit dezelfde groep uit. Hiermee kan bekeken worden hoe fout resistent de oplossing is.
Verwachte resultaten	Applicatie kan doorwerken

Testcase SEQ-9	
Doel van de test:	Aantonen dat het tegelijkertijd uitvallen van een database en controller node in verschillende groepen geen gevolgen heeft voor de applicatie
Wat wordt er getest :	1. De uitgangssituatie wordt ingesteld. 2. De lees en schrijfapplicaties worden aangezet op de primaire locatie (A) 3. Een van de database nodes op de primaire locatie wordt uitgeschakeld. 4. De controller die niet bij deze database hoort wordt uitgeschakeld. 4. Er moet gecontroleerd worden of de schrijfapplicatie kan blijven schrijven en of de leesapplicatie rapporten op kan vragen.
Uitgangssituatie	Het systeem bevindt zich in snapshot 3 en is operationeel
Motivatie	Deze test is vergelijkbaar met Seq-8, alleen nu zijn het 2 componenten die niet in dezelfde groep zitten. In dit geval zal de applicatie zijn gegevens in de secundaire locatie opslaan, terwijl nog een van de actieve controllers op de primaire locatie worden gebruikt.
Verwachte resultaten	Applicatie kan doorwerken (op secundaire locatie)

Testcase SEQ-10	
Doel van de test:	Aantonen dat de applicatie door kan werken op locatie B, als de volledige locatie A (de primaire locatie) uitvalt
Wat wordt er getest :	1. De uitgangssituatie wordt ingesteld. 2. De lees en schrijfapplicaties worden aangezet op de primaire locatie (A) 3. De lees en schrijfapplicaties worden aangezet op de secundaire locatie (B) 4. Alle systemen op locatie A worden uitgezet door de power eraf te halen. 5. Controleren of de applicaties op locatie B blijven doorwerken. 6. Zo nee, dient het cluster herstart te worden en daarna gecontroleerd te worden of de data nog compleet is.
Uitgangssituatie	Het systeem bevindt zich in snapshot 3 en is operationeel
Motivatie	Deze test toont aan of het systeem bomb-proof is, oftewel als een volledige locatie uitvalt door een bom of ramp, dat dan de applicatie wel door kan werken op de secundaire locatie.
Verwachte resultaten	Applicatie kan doorwerken op de secundaire locatie, eventueel met benodigde switch

Testcase SEQ-11	
Doel van de test:	Aantonen dat de applicatie door kan werken op locatie A, als de volledige locatie B (de backup locatie) uitvalt
Wat wordt er getest :	1. De uitgangssituatie wordt ingesteld. 2. De lees en schrijfapplicaties worden aangezet op de primaire locatie (A) 3. De lees en schrijfapplicaties worden aangezet op de secundaire locatie (B) 4. Alle systemen op locatie B worden uitgezet door de power eraf te halen. 5. Controleren of de applicaties op locatie B blijven doorwerken.
Uitgangssituatie	Het systeem bevindt zich in snapshot 3 en is operationeel
Motivatie	Met deze testcase wordt aangetoond of het systeem door kan werken als de tweede locatie niet aanwezig is. Indien dat niet het geval is, zal het uitvallen van de tweede locatie toch downtime opleveren.
Verwachte resultaten	Applicatie kan doorwerken op de primaire locatie

D.2.2. Onderzoekscases

Onderzoekscase SEQ-12	
Doel van de test:	Onderzoeken hoeveel stappen nodig zijn om een gecrashte controller of database te kunnen herstellen en de benodigde tijd vast te stellen.
Wat wordt er getest :	De situatie van SEQ-5 wordt herhaald, waarna bekeken wordt welke stappen nodig zijn om een gecrasht component weer in de lucht te krijgen.
Uitgangssituatie	Het systeem bevindt zich in snapshot 2 en is operationeel
Motivatie	Deze case geeft een beeld van de beheersbaarheid en mogelijke problemen bij het herstellen van opgetreden problemen.
Verwachte resultaten	Het volgen van de herstelprocedure uit de handleiding levert herstel van het cluster op.

Onderzoekscase SEQ-13	
Doel van de test:	Onderzoeken of het verwijderen van een tabel of database gevolgen heeft voor de applicatie en wat deze gevolgen zijn. Is er verschil tussen het verwijderen via sequoia of direct op de database?
Wat wordt er getest :	1. De uitgangssituatie wordt ingesteld. 2. De lees en schrijfapplicaties worden aangezet op de primaire locatie (A) 3. Via een mysql client wordt zonder tussenkomst van een sequoia controller een database gedropd. 4. Wat gebeurt er? Blijven de applicaties functioneren, hoe reageert Sequoia hierop? 5. Stap 1 en 2 worden herhaald, waarna via de sequoia controller een database wordt gedropd. 6. Wat gebeurt er? Blijven de applicaties functioneren, hoe reageert Sequoia hierop?
Uitgangssituatie	Het systeem bevindt zich in snapshot 2 en is operationeel
Motivatie	Deze test toont aan hoe sequoia omgaat met problemen die door menselijk handelen worden veroorzaakt. Wat gebeurt er als een database beheerder direct op een datanode aan de slag gaat in plaats van via sequoia. Blijft de boel correct functioren, geeft sequoia aan dat er problemen zijn?
Verwachte resultaten	Systeem crasht als controller niet gebruikt wordt. Met controller worden alle databases verwijderd

Onderzoekscase SEQ-14	
Doel van de test:	Onderzoeken wat de invloed is van latency op de applicatie
Wat wordt er getest :	1. De uitgangssituatie wordt ingesteld. 2. De lees en schrijfapplicaties worden aangezet op de primaire locatie (A) 3. De latency van de WAN link wordt ingesteld op geen, 50ms, 100ms en 300ms waarna de invoegtest wordt uitgevoerd. Daarna wordt gekeken of meetbaar is wat de gevolgen daarvan zijn. 4. Zijn de gevolgen van de verschillende latency's zichtbaar? 5. Door middel van een gelijktijdig uitgevoerde mysqldump kan bekeken worden of beide databases (op locatie A als R) dezelfde data bevatten.
Uitgangssituatie	Het systeem bevindt zich in snapshot 3 en is operationeel
Motivatie	Zie IBM-07.
Verwachte resultaten	Resultaat van een uitgevoerde transactie duurt langer bij een latency van 300ms vergeleken met 50 ms of geen latency.

D.3. Product 3 – MySQL Cluster

Omdat de replicatie bij MySQL cluster asynchroon is, zal de WAN link simulatie slechts gebruikt worden om de latency te testen in de case die daarvoor opgezet is. Indien daarmee wordt aangetoond dat de asynchrone replicatie geen invloed uitoefent op de replicatie, zal de WAN link simulatie verder niet worden gebruikt voor de overige testen.

D.3.1. Testcases

Testcase MSC-01	
Doel van de test:	Aantonen dat de uitval van een storage node geen gevolgen heeft voor het cluster en daarmee de applicatie.
Wat wordt er getest :	1. De uitgangssituatie wordt ingesteld. 2. De lees en schrijfapplicaties worden op locatie A aangezet 3. Nadat het cluster operationeel is, wordt een van de storage nodes uitgeschakeld. 4. Blijven de applicaties functioneren?
Uitgangssituatie	Het systeem bevindt zich in snapshot 4 en is operationeel
Motivatie	Het uitvallen van een storage node mag volgens mysql geen gevolgen hebben voor het functioneren van het cluster. Deze test toont aan of de cluster oplossing daarin slaagt.
Verwachte resultaten	Het cluster herstelt zich van de uitval van de storage node en functioneert verder.

Testcase MSC-02	
Doel van de test:	Aantonen dat de uitval van een mysqld node geen gevolgen heeft voor het cluster en daarmee de applicatie.
Wat wordt er getest :	1. De uitgangssituatie wordt ingesteld. 2. De lees en schrijfapplicaties worden op locatie A aangezet 3. Nadat het cluster operationeel is, wordt een van de mysqld nodes uitgeschakeld. 4. Blijven de applicaties functioneren?
Uitgangssituatie	Het systeem bevindt zich in snapshot 4 en is operationeel
Motivatie	Het uitvallen van een mysqld node mag volgens mysql geen gevolgen hebben voor het functioneren van het cluster. Deze test toont aan of de cluster oplossing daarin slaagt.
Verwachte resultaten	Het cluster herstelt zich van de uitval van de mysqld node en functioneert verder.

Testcase MSC-03	
Doel van de test:	Aantonen dat het uitvoeren van zowel de mysqld als storage node op een machine geen gevolgen heeft voor het mysql cluster als deze machine uitvalt.
Wat wordt er getest :	1. De uitgangssituatie wordt ingesteld. 2. De lees en schrijfapplicaties worden op locatie A aangezet 3. Een van de machines waarop de storage node en mysqld node draait wordt uitgeschakeld. 4. Wat gebeurt er, functioneert het cluster nog?
Uitgangssituatie	Het systeem bevindt zich in snapshot 4 en is operationeel
Motivatie	Indien testen MSC-01 en MSC-02 goed waren, dan kan met deze test worden aangetoond of beide op een machine kunnen draaien. Indien deze machine dan uitvalt, vallen beide processen uit. Als het tegelijk uitvallen van beide geen gevolgen heeft, dan kan een machine voor beide processen worden gebruikt en kunnen de hardware kosten voor een minimale setup worden geminimaliseerd
Verwachte resultaten	Uitval van een machine heeft geen gevolgen.

Testcase MSC-04	
Doel van de test:	Aantonen dat het cluster stopt met functioneren als de management node ontbreekt bij uitval van een storage node.
Wat wordt er getest :	1. De uitgangssituatie wordt ingesteld. 2. De lees en schrijfapplicaties worden op locatie A aangezet 3. Nadat het cluster operationeel is, wordt de management node uitgeschakeld. Na een paar minuten wordt een van de storage nodes uitgeschakeld. 4. Blijven de applicaties functioneren?
Uitgangssituatie	Het systeem bevindt zich in snapshot 4 en is operationeel
Motivatie	De management node is de software die gebruikt wordt om het cluster op te starten en bij uitval van nodes het cluster herconfigeert naar de nieuwe situatie. Deze test toont aan of het cluster in staat is door te functioneren als deze management node ontbreekt (bijv voor onderhoud) op moment dat een component uitvalt.
Verwachte resultaten	Het cluster kan geen schrijf of leesacties meer uitvoeren

Testcase MSC-05	
Doel van de test:	Aantonen dat de secundaire locatie alleen voor leesacties gebruikt kan worden.
Wat wordt er getest :	1. De uitgangssituatie wordt ingesteld. 2. De lees en schrijfapplicaties worden op locatie A aangezet 3. De lees en schrijfapplicaties worden op locatie B aangezet 4. Gecontroleerd moet worden of beide kunnen functioneren
Uitgangssituatie	Het systeem bevindt zich in snapshot 5 en is operationeel
Motivatie	Zowel op het cluster in A als op het cluster in B zijn lees en schrijfacties mogelijk, omdat beide standalone kunnen opereren. Doordat replicatie plaatsvindt van A naar B treden wel problemen op als op B andere gegevens worden toegevoegd als op A, dan treden conflicten op. Omdat slechts replicatie van A naar B mogelijk is en niet andersom, is het schrijven op B van gegevens die op A ook opgeslagen zouden moeten worden niet verstandig, doordat deze verloren gaan indien B down gaat. Deze test toont aan of beide clusters onafhankelijk van elkaar functioneren of dat door de replicatie cluster b niet beschikbaar is voor acties.
Verwachte resultaten	Leesacties van het cluster in R zijn mogelijk, schrijfacties leveren problemen op doordat deze niet terug worden gerepliceerd naar locatie A. Beide kanten op

	repliceren is niet mogelijk in de standaard setup.

Testcase MSC-06	
Doel van de test:	Aantonen dat het uitvallen van de WAN link geen gevolgen heeft voor de applicatie.
Wat wordt er getest :	1. De uitgangssituatie wordt ingesteld. 2. De lees en schrijfapplicaties worden op locatie A aangezet. 3. De WAN link wordt uitgeschakeld 4. Blijft locatie A doordraaien? 5. De WAN link wordt hersteld (na 5 min). Wordt locatie B automatisch gesynchroniseerd met locatie A?
Uitgangssituatie	Het systeem bevindt zich in snapshot 5 en is operationeel.
Motivatie	Deze test toont aan of de twee clusters niet van de onderlinge connectie afhankelijk zijn. Als de WAN link uit kan vallen zonder gevolgen, biedt de oplossing een resistentie tegen connectieproblemen. Indien de uitval van de WAN link wel gevolgen heeft, dient deze link goed beschikbaar te worden gemaakt.
Verwachte resultaten	Locatie A blijft doordraaien, locatie B wordt automatisch gesynchroniseerd bij herstel van de WAN link.

D.3.2. Onderzoekscases

Onderzoekscase MSC-07	
Doel van de test:	Onderzoeken of het replicatieproces van de storage nodes of mysqld nodes afhankelijk is..
Wat wordt er getest :	1. De uitgangssituatie wordt ingesteld. 2. De lees en schrijfapplicaties worden op locatie A aangezet 3. Replicatie is operationeel naar locatie B. 4. Een van de storage nodes wordt uitgeschakeld. 5. Blijft de replicatie functioneren? 6. Zelfde nadat boel hersteld is voor een mysqld node.
Uitgangssituatie	Het systeem bevindt zich in snapshot 5 en is operationeel
Motivatie	Deze case onderzoekt van welke processen de replicatie afhankelijk is. Dit geeft inzicht in de manier waarop de replicatie plaatsvindt, of dat cluster afhankelijk is of dat de gewone mysql replicatie via de mysqld nodes plaatsvindt.
Verwachte resultaten	Replicatie is afhankelijk van de mysqld nodes en is niet cluster specifiek.

Onderzoekscase MSC-08	
Doel van de test:	Onderzoeken welke stappen nodig zijn om een cluster te repareren als een van de nodes kapot gaat en de tijd die daarvoor benodigd is.
Wat wordt er getest :	1. De uitgangssituatie wordt ingesteld. 2. De lees en schrijfapplicaties worden op locatie A aangezet 3. Een van de storage nodes wordt uitgeschakeld. 5. Welke stappen zijn nodig om de boel te repareren? 6. Zelfde nadat boel hersteld is voor een mysqld node.
Uitgangssituatie	Het systeem bevindt zich in snapshot 4 en is operationeel
Motivatie	Deze case geeft een indicatie van de tijd die nodig is om het cluster te repareren indien fouten optreden en wat voor gevolgen dat heeft op het beheertechnische vlak.
Verwachte	Stappenplan. Handleiding mysql toetsen?

resultaten	

Onderzoekscase MSC-09	
Doel van de test:	Onderzoeken welke stappen nodig zijn om het secundaire systeem te kunnen gebruiken als de primaire locatie faalt en de tijd die daarvoor benodigd is.
Wat wordt er getest :	1. De uitgangssituatie wordt ingesteld. 2. De lees en schrijfapplicaties worden op locatie A aangezet. 3. De primaire locatie A gaat uit. 4. Welke stappen zijn nodig om locatie B in de lucht te krijgen?
Uitgangssituatie	Het systeem bevindt zich in snapshot 5 en is operationeel
Motivatie	Zie MSC-08
Verwachte resultaten	Aanzetten van applicatieserver is voldoende?

Onderzoekscase MSC-10	
Doel van de test:	Onderzoeken wat de invloed is van latency op de applicatie
Wat wordt er getest :	1. De uitgangssituatie wordt ingesteld. 2. De lees en schrijfapplicaties worden aangezet op de primaire locatie (A) 3. De latency van de WAN link wordt ingesteld op geen, 50ms, 100ms en 300ms waarna de invoegtest wordt uitgevoerd. Daarna wordt gekeken of meetbaar is wat de gevolgen daarvan zijn. 4. Zijn de gevolgen van de verschillende latency's zichtbaar? 5. Door middel van een gelijktijdig uitgevoerde mysqldump kan bekeken worden of beide databases (op locatie A als B) dezelfde data bevatten.
Uitgangssituatie	Het systeem bevindt zich in snapshot 5 en is operationeel
Motivatie	Zie IBM-07
Verwachte resultaten	Locatie A en locatie B zijn niet consistent ten tijde van uitvoer van applicaties door de asynchroniteit van de replicatie.

Als de tijd het toelaat kunnen de volgende cases nog worden onderzocht

Onderzoekscase MSC-11	
Doel van de test:	Onderzoeken of de replicatietijd lineair toeneemt met de latency
Wat wordt er getest :	1. De uitgangssituatie wordt ingesteld. 2. De lees en schrijfapplicaties worden aangezet op de primaire locatie (A) 3. De latency van de WAN link wordt ingesteld op 1,2 en 5 sec waarna de invoegtest wordt uitgevoerd. 4. Zijn de rijen op locatie B precies na de delaytijd op locatie A aanwezig of loopt de replicatietijd harder op als de delay's groter worden? 5. Door middel van een gelijktijdig uitgevoerde mysqldump kan bekeken worden of beide databases (op locatie A als B) dezelfde data bevatten.
Uitgangssituatie	Het systeem bevindt zich in snapshot 5 en is operationeel
Motivatie	Met deze case kan worden gekeken of er behalve de latency van de WAN link nog een extra vertraging optreedt bij de replicatie of dat deze lineair verloopt met de latency. Dit kan inzicht bieden in de manier waarop de replicatie wordt aangepakt en of bijv batches worden toegepast.
Verwachte	Behalve de delay tijd wordt de replicatietijd niet vertraagd, hoe lang de delay ook is.

resultaten	

D.4. Client Applicatie

De client applicatie in deze testopstelling is een simpele browser, die een applicatie aanroept op de applicatie server. Het automatisch overschakelen naar de tweede locatie is met een browser niet mogelijk. Hoewel voor deze overschakeling in sommige gevallen het Domain Name System (DNS) wordt gebruikt is het hier niet goed voor geschikt. Het DNS systeem zet domeinnamen om in ip-adressen waarmee een browser het systeem kan bereiken. Door het domein naar een ander ip-adres te laten wijzen, kan de client naar een ander systeem worden overgeschakeld. Doordat het wijzigen van deze instellingen vaak langer dan een uur duurt en doordat de DNS gegevens vaak op de client gecached worden vanwege performance, duurt een automatische overschakeling van een client te lang en is er sprake van downtime voor de gebruiker. Een derde probleem is het feit dat DNS niet controleert of een systeem beschikbaar is. Een ip-adres van een niet beschikbaar systeem zal als het niet uit het antwoord wordt gehaald ervoor zorgen dat de client op een onbereikbaar systeem zal uitkomen.

Een oplossing voor dit probleem is het inbouwen van een check op de connectie in de client en deze door de client te laten afhandelen. Het IBM HADR systeem biedt een automatische client reroute optie, maar omdat gebruik wordt gemaakt van een tussenliggende applicatie server is dit niet mogelijk op de client. De applicatie server kan wel gebruik maken van deze functionaliteit voor de connectie met de database. De functionaliteit in de client om op de goede applicatieserver uit te komen dient dan zelf te worden geïmplementeerd.

In deze testopstelling wordt handmatig het ip-adres in de client gewijzigd, als de gehele primaire locatie uitvalt. Het implementeren van een automatisch systeem valt buiten de scope.

D.5. Applicatie Server

De applicatie server in de testopstelling bevat twee applicaties. De ene verzorgt het invoegen van nieuwe gegevens in de database, de andere verzorgt het ophalen van de data uit de database. Beide kunnen door middel van een webbrowser worden aangeroepen. Hierin wijkt de testopstelling af van het originele project, waarin voor de meetclient aparte software wordt geschreven. Daarbij wordt niet gebruik gemaakt van een webbrowser. Gezien de beperkte tijd is het zelf schrijven van deze meetclient geen optie in deze testopstelling.

Omdat de functionaliteit die getest gaat worden in deze testopstelling zich tussen de applicatie server en database server bevindt, is het niet van belang of voor de client een webbrowser of client applicatie wordt gebruikt. De functionaliteit bevindt zich op de applicatie server. Een voordeel van het gebruik

van een webapplicatie is de mogelijkheid om met een website benchmark tool eenvoudige performance testen uit te kunnen voeren op het geïnstalleerde product.

De applicatieserver is stateless, deze bevat geen informatie die benodigd is op andere applicatieservers als hij uitvalt. Doordat er geen state wordt bijgehouden, is elke applicatieserver hetzelfde en is geen replicatie van data benodigd tussen de applicatie servers. Op deze manier is het clusteren van de applicatieservers niet nodig. Voor het bereiken van de applicatie server kan gekozen worden tussen een oplossing met een virtueel ip-adres (en dan takeover) of het gebruik van een load balancer. De load balancer verzorgt dan het overzetten van de client requests naar de actieve applicatie server. In de testopstelling valt het clusteren van de applicatie servers buiten de scope. Het uitvallen van de applicatieserver zal dan ook niet getest worden in combinatie met de client. Wel zal gekeken worden wat de uitval voor gevolgen heeft voor de database connecties en of de applicatie server weer eenvoudig online gebracht kan worden.

D.6. Database Server

Voor de opslag van de meetdata wordt gebruik gemaakt van een database server. De database software op deze server is afhankelijk van het gekozen product, waarmee getest zal worden. Als operating system zal gebruik worden gemaakt van Linux, vooral omdat MySQL cluster niet op Windows beschikbaar is en Linux gratis beschikbaar is. Alle overige producten zijn op Linux verkrijgbaar, op Microsoft SQL Server na. Met SQL Server zal niet getest worden in deze opstelling, waardoor dit geen problemen oplevert.

Voor de database zal gebruik worden gemaakt van een eenvoudig database schema. Het volledig implementeren van het schema uit het project kost teveel tijd en brengt teveel complexiteit met zich mee. Door de database simpel te houden kan meer tijd worden besteedt aan het grondig testen van de producten en gaat geen tijd verloren aan het implementeren van details in de client applicatie en database schema's. Als het product ingezet zal worden voor het project, zal deze met de echte data worden getest.

D.7. Benodigdheden

Voor het opzetten van de testopstelling zijn verschillende producten benodigd. De benodigde hardware staat beschreven in D.7.1, de software producten in D.7.2 en de onderdelen van de te schrijven applicatie in D.7.3.

D.7.1. Hardware

Voor het testen zijn een aantal verschillende onderdelen nodig. Per onderdeel staan de benodigde spullen. Met het gebruik van virtualisatie software kan een deel van de hardware worden gesimuleerd. Vooral bij de Sequoia en MySQL opstelling scheelt dit behoorlijk in de benodigde hardware. Het gebruik van virtualisatie biedt ook voordelen met testen door het kunnen gebruiken van snapshots en het eenvoudig uit kunnen zetten van systemen of

netwerkverbindingen. Snapshots van het systeem op een bepaald moment voorkomen dat het instellen van uitgangssituaties veel handwerk met zich meebrengt en het makkelijk uitschakelen zorgt ervoor dat geen schade optreedt aan de gebruikte hardware.

- Wan link simulator
  - Standaard pc
  - Standaard hardware,functionerend onder linux
  - 2 netwerkkaarten met 100Mbit ondersteuning
  - 2 hubs met 100Mbit ondersteuning
- 1 client pc
  - Standaard pc met netwerkaansluiting
- Clusterhardware
  - Voor de verschillende producten zijn 2 fysieke machines nodig. Deze kunnen hergebruikt worden voor de verschillende producten. Op deze machines worden afhankelijk van het product een aantal machines gesimuleerd.
  - De pc's waarop de virtualisatiesoftware draait dienen voorzien te zijn van een snelle processor en zoveel mogelijk geheugen, zodat de gesimuleerde machines elkaar niet in de weg zitten. Bij zowel Sequoia als MySQL dienen 3 images tegelijk te kunnen draaien in de virtualisatiesoftware, waardoor een minimum van 4 (1 Host + 3 Guests) x 256 MB = 1GB wel aan te raden is.

Samenvattend :
1x Wanlink pc
1x Client pc
2x Cluster pc

Gebruikte PC configuratie's:
2x Dell optiPlex GX620DT – Pentium D 830 (3.00 GHz/800MHz/2x1MB,Int NIC)
2 GB geheugen
160GB Hardeschijf 7200rpm SATA 3.0Gb/s
Windows XP Professional
VMware Workstation 5.5

1x Dell 1.6 Ghz, als router en WAN emulator
1x Dell optiPlex GX280 Pentium 4 3.00 Ghz

D.7.2. Software

De volgende software is benodigd:

Producten :

- IBM DB2 UDB Enterprise Edition versie 8.2
  - Inclusief HADR.
  - 90 dagen trial versie
- MySQL 5.1
  - Versie 5.1.11 Beta
  - Gratis beschikbaar
- Sequoia v2.8.1 Middleware
  - Gratis beschikbaar

Operating Systems:

- Linux OS met ondersteuning voor DB2
  - Red Hat Enterprise Linux 4
- Netem pakket voor network simulatie
  - Gratis beschikbaar
- JBoss Applicatie Server
  - Gratis beschikbaar
- VMWare voor virtualisatie
  - VMWare Server Beta
  - Gratis beschikbaar

Ontwikkelomgeving

- Eclipse
  - Gratis beschikbaar
- Java SDK
  - Gratis beschikbaar
- Internet Explorer of Firefox
  - Gratis beschikbaar
- Database Open-source Test Suite (DOTS)
  - Versie: 15 mei 2006

D.7.3. Applicatie

De applicatie dient zelf geschreven te worden. De applicatie bevat de volgende onderdelen:

- Database schema's
- Meetapplicatie
- Rapportageapplicatie

APPENDIX E. TESTRESULTATEN

E.1. IBM

Alle IBM testen zijn uitgevoerd met de evaluatieversie van de IBM DB2 ESE database V8.2. Er zijn voor deze evaluatieversie meerdere updates verschenen sinds het testen, daardoor is het mogelijk dat testen met nieuwere versies andere resultaten opleveren.

Een probleem tijdens het testen was een bekend probleem met de fault manager die in een latere versie dan de evaluatie versie is opgelost, deze zorgde voor een hoge belasting van het systeem terwijl er niks aan de hand was. Dit had een negatieve invloed op de stresstest, waardoor de fault monitor is uitgezet tijdens het testen.
[http://www-1.ibm.com/support/docview.wss?uid=swg1LI70533]

Testcase IBM-01	
Doel van de test:	Aantonen dat beide locaties tegelijk gebruikt kunnen worden.
Verwacht Resultaat:	De database op locatie B is niet bruikbaar als deze zich als stand-by in HADR configuratie bevindt
Resultaat	Bij het uitvoeren van een applicatie op de database op locatie B als deze zich in stand-by bevindt, treedt een foutmelding op die aangeeft dat de stand-by database niet gebruikt kan worden. Het HADR systeem is dus een echt actief-passief systeem. De weergegeven foutmelding : “[IBM][CLI Driver] SQL1776N The command cannot be issued on an HADR Standby database. Reason code = ‘1’ “

Testcase IBM-02	
Doel van de test:	Aantonen dat de slave de rol van de master overneemt als deze uitvalt. De applicatie kan op de slave worden voortgezet
Verwachte resultaten	Het slave systeem neemt de database rol van de master over.
Resultaat	De slave database neemt de taken van de master wel over, maar dit gebeurt niet automatisch. Hiervoor dient een handmatige fail-over te worden uitgevoerd. Het DB2 systeem bevat geen monitoring die het mogelijk maakt om de database taken automatisch over te laten gaan naar het slave systeem. Op moment dat de master uitvalt kan wel handmatig de database van het master systeem naar het slave systeem worden gefailoverd. Om de database automatisch over te laten schakelen, moet een monitoring pakket worden gebruikt of dienen zelf scripts te worden geschreven. De database tools bieden wel ondersteuning voor scripting, zodat met een script de taak geautomatiseerd kan worden.

Testcase IBM-03	
Doel van de test:	Aantonen dat de uitval van de slave database geen gevolgen heeft voor de master en de applicatie.
Verwachte resultaten	De master ondervindt geen gevolgen van het uitvallen van de slave en de slave wordt na inschakeling automatisch gesynchroniseerd.
Resultaat	Het uitvallen van de slave levert geen problemen op voor de master. De applicatie kan zonder problemen doordraaien. Als de slave weer in actieve staat is gebracht dient het HADR systeem te worden herstart, zodat de slave weer gaat synchroniseren met de master. Dit gebeurt niet automatisch, zodat na een herstart van de slave ook naar het HADR systeem gekeken moet worden. Er is geen automatische synchronisatie mogelijk als de slave database is uitgevallen, dit dient handmatig te worden gestart. Het herstarten van de slave database heeft dus tot gevolg dat ook de HADR cyclus moet worden herstart. Dit herstarten heeft verder geen negatieve gevolgen voor de master of de applicatie. Uiteraard moet bij een volledige crash van de slave eerst een backup worden teruggezet en HADR opnieuw worden geconfigureerd.

Testcase IBM-04	
Doel van de test:	Aantonen dat de uitval van de WAN link geen gevolgen heeft voor de applicatie of de master, onafhankelijk welke synchronisatie mode gebruikt wordt.
Verwachte resultaten	Het master systeem blijft functioneren, en de taken van de clients kunnen gewoon worden afgehandeld.
Resultaat	De Master ondervindt geen gevolgen van het uitvallen van de connectie en kan de transacties blijven verwerken, onafhankelijk van de synchronisatiemodus waarin het systeem zich bevindt. Zolang de master en slave beide actief blijven, zal automatisch bij het herstellen van de WAN link worden gehersynchroniseerd. Wel is er een achterstand mogelijk van de slave op de master. Een failover of takeover nadat de WAN link niet beschikbaar was, kan dan dataverlies met zich meebrengen, omdat nog niet volledig synchroon wordt gelopen.

Testcase IBM-05	
Doel van de test:	Aantonen dat de data automatisch wordt gesynchroniseerd als de WAN link na een failure weer wordt hersteld.
Verwachte resultaten	Het master systeem blijft functioneren, en de taken van de clients kunnen gewoon worden afgehandeld. Het slave systeem wordt automatisch gesynchroniseerd bij herstel.
Resultaat	Zoals al aangegeven in IBM-04 wordt de data automatisch gesynchroniseerd nadat de WAN link weer is hersteld. De tijd die nodig is voor volledige synchronisatie is afhankelijk van de belasting. Hoe drukker de master database is, hoe langer de synchronisatie duurt en hoe meer risico er is op dataverlies door uitval van de master voordat synchronisatie is afgerond. De tijdelijke uitval van de WAN link heeft behalve in toename van het risico van dataverlies geen gevolgen.

Testcase IBM-06	
Doel van de test:	Aantonen dat een client automatisch wordt overgeschakeld naar de standby indien de standby een takeover uitvoert op de master
Verwachte resultaten	Het master systeem blijft functioneren, en de taken van de clients kunnen gewoon worden afgehandeld. Het slave systeem wordt automatisch gesynchroniseerd bij herstel.
Resultaat	Bij het gebruik van de DB2 Universal JDBC Driver is er alleen ondersteuning voor automatic client reroute als gebruik wordt gemaakt van een DataSource. Voor een uitgebreide uitleg (http://publib.boulder.ibm.com/infocenter/db2luw/v8/index.jsp?topic=/com.ibm.db2.udb.doc/ad/cjvclrrt.htm). Automatic Client Reroute is de oplossing van IBM waarmee de client automatisch naar het standby systeem wordt geleid als deze de taken van de primary heeft overgenomen. Bij het programmeren van de applicatie dient dus rekening te worden gehouden met de manier waarop de connectie gemaakt wordt met de database. De gebruikte DOTS test suite maakt op een andere manier een connectie met de database, waardoor deze een error toont indien de primary en backup van role wisselen of een failover plaats moet vinden. Bij het gebruik van een dataSource is er wel ondersteuning voor het ACR en wordt dit automatisch gebruikt indien het is aangezet op de server. Bij het configureren van de HADR setup dient een van de pagina's om de client reroute functionaliteit aan te zetten. Hoewel IBM het doet voorkomen dat ACR in alle gevallen automatisch werkt, geldt dit voor de JDBC driver dus alleen indien van de javax.sql.DataSource interface gebruik wordt gemaakt. Het implementeren en testen van een applicatie die een DataSource gebruikt is niet mogelijk gezien de tijd, waardoor een volledig antwoord op deze testcase niet mogelijk is.

Onderzoekscase IBM-07	
Doel van de test:	Onderzoeken wat de invloed is van de HA features en latency op de performance van het DB2 systeem
Verwachte resultaten	Resultaat van een uitgevoerde transactie duurt langer bij een latency van 300ms vergeleken met 50 ms of geen latency. Transactie response duurt bij een synchrone mode langer dan bij asynchrone mode.
Resultaat	Zie document testresultaat latency en synchronisatie

E.2. Sequoia

Alle testen zijn uitgevoerd met de Sequoia 2.8.2 versie van 7 juni 2006. De gebruikte database is MySQL 5.1.11 beta, zodat de resultaten vergeleken kunnen worden met de resultaten uit de testen van MySQL. Hoewel deze versie nog in beta is, zijn de gebruikte functies uit het standalone deel van MySQL die gebruikt worden door Sequoia stabiel, zodat dit geen noemenswaardige invloed heeft op de resultaten.

Testcase SEQ-1	
Doel van de test:	Aantonen dat de uitval van één sequoia controller op de primaire locatie geen invloed heeft op de applicatie.
Verwachte resultaten	De schrijfapplicatie ondervindt geen gevolgen van de controller uitval en kan zijn taak vervolgen. De leesapplicatie ondervindt geen gevolgen en kan rapporten opvragen.
Resultaat	Als bij het configureren van de client ervoor wordt gezorgd dat beide controllers in de database connectiestring voorkomen, dan zorgt de Sequoia JDBC driver ervoor dat automatisch overgeschakeld wordt naar een alternatieve controller. Als slechts een controller geconfigureerd is en deze valt uit, dan zal de applicatie niet verder kunnen functioneren. Het automatisch overschakelen werkt zonder problemen. Transacties die onderweg zijn ten tijde van een uitval worden op de overgebleven controller uitgevoerd. Als dat niet lukt worden ze ongedaan gemaakt en kunnen door de applicatie opnieuw uitgevoerd worden. Uit de logs van de controller : 2006-06-27 18:38:52,106 WARN controller.virtualdatabase.myDB Controller Member(address=bpsseq1.technolution.nl/172.16.200.46:32776, uid=myDB) has left the cluster. 2006-06-27 18:38:52,107 INFO controller.virtualdatabase.myDB 3 requests were waiting responses from Member(address=bpsseq1.technolution.nl/172.16.200.46:32776, uid=myDB) 2006-06-27 18:38:52,412 WARN controller.RequestManager.myDB 1 controller(s) died during execution of request 3090 2006-06-27 18:38:52,413 WARN controller.RequestManager.myDB 1 controller(s) died during execution of request 3096 2006-06-27 18:38:52,413 WARN controller.RequestManager.myDB 1 controller(s) died during execution of request 3094 2006-06-27 18:38:52,468 INFO controller.requestmanager.cleanup Waiting 120000ms for client of controller 562949953421312 to failover Op de applicatie is niks te zien van een falende controller, voor deze wordt de fout netjes gemaskeerd door de JDBC driver en de alternatieve controller.

Testcase SEQ-2	
Doel van de test:	Aantonen dat het systeem op beide locaties tegelijkertijd gebruikt kan worden
Verwachte resultaten	De operaties worden succesvol verwerkt.
Resultaat	Doordat alle controllers tegelijkertijd actief zijn, is het gebruik van de secundaire locatie mogelijk. Daardoor kunnen zowel op locatie A als op locatie B applicaties worden gedraaid. Het overschakelen van de primaire naar de secundaire locatie behoeft dus ook geen actie op Sequoia of database niveau. Indien de applicaties met alle controllers worden ingesteld, zoals bij SEQ-1 aangegeven, dan is voor de overschakeling van deze applicaties ook geen actie benodigd. Alle transacties worden succesvol verwerkt.

Testcase SEQ-3	
Doel van de test:	Aantonen dat de uitval van de WAN-link geen gevolgen heeft voor de applicatie op de primaire locatie.
Verwachte resultaten	De 2e locatie wordt uitgeschakeld, 1e locatie draait door.
Resultaat	In tegenstelling tot de verwachte resultaten, blijven beide locaties doordraaien. Beide locaties zijn van mening dat de andere locatie gecrasht is. Er treedt een zogenaamd split-brain scenario op. Op beide locaties blijven de controllers functioneren, waarbij ze aannemen dat de andere controller niet meer draait. Dit betekent dat als beide locaties tegelijkertijd door andere applicaties gebruikt worden, deze niet meer consistent zijn met elkaar. Na herstel van de WAN link treedt, zoals bij SEQ-4 besproken, dan ook geen automatische synchronisatie meer op. Dit betekent dat de status van de WAN link goed in de gaten moet worden gehouden als beide locaties gebruikt moeten kunnen worden. Indien slechts een locatie gebruikt hoeft te worden, is het ontbreken van de WAN link minder problematisch. In dat geval kan de secundaire locatie na herstel van de link weer gemakkelijk gesynchroniseerd worden.

Testcase SEQ-4	
Doel van de test:	Aantonen dat bij herstel van de WAN link de cluster configuratie automatisch hersteld wordt
Verwachte resultaten	Systeem herstelt de controller configuratie en synchroniseert de gegevens automatisch
Resultaat	Zoals al bij SEQ-3 is aangetoond, ontstaat een split-brain scenario bij uitval van de WAN-link. Dit betekent dat beide locaties niet automatisch met elkaar gesynchroniseerd kunnen worden er wordt door sequoia geen automatische oplossing voor dit probleem geboden. Het wordt aan de beheerder overgelaten om in dat geval beide locaties weer in een consistente staat te brengen. Dit wordt ook aangegeven in de documentatie: http://sequoia.continuent.org/doc/latest/userGuide/ar01s07.html#current_controller_limitations * network partition/reconciliation is not supported

Testcase SEQ-5	
Doel van de test:	Aantonen dat de uitval van één databasenode geen gevolgen heeft voor de applicatie.
Verwachte resultaten	De applicatie ondervindt geen gevolgen van de controller uitval en kan zijn taak vervolgen.
Resultaat	Het uitvallen van een backend node levert geen enkel gevolg op voor de applicatie. Indien een controller maar één backend heeft die uitvalt, zal deze alle requests doorsturen naar een van de andere controllers. Indien er meerdere backends zijn, zal een van de actieve backends gebruikt worden voor requests. De uitgevallen backend wordt uitgeschakeld in de controller, zonder dat de applicatie hier iets van opmerkt. De uitval van een databasenode wordt door het sequoia cluster dus gemaskeerd voor de applicatie.

Testcase SEQ-6	
Doel van de test:	Aantonen dat het uit en aanzetten van de applicatie server geen gevolgen heeft voor de cluster configuratie
Verwachte resultaten	De applicatie kan na het herstarten van de applicatieserver zonder problemen verder werken. Er hoeven geen cluster aanpassingen te worden gedaan.
Resultaat	De sequoia controllers en database backends zijn niet afhankelijk van de applicatieserver of applicatie. Het uitzetten van de applicatieserver heeft dan ook geen gevolgen, tenzij ook de controller of backend op deze machine draaien. In dat geval zal het cluster maatregelen moeten nemen om de uitval op te vangen. Bij alleen de uitval van de applicatieserver zijn er geen gevolgen voor het cluster, wel kan op dat moment de applicatie niet blijven functioneren (tenzij een HA oplossing voor het applicatieserver wordt toegepast)

Testcase SEQ-7	
Doel van de test:	Aantonen dat beide locaties nog bruikbaar zijn als de WAN-link niet meer beschikbaar is
Verwachte resultaten	Een van beide locaties wordt uitgeschakeld.
Resultaat	Zoals al aangetoond bij SEQ-3, blijven beide locaties bruikbaar indien de WAN link niet beschikbaar is. In tegenstelling tot de verwachte resultaten, blijven beide dus operationeel. Het uitschakelen van een deel van het cluster indien de WAN link uitvalt is mogelijk door de code aan te passen. Zowel het gebruik van een zogenaamde master als een probeer-ping zijn gediscussieerd op de sequoia mailinglijst. Beide zijn nog niet geïmplementeerd.

Testcase SEQ-8	
Doel van de test:	Aantonen dat het tegelijkertijd uitvallen van een database en controller node in dezelfde groep, waarbij de controller beheer voert over de database node, geen gevolgen heeft voor de applicatie.
Verwachte resultaten	Applicatie kan doorwerken
Resultaat	Deze situatie is vergelijkbaar met het uitvallen van alleen de controller, omdat de database backend zoiezo niet meer beschikbaar is als de controller uitvalt. De database nodes worden niet naar een andere controller geschoven indien de controller uitvalt. Dat betekent dat als met slechts een controller gewerkt zou worden, deze controller een single point of failure in het systeem zou worden. Uitval van deze controller betekent dan uitval van het gehele database cluster. Uitval van zowel de database node als controller heeft geen extra gevolgen ten opzicht van alleen uitval van de controller (zolang deze in dezelfde groep zitten). Als beide in een andere groep zitten, treedt SEQ-9 op. Bij het uitvallen van een controller als er nog andere controllers over zijn, heeft geen gevolgen voor de applicatie mits deze een connectiestring met meerdere controllers bevat zoals aangegeven bij SEQ-1.

Testcase SEQ-9	
Doel van de test:	Aantonen dat het tegelijkertijd uitvallen van een database en controller node in verschillende groepen geen gevolgen heeft voor de applicatie
Verwachte resultaten	Applicatie kan doorwerken (op secundaire locatie)
Resultaat	Als zowel een controller als een database node uitvallen die niet in dezelfde groep zitten, dan blijft alleen de groep die op de secundaire locatie draait goed functioneren. De eerste controller valt uit, dus is niet meer bereikbaar. De tweede controller op de primaire locatie draait wel verder, maar heeft geen actieve backend meer. Daardoor zal deze alle requests moeten forwarden naar de controller op de secundaire locatie. De applicatie ondervindt hier geen hinder van, wel is er een invloed op performance te verwachten. Indien meerdere backends achter een controller hangen (alternatieve opstelling), dan zal de tweede controller zelf nog requests kunnen afhandelen en wordt de secundaire locatie niet volledig gebruikt.

Testcase SEQ-10	
Doel van de test:	Aantonen dat de applicatie door kan werken op locatie B, als de volledige locatie A (de primaire locatie) uitvalt
Verwachte resultaten	Applicatie kan doorwerken op de secundaire locatie, eventueel met benodigde switch
Resultaat	Het doorwerken op de secundaire locatie B is geen enkel probleem voor de applicatie en deze zal daar ook geen gevolgen van onder vinden. Wel is van belang hoe de gebruiker bij de applicatie komt. Zolang de applicatie niet getroffen wordt door de uitval van de volledige A locatie, zal de gebruiker geen actie hoeven te ondernemen. Indien dit wel het geval is, zal de gebruiker zelf de applicatie op locatie B moeten benaderen.

Testcase SEQ-11	
Doel van de test:	Aantonen dat de applicatie door kan werken op locatie A, als de volledige locatie B (de backup locatie) uitvalt
Verwachte resultaten	Applicatie kan doorwerken op de primaire locatie
Resultaat	De uitval van de volledige B locatie heeft voor het cluster slechts als gevolg dat een van de controllers uitvalt. Voor de applicatie en de gebruiker is dit niet zichtbaar, deze blijven zonder problemen correct functioneren.

Onderzoekscase SEQ-12	
Doel van de test:	Onderzoeken hoeveel stappen nodig zijn om een gecrashte controller of database te kunnen herstellen en de benodigde tijd vast te stellen.
Verwachte resultaten	Het volgen van de herstelprocedure uit de handleiding levert herstel van het cluster op.
Resultaat	Het herstellen van een database node in het cluster is vrij eenvoudig. Kortweg komt het erop neer dat een backup van het draaiende cluster gemaakt moet worden, waarna deze op de gecrashte database wordt hersteld. Hierna kan de database node weer opnieuw worden gestart en worden de wijzigingen sinds de backup aan de node doorgegeven. Na deze hersynchronisatie wordt de node op enable gezet en is het cluster weer operationeel. Het herstellen van een controller is iets ingewikkelder, omdat voor deze ook de recovery log hersteld moet worden. Deze recovery log zorgt ervoor dat uitgevallen nodes hersteld kunnen worden tot de meest actuele situatie door het opnieuw uitvoeren van statements. Het volgen van de handleiding leidt tot het gewenste effect. [http://sequoia.continuent.org/doc/latest/sequoia-admin-guide.pdf] De benodigde hersteltijd is afhankelijk van de drukte op het systeem en de grootte van de backup die gemaakt en hersteld moet worden. Hoe drukker het systeem is, hoe meer transacties er na de restore actie uitgevoerd moeten worden en hoe langer het dus duurt. Hoewel het systeem wel blijft functioneren, is een degradatie van de performance en mogelijke uitval bij extra optredende problemen wel belangrijk genoeg om het systeem zo snel mogelijk weer in goede staat te krijgen. Sequoia biedt verschillende backupers om databases van een backup te voorzien. De meegeleverde MySQLBackuper werkt echter nog niet samen met de 5.1.11 beta van MySQL. Op de versies 4.1 en 5.0 werkt dit echter probleemloos, hoogstwaarschijnlijk wordt dit probleem snel opgelost als de 5.1 versie van MySQL in productie gaat. Een issue bij het backupen is dat een van de nodes gedisabled moet worden zodat daarvan de backup gemaakt kan worden. Dit is een probleem indien slechts twee controllers beschikbaar zijn die allebei maar één node hebben. Indien een van beide crasht, moet de andere node uitgezet worden om de backup te maken. Op dat moment is het systeem dus niet meer beschikbaar. Er zijn dus minimaal 3 controllers met 1 backend nodig of twee backends per controller als 2 controllers gebruikt worden, om onbeschikbaarheid bij het maken van een backup te voorkomen. Het maken van backups kan geautomatiseerd worden in Sequoia, waardoor extra veiligheid ingebouwd kan worden tegen menselijke fouten.

Onderzoekscase SEQ-13	
Doel van de test:	Onderzoeken of het verwijderen van een tabel of database gevolgen heeft voor de applicatie en wat deze gevolgen zijn. Is er verschil tussen het verwijderen via sequoia of direct op de database?
Verwachte resultaten	Systeem crasht als controller niet gebruikt wordt. Met controller worden alle databases verwijderd
Resultaat	Hoewel Sequoia in sommige gevallen de wijzigingen kan detecteren, is het resultaat van een verwijdering van een tabel in de meeste gevallen een error in de applicatie en kan deze niet verder werken. In een aantal gevallen schakelde sequoia de backend waarop de tabel verwijderd was uit, en kon op de overige nodes worden doorgegaan. Dit was echter van korte duur, binnen 10 tot 30 seconden volgende alsnog de errors richting de client en werkte de applicatie niet correct meer. Dus hoewel Sequoia sommige menselijke fouten wel kan detecteren, levert dit bijna altijd een probleem op dat niet door Sequoia zelf opgelost kan worden. Indien de statements via de controllers van Sequoia worden gegeven, worden deze netjes op alle backends uitgevoerd en kan de applicatie normaal blijven functioneren. Het is dus belangrijk beheer op het database systeem op de correcte manier uit te voeren.

Onderzoekscase SEQ-14	
Doel van de test:	Onderzoeken wat de invloed is van latency op de applicatie
Verwachte resultaten	Resultaat van een uitgevoerde transactie duurt langer bij een latency van 300ms vergeleken met 50 ms of geen latency.
Resultaat	Zie stresstest resultaat

E.3. MySQL

Alle MySQL testen zijn uitgevoerd met de Beta 5.1.11 release van MySQL van 26 mei 2006 op een Red Hat Enterprise Linux 4 besturingssysteem. Omdat het hier om een beta release gaat, is het mogelijk dat de uiteindelijke release andere functionaliteit bevat en/of performance optimalisaties.

Testcase MSC-01	
Doel van de test:	Aantonen dat de uitval van een storage node geen gevolgen heeft voor het cluster en daarmee de applicatie.
Verwachte resultaten	Het cluster herstelt zich van de uitval van de storage node en functioneert verder.
Resultaat	Nadat het cluster succesvol is gestart, functioneert het zonder problemen verder als een van de storage nodes uitvalt of wordt gestopt. De applicatie kan zonder problemen transacties blijven uitvoeren en ziet geen gevolgen van de uitval. Indien alleen het ndbd proces is gestopt of gecrashed kan door het simpelweg opnieuw uitvoeren van het ndbd commando de cluster functionaliteit van de utigevallen node worden hersteld. Indien de volledige harde schijf is gecrashed zal een nieuw image eropgezet moeten worden met dezelfde instellingen, waarna door middel van een backup van de cluster database de functionaliteit moet worden hersteld. Meer hierover in MSC-08. In de cluster logs wordt de volgende melding getoond als een storage node uitvalt : “Node 2 disconnected”, “Communication to node 2 closed”. Bij een shutdown staat er “Node 2: Node shutdown completed. Initiated by signal 15”. In beide gevallen wordt een netwerk partitie reconciliatie protocol gestart die ervoor zorgt dat de er geen split-brain scenario optreedt. Bij het gebruik van twee nodes, zal dit ervoor zorgen dat de overgebleven node de nieuwe master node wordt, bij meerdere nodes wordt gekeken of er voldoende nodes over zijn om het cluster voort te zetten.

Testcase MSC-02	
Doel van de test:	Aantonen dat de uitval van een mysqld node geen gevolgen heeft voor het cluster en daarmee de applicatie.
Verwachte resultaten	Het cluster herstelt zich van de uitval van de mysqld node en functioneert verder.
Resultaat	Het cluster is niet afhankelijk van de mysqld nodes en het uitzetten of uitvallen van een van deze nodes is daarom geen probleem voor het cluster zelf. Na het uitzetten van een van de mysqld nodes functioneert het cluster zonder problemen verder. De applicatie of client die deze mysqld node gebruikt, wordt echter niet automatisch overgeschakeld naar een andere node indien de gebruikte node uitvalt. Dat betekent dat deze dan wel de gevolgen van uitval ondervindt, doordat deze niet meer bij de data kan en dus niet verder kan functioneren. Alleen MySQL Cluster gebruiken is dus niet voldoende, het gebruik van een load balancer of een oplossing in de applicatie is noodzakelijk om de gevolgen te vermijden.

Testcase MSC-03	
Doel van de test:	Aantonen dat het uitvoeren van zowel de mysqld als storage node op een machine geen gevolgen heeft voor het mysql cluster als deze machine uitvalt.
Verwachte resultaten	Uitval van een machine heeft geen gevolgen.
Resultaat	Als zowel de mysqld node als de data node op een machine draaien bestaat de kans dat beide uitvallen, doordat de machine uitvalt. Dit levert dan een combinatie op van de resultaten bij MSC-01 en MSC-02. Het cluster kan blijven functioneren, omdat op een andere machine de data node door blijft draaien. De applicaties die van de uitgevallen machine gebruik maken zullen niet verder kunnen werken, zoals al bij MSC-02 aangegeven. Ook hier is dus eenzelfde oplossing nodig. Een reden om beide processen te scheiden is het niet tegelijkertijd uitvallen indien de machine uitvalt, waardoor het cluster slechts een van beide processen verliest en niet allebei tegelijk. Het is echter goed mogelijk om beide op dezelfde machine te draaien, zolang in de cluster opstelling rekening is gehouden (qua performance en redundantie) met het uitvallen van beide tegelijk.

Testcase MSC-04	
Doel van de test:	Aantonen dat het cluster stopt met functioneren als de management node ontbreekt bij uitval van een storage node.
Verwachte resultaten	Het cluster kan geen schrijf of leesacties meer uitvoeren
Resultaat	Het cluster kan zonder problemen doorfunctioneren als de management node niet aanwezig is. Echter op moment dat dan een fout optreedt waarbij de management node benodigd is, zoals bij uitval van een storage node, stopt het cluster met functioneren. De MYSQLD nodes kunnen op dat moment de storage nodes niet meer vinden, omdat zij door de management node op de hoogte moeten worden gehouden van de locatie en status van de storage nodes. Het doordraaien van het cluster is dan ook uitgesloten als de management node uit is op moment dat een fout optreedt binnen het cluster. Het uitzetten van de management node is dus niet verstandig en dient alleen te gebeuren indien onderhoud aan het systeem waarop de management node draait moet worden gedaan. In dat geval is het verstandig de onderhoudstijd zo kort mogelijk te maken, omdat het volledige cluster uitvalt als dan een probleem optreedt.

Testcase MSC-05	
Doel van de test:	Aantonen dat de secundaire locatie alleen voor leesacties gebruikt kan worden.
Verwachte resultaten	Leesacties van het cluster in B zijn mogelijk, schrijfacties leveren problemen op doordat deze niet terug worden gerepliceerd naar locatie A. Beide kanten op repliceren is niet mogelijk in de standaard replicatie opstelling.
Resultaat	Alle data wordt door de replicatieopzet van het cluster in A naar het cluster in B overgezet. Deze data is op het cluster in B toegankelijk voor applicaties en clients om leesacties op uit te voeren. In de applicatie dient wel rekening te worden gehouden met de mogelijke vertraging door de asynchrone replicatie. Data die op A is ingevoerd, hoeft niet direct bij B terug te komen als deze data gelezen wordt door de applicatie. Ook schrijfacties zijn mogelijk, alleen gaan deze acties verloren indien het cluster in B crasht. Er kunnen ook update conflicten en primary key conflicten optreden, omdat de data in B al is ingevoegd voordat de replicatie ze vanuit A in B in wil voegen. Dat kan een stop betekenen van het replicatie proces, waardoor het niet verstandig is het cluster in B ook voor schrijfacties te gebruiken. Wel is een snelle failover van cluster A naar cluster B mogelijk, omdat alleen het replicatieproces gestopt hoeft te worden. Doordat de replicatie asynchroon verloopt, kan dan wel dataverlies optreden.

Testcase MSC-06	
Doel van de test:	Aantonen dat het uitvallen van de WAN link geen gevolgen heeft voor de applicatie.
Verwachte resultaten	Locatie A blijft doordraaien, locatie B wordt automatisch gesynchroniseerd bij herstel van de WAN link.
Resultaat	Het uitvallen van de netwerkverbinding tussen locatie A en B heeft geen gevolgen voor de transacties van de applicatie. Nadat de link weer is hersteld wordt automatisch de wijzigingen gehersynchroniseerd. Tijdens het afwezig zijn van de verbinding worden de wijzigingen opgeslagen op A en neemt het verschil tussen A en B toe. Afhankelijk van de belasting neemt dit verschil weer af als de link wordt hersteld. Indien de link te lang niet beschikbaar was, kan het noodzakelijk worden om de achterstand in te lopen via een backup in plaats van synchronisatie. Dit is noodzakelijk als de belasting van de systemen te hoog is om via synchronisatie het verschil in te lopen.

Onderzoekscase MSC-07	
Doel van de test:	Onderzoeken of het replicatieproces van de storage nodes of mysqld nodes afhankelijk is.
Verwachte resultaten	Replicatie is afhankelijk van de mysqld nodes en is niet cluster specifiek.
Resultaat	De replicatie oplossing is niet specifiek ontworpen voor MySQL cluster maar is hetzelfde als de normale replicatie indien een gewone MySQL server gebruikt wordt. Volgens de documentatie [http://dev.mysql.com/doc/refman/5.1/en/mysql-cluster-replication.html] is de replicatie afhankelijk van de mysqld nodes en niet van de data nodes. Uiteraard dient het cluster wel te functioneren om de data naar een ander cluster of standalone slave te kunnen repliceren. Hierdoor kan gekozen worden tussen een cluster opstelling als backup of alleen een enkele server. De eerste biedt dan een goede opstelling om de taken van de primaire uitgevallen cluster over te nemen, het tweede biedt vooral data backup aan en zal de belasting mogelijk niet aankunnen indien het cluster ook om performance en schaalredenen gebruikt wordt.

Onderzoekscase MSC-08	
Doel van de test:	Onderzoeken welke stappen nodig zijn om een cluster te repareren als een van de nodes kapot gaat en de tijd die daarvoor benodigd is.
Verwachte resultaten	Stappenplan. Handleiding mysql toetsen?
Resultaat	Indien alleen het cluster is uitgevallen of gecrashd, maar geen hardware problemen zijn opgetreden is het in veel gevallen voldoende het cluster opnieuw te starten door ndbd te gebruiken. Indien er een groot verschil zit tussen inhoud van het cluster en de kapotte node is het verstandig een backup en restore operatie uit te voeren, zoals aangegeven in hoofdstuk 17.7.5 Online backup of MySQL Cluster. Als een hardware probleem optreedt, dient deze eerst te worden opgelost. Omdat een mysql cluster door kan draaien als een van de nodes ontbreekt, kan dit zorgvuldig worden behandeld. Uiteraard is het cluster op dat moment erg kwetsbaar, doordat uitval van een tweede node kan betekenen dat het cluster wel in zijn geheel uitvalt (afhankelijk van het aantal nodes en node groups) . In sommige gevallen is het enkel opstarten van ndbd niet voldoende en wordt dezelfde node weer uitgezet. Het heropstarten wil dan nog wel eens het probleem oplossen. Omdat het cluster proces vrij streng is op netwerk storingen, wil dat nog wel is tegenwerken als aan de opstart van het cluster wordt begonnen.

Onderzoekscase MSC-09	
Doel van de test:	Onderzoeken welke stappen nodig zijn om het secundaire systeem te kunnen gebruiken als de primaire locatie faalt en de tijd die daarvoor benodigd is.
Verwachte resultaten	Aanzetten van applicatieserver is voldoende?
Resultaat	Indien de volledige primaire locatie uitvalt, is het secundaire systeem snel in de lucht te krijgen. Het backup cluster draait gewoon als standby mee tijdens het repliceren vanaf de primaire cluster en dus is het alleen een kwestie van het omzetten van de applicaties die het cluster moeten gebruiken. In de applicaties moeten de ip adressen van de mysqld nodes van de secundaire locatie worden ingevoerd, waarna het systeem weer compleet werkt. Het is verstandig om de replicatie processen uit te zetten, hoewel dit niet perse noodzakelijk is. Nadat de primaire locatie weer in de lucht is, kan door replicatie deze weer van data worden voorzien. Wel dient ervoor gezorgd te worden dat geen data verloren gaat door de asynchrone replicatie. Het terugswitchen van de secundaire naar primaire locatie dient dus zorgvuldig plaats te vinden.

Onderzoekscase MSC-10	
Doel van de test:	Onderzoeken wat de invloed is van latency op de applicatie
Verwachte resultaten	Locatie A en locatie B zijn niet consistent ten tijde van uitvoer van applicaties door de asynchroniteit van de replicatie.
Resultaat	Het cluster in locatie B loopt een aantal secondes achter tov de locatie A ten tijde van het draaien van de DOTS test suite. Indien op beide systemen grote belasting is, loopt deze tijd op. Als het weer rustiger wordt voor de systemen wordt deze achterstand zeer snel weer ingehaald. Vooral de belasting op het slave systeem bepaalt hoe ver de replicatie achterloopt, omdat het verwerken van de transacties van de cpu van de slave afhankelijk is. Het ophalen van de transacties van de master heeft relatief weinig last van de belasting van de systemen, die is meer afhankelijk van de belasting van het netwerk.