Spill Productions BV

Eindverslag

Pakketselectie en implementatie van een prototype bij Spill Productions

Auteur:	Erik van den Berg (20045027)
Opdrachtgever:	Ivo Teel
Docent:	R. Bechet-Tjoonk, R.A. Mantel
Blok:	afstuderen
Datum:	14-05-2007
Plaats:	Hilversum

Referaat

Aan:

Dhr R. A. Mantel	(examinator)
Mevr. H.G.J. Bechect-Tjoonk	(examinator)
Dhr. I. Teel	(bedrijfsbegeleider)

Afstudeerbedrijf:

Spill productions B.V.
Arendstraat 23
1223 RE
Telefoon: +31 35 646 63 00
Fax: +31 35 646 63 01

Auteur:

Erik van den Berg
20045027

Studie:

Instelling: Haagse Hogeschool
Afdeling: Informatica
Periode: 2007-1.2

Datum:

13 september 2007

Descriptoren:

Pakketselectie
Indora
KPMG
MySQL
PostgreSQL
Cluster
Replicatie
Scalability
High Availability
Shortlist
Longlist
DBMS

Voorwoord

In het kader van mijn studie Informatica aan de Haagse Hogeschool is deze opdracht uitgevoerd voor het bedrijf Spill Productions B.V. De opdracht behelsde pakketselectie en implementatie van een prototype voor een DBMS.

Tevens wil ik mijn bedrijfsbegeleider dhr Teel bedanken voor zijn begeleiding tijdens het project en het beschikbaar stellen van de opdracht. Daarnaast wil ik ook alle collega's bedanken die mijn vragen hebben helpen beantwoorden en het geven van feedback op documentatie.

Dit procesverslag is bedoeld voor personen en/of organisaties die inzicht willen hebben in het proces van pakketselectie en het proces van de implementatie van een DBMS.

Tevens wil ik de docenten dhr Mantel en mevr Bechet-Tjoonk bedanken voor hun feedback op mijn documentatie.

Erik van den Berg
Hilversum, 28 september 2007

Inhoudsopgave

1.0 Inleiding

Dit verslag is opgesteld naar aanleiding van het 20 weken durende afstudeerproject "**Pakketselectie en implementatie van een prototype bij Spill Productions**" dat ik heb uitgevoerd als onderzoeksproject voor het bedrijf Spill Productions, in het kader van het afstuderen bij de opleiding Informatica te Haagsche Hoge school.

Het doel van dit verslag is om inzicht te geven op het afgelegde voorbereidend afstudeertraject, om zo de examinatoren en de gecommitteerde in staat te stellen het voorbereidend afstudeertraject te kunnen beoordelen. In het verslag zult u lezen over de organisatie, de opdracht, de werkzaamheden, de onderzochte en toegepaste methoden, technieken en een evaluatie van de gemaakte producten.

Het procesverslag is als volgt opgesteld:

- In Hoofdstuk 2 wordt een schets van het bedrijf gegeven en van de opdrachtomschrijving.
- In Hoofdstuk 3 bespreek ik de voorbereiding op dit project, dat wil zeggen risicoanalyse, ontwikkelingstrategie, analyses van de huidige en de gewenste situatie en de standaard en richtlijnen voor het project en de projectplanning.
- In hoofdstuk 4 beschrijf ik Scalability en Availability en wat daaronder verstaan wordt bij database management systemen en services in het algemeen.
- In hoofdstuk 5 ga ik een pakketselectie methode definiëren die ik tijdens het project ga gebruiken.
- In hoofdstuk 6 beschrijf ik de longlist en plaats ik de uiteindelijk gekozen longlist.
- In Hoofdstuk 7 ga ik in op het shortlist gedeelte van het pakket selectie traject . Hierbij beschrijf ik tevens de test configuraties die ik gemaakt heb bij elke gekozen database management systeem.
- In hoofdstuk 8 beschrijf ik mijn conclusies en advies voor Spill Productions met betrekking tot een database management systeem.
- In hoofdstuk 9 beschrijf ik mijn evaluatie van het product en de procesgang.

2.0 Bedrijf en Opdracht

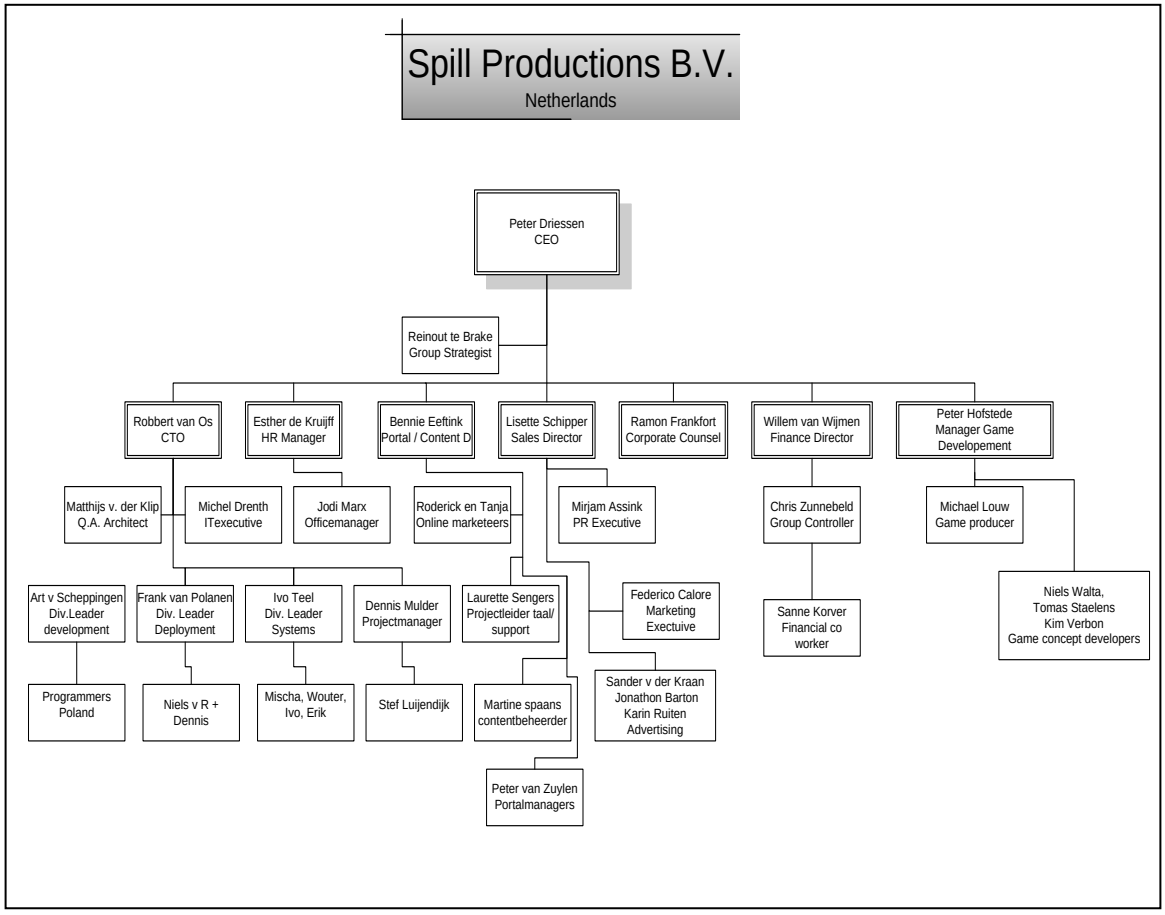
2.1 Inleiding

In dit hoofdstuk wordt het bedrijf, Spill Productions BV, beschreven evenals de door hun beschikbaar gestelde afstudeer opdracht. Hierin wordt beschreven wat voor organisatie Spill Productions is. Vervolgens wordt ingegaan op de gemaakte afstudeer- opdracht, om zo inzicht te geven in de gemaakte opdracht.

2.1 Bedrijf

Ik heb mijn afstudeer opdracht gedaan bij het bedrijf Spill Productions BV.
Spill Productions is gespecialiseerd in online gaming portals en beheert tientallen hiervan. Met afdelingen in Amerika, China en Polen is Spill een van de grotere casual gamesproviders van Europa.

Mijn plaats in het geheel is te zien in onderstaand organogram, linksonderaan bij *Ivo Teel, Div. Leader Systems.*



Tabel 1: Organogram van Spill Production's organisatie

Spill productions heeft ook, net als elk ander bedrijf concurrenten. De websites van enkele concurrenten zijn:

- http://www.miniclip.com
- http://www.pogo.com
- http://Games.yahoo.com
- http://www.games.com
-

Volgens comScore, staat Spillgroup nu in de top 5. Comscore die metingen en statistieken op het internet bijhoudt. ComScore's top 5 wereldwijd in online gaming portals zijn [1]:

1 Yahoo! Games
2 MSN Games
3 MINICLIP.COM
4 EA Online
5 Spill Group

Spill Group is in verschillende landen marktleider, en heeft als doelstelling om eind 2008 wereldmarktleider te worden. Enkele landen waarin Spill Group marktleider is zijn:

- Polen
- Frankrijk
- Duitsland
- Italië
- Zweden
- Brazilië

2.2 Opdracht

Aanleiding
De aanleiding van de opdracht zijn de recente uitbreidingen van Spill Productions. Hierdoor moet er aanzienlijk rekening gehouden worden met het design van de netwerken waarop de software van Spill Productions draait.

Spill productions heeft een groot belang bij een goed draaiende database, omdat hierin advertenties en statistieken worden opgeslagen. De advertenties zijn een grote bron van inkomsten voor Spill Productions. Tevens wordt allerlei informatie over spelletjes opgeslagen in verschillende databases. Mocht de database server niet draaien, dan is de site niet opvraagbaar voor bezoekers.

[1] http://www.comscore.com

Probleemstelling
Door de groei van de laatste tijd begint de vraag te komen bij de organisatie of hun huidige database servers nog wel schaalbaar zijn met de groei van het bedrijf. Door deze groei wordt de drukte op de database servers namelijk steeds groter en kunnen deze vertragingen gaan opleveren, of zelfs helemaal uitvallen. De oplossing die Spill Productions voor ogen heeft:

- Een onderzoek naar een eventueel alternatief DataBase Management Systeem (vanaf nu DBMS genoemd), alsmede moet er bekeken worden of de huidige DBMS mee zou kunnen groeien met de groei van het bedrijf, of dat deze anders geconfigureerd dient te worden.

2.3 Doelstelling

Om het probleem van Spill Productions op te lossen is het doel van de afstudeeropdracht: Vind een DBMS dat geschikt is om de groei van Spill Productions aan te kunnen

De documentatie bij het pakketselectie traject en van de opstelling van het prototype dienen tevens in het Engels opgeleverd te worden aan Spill Productions. De opdrachtomschrijving (zoals te vinden in de bijlagen I en II) dient in het Nederlands en in het Engels opgeleverd te worden. Dit is vanwege het internationale karakter van Spill Productions.

De op te leveren producten zoals deze in de opdrachtomschrijving omschreven zijn:
- Plan van aanpak (in het Nederlands en Engels)
- Pakketselectie onderzoek rapport en conclusie (in het Engels)
- Prototype opstelling rapport (in het Engels)
- Rapport met aanbevelingen over de migratie (in het Engels)
- Eindverslag procesgang (in het Nederlands)

3.0 Voorbereiding

3.1 Inleiding

Dit hoofdstuk beschrijft het voorbereidende werk dat ik heb uitgevoerd om tot een oplossing voor het probleem van de opdrachtgever te komen. Dit hoofdstuk behandelt het voorwerk met betrekking tot de risico analyse, pakket selectie strategie, gekozen oplossing, standaarden en richtlijnen, te gebruiken methoden en technieken, de beschikbare faciliteiten en de projectorganisatie.

3.2 Risico analyse

Ik kan tijdens het uitvoeren van het project de volgende ingecalculeerde risico's tegenkomen. Hierbij wordt aangegeven hoe ik denk het betreffende risico kan tegen gaan of eventueel beperken. Deze risico's zijn tot standgekomen via ervaring die de ik heb opgedaan uit vorige projecten.

De volgende risico's en oplossing zijn voor dit project ingecalculeerd:

Risico:	Opdrachtgever ziet tijdens het project af van de opdracht.
Oplossing:	geen. De tot dan toe opgeleverde producten zoals documentatie opleveren aan school.
Risico:	Door gebrek aan tijd kan de eindoplossing niet volledig gerealiseerd worden.
Oplossing:	In overleg met de opdrachtgever besluiten welke gedeeltes wel gerealiseerd moeten worden. De deadline verlengen in overleg met de opdrachtgever/school.
Risico:	Bedrijfsbegeleider valt door ziekte of door een andere reden tijdelijk of geheel uit.
Oplossing:	Aanvankelijk proberen door te werken, mocht dit niet lukken in overleg met de opdrachtgever deadline verlengen of functionaliteit laten vervallen. Eventueel is het ook mogelijk dat een andere begeleider wordt aangewezen.
Risico:	Het kennisniveau van de projectgroep schiet tekort voor de implementatie van de eindoplossing.
Oplossing:	Hulp inschakelen van experts.

Deze risico's zijn tot stand gekomen door ervaringen van voorgaande projecten bij de Haagse Hogeschool en bij stages en projecten op mijn vorige opleiding (mbo technische informatica).

3.3 Vaststellen wensen en eisen

Via de opdrachtomschrijving en gesprekken met de opdrachtgever werd vrij snel duidelijk wat de wensen en eisen waren. De opdrachtgever gaf mij zeer veel vrijheid om de hoofdvraag van de opdracht in te vullen.

De hoofdvraag luidt:

Kan er nog wel doorgegaan worden met MySQL of moet er overgeschakeld worden op een ander DBMS?

Daarnaast had de opdrachtgever nog enkele andere vragen die onderdeel zijn van de hoofdvraag. Hieronder staan die vragen cursief en schuingedrukt, aangevuld met mijn eigen vragen, de tekst eronder is mijn interpretatie van de vragen. Deze interpretaties zijn ook voorgelegd aan de opdrachtgever en die is het eens met mijn interpretatie.

MySQL Community of MySQL Enterprise?

Spill Productions draait nu op MySQL DBMS'en. MySQL AB (bedrijf dat eigenaar is van MySQL) heeft sinds niet al te lange tijd 2 versies beschikbaar, MySQL Community & MySQL Enterprise. De opdrachtgever wil graag weten wat het verschil tussen deze twee pakketten is.

Alternatieve DBMS'en (oracle, postgresql, enz)

Spill productions wil graag weten wat er voor alternatieven zijn voor MySQL. Welke DBMS'en zijn er nog meer beschikbaar op de markt, zowel in de commerciële sector als in de open source sector.

Opensource of commercieel DBMS

Spill productions wil ook een rapport waarin het verschil/overeenkomsten tussen opensource en commerciële DBMS'en wordt beschreven. Na een interview werd duidelijk dat hier voornamelijk de licentie modellen werd bedoeld.

Scalability
Wat zijn de mogelijkheden om te schalen, welke technieken zijn er hiervoor beschikbaar en hoe zijn deze toe te passen op de situatie van Spill Productions? Is de manier van scaling die nu gebruikt wordt bruikbaar voor de toekomst? Al deze vragen dienen beantwoord te worden.

Database clustering als alternatief voor replicatie
Spill productions gebruikt nu de techniek “replicatie” om backups en failovers van hun DBMS’en te maken. Ze hebben gezien dat clustering een alternatief hiervoor is en vragen zich af hoe dat toepasbaar is op hun situatie.

Rekening houden met huidige database migratie naar nieuwe DBMS omgeving
Wanneer er een andere DBMS dan MySQL uit de bus komt, hoe zit het dan met het migreren van de huidige databases naar deze nieuwe DBMS? Kan dit zomaar, en welke problemen kunnen zich voordoen?

Kosten, upgrades en support
Welke kosten komen er bij kijken voor de aanschaf van de DBMS, welke licentie dient er genomen te worden als dit van toepassing is. Hoeveel kosten upgrades, en komen deze met regelmatige periodes uit. Hoe zit het met de support die door de maker van de DBMS erbij wordt geleverd?

Onderhoud
Zijn er goede tools beschikbaar voor onderhoud? Is er duidelijke externe ondersteuning mochten er problemen ontstaan? Is het onderhoud ‘makkelijk’ uit te voeren, en kan dit uitgevoerd worden zonder dat de DBMS offline moet?

3.4 Analyse huidige situatie

In dit gedeelte geef ik aan hoe ik de huidige situatie bij Spill Productions heb geanalyseerd. Ik begon met het opstellen van een vragenlijst om zoveel mogelijk te weten te komen over de huidige DBMS setup in Spill Productions. Met deze informatie kan ik dan rekening houden tijdens het doorlopen van het pakketselectie traject en het opstellen van de prototypes.

De volgende vragenlijst heb ik samengesteld, waarbij ik bij elke vraag toelichting geef waarom ik deze vraag heb gesteld. Hierna zijn al deze vragen beantwoord door mijn stagebegeleider, dhr Teel. De vragen zijn grotendeels door mijzelf geformuleerd. Deze vragen zijn geformuleerd aan de hand van mijn eigen kennis en ervaring niveau om een zo duidelijk mogelijk beeld te krijgen van de huidige situatie. Ik ben uitgegaan van het hoe / wat principe. Dat is terug te vinden in de vraagstelling als ik deze in een tabel tegenover elkaar zet.

Vraag nummer	Hoe	Wat
1		X
2	X	
3		X
4	X	
5	X	
6		X
7	nvt	nvt
8		X
9		X
10		X
11	nvt	nvt
12	nvt	nvt

Tabel 2: Vragenlijst

Bij “hoe” kan bijvoorbeeld gedacht worden aan:

- Hoe zit de huidige setup van servers in elkaar
- Hoe is backup geregeld

Bij “wat” kan gedacht worden aan:

- Wat is de huidige hardware
- Wat zijn bedrijfskritische servers

De vragen 11, 12 en 13 zijn er na een interview met de opdrachtgever bijgekomen. De overige vragen waren voor het eerste interview al bedacht en beginnen op de volgende pagina.

1. *What kind of DBMS does Spill Productions run as of today?*

Met deze vraag stel ik vast welke DBMS er op dit moment gebruikt wordt bij het bedrijf. Dit is belangrijk omdat ik verdere vragen kan specifieker kan richten op MySQL in plaats van algemene vragen te hoeven vragen.

2. *How many servers that are in production run MySQL?*

Met deze vraag krijg ik een beeld van de organisatie in Spill Productions.

3. *What is the name and role/purpose of each server (masters, slaves, etcetera)*

Met deze vraag begin ik al te kijken naar wat mogelijkerwijs bedrijfskritische servers kunnen zijn. Hierdoor krijg ik ook een idee wat de master en de slave servers zijn.

4. *How big is the database of each server (in megabytes or gigabytes)*

Ik krijg nu een idee van de grootte van de databases, wat van belang is tijdens het doorlopen van het pakketselectie traject.

5. *What kind of query does every server get mostly? (update? Insert? select?)*

Deze vraag is relevant om te zien wat voor soort load een DBMS krijgt. Dit heeft weer invloed op het soort prototype wat ik uiteindelijk ga configureren.

6. *What is the current hardware the servers run on?*

Met deze vraag prober ik inzicht te krijgen op wat voor soort hardware de DBMS'en nu draaien, en op wat voor hardware mijn gekozen pakket eventueel gaat draaien.

7. *Are all the servers dedicated mysql servers?*

Dit is een belangrijke vraag om zo erachter te komen of alle server kracht (cpu, geheugen, harddisk) wel gebruikt wordt het meeste uit de DBMS te halen qua aantal transacties die verwerkt kunnen worden. Als er bijvoorbeeld een webserver op draait neemt deze aanzienlijk geheugen en cpu cycles in beslag die gebruikt zouden kunnen worden voor de DBMS.

8. *What is the amount of queries all of the servers generally get?*

Met deze vraag krijg ik een goede indruk wat de drukste servers van Spill Productions zijn. Mijn pakketselectie traject zal zich baseren op het antwoord van deze vraag en het antwoord op vraag 10

9. *What operating system does every server run on, including version numbers ?*

Met deze vraag krijg ik te zien op welke operating systems de servers draaien, inclusief kernelversie, distributie versie. Het is al bekend dat alle servers op het Linux operating system draaien.

10. *What filesystem do the servers run on ?*

Deze vraag is relevant om bij een DBMS die veel data moet lezen een filesysteem hoort wat snel data kan inlezen van een hardeschijf. Dit is een punt waar rekening mee gehouden moet worden bij het pakketselectie traject.

11. *What database(s) are considered business critical ?*

Dit is de belangrijkste vraag. Omdat Spill Productions' core business is online gameportals, dienen deze altijd online te zijn. Een essentieel onderdeel daarvan zijn de databases, die gebruikersnamen hebben opgeslagen, maar ook de advertisement database. Deze database bevat de reclames die op de game portals worden laten zien. Deze database mag onder geen beding ooit offline zijn.

12. *There is a mention of application conversion besides data conversion if a choice is made for a different DBMS, what kind of applications do you consider that need to be 'converted' ?*

Spill Productions maakt gebruik van bepaalde php classes die toegespitst zijn op MySQL. Spill Productions will graag dat rekening gehouden wordt met conversie naar een andere DBMS. Ook maar Spill Productions gebruik van het opensource pakket OpenADS voor het distribueren van de Ads op de verschillende game portals.

13. *Do you have more detailed statistics then the one contained in http://spi087.spillgroup.com/admin/rrdtool/graphs/compare.php*

De genoemde website laat bepaalde statistieken over een server zien, zoals geheugengebruik, aantallen queries, load gemiddelde. Deze statistieken zijn helaas niet zo heel precies, omdat er over grotere periodes gemeten wordt. Het is bijvoorbeeld onmogelijk om aantallen transacties per minuut te kunnen zien. Er kan alleen per week of per 24 uur gekeken worden. Zo kan er wel een gemiddelde gevormd worden, maar is het moeilijk om te zien hoeveel transacties er gedaan worden op piektijden.

De antwoorden op de vragen zijn terug te vinden in bijlage VI.

Het antwoord op vraag 11 gaf 4 database servers op die als belangrijkste voor Spill Productions worden gezien. Deze 4 heb ik nader beschreven daarna, en een korte analyse gemaakt van wat hun bezigheden zijn. Dit hield in, aantallen key lookups, transaction questions, table locks en connecties die in gebruik zijn. Deze informatie is opgehaald met de tool mysqlreport en kan teruggevonden worden in de externe bijlagen.

Hieronder nog een overzicht van deze 4 database servers:

- Server 1: ZIG082
    - Dit is de database van www.zigiz.nl, met ZIG081 als slave.
- Server 2: SPI086
    - Dit is de database van voor de statistieken, met als slave SPI087.
- Server 3: SPE080
    - Dit is de database van www.spelletjes.nl met SPE081 als slave.
    - Voornaamste taak de zoekmogelijkheid op de site.
- Server 4: SPI084
    - Dit is de ads database server, en heeft als slave SPI085.
    - Maakt gebruik van het open source programma OpenADS.

Het hele pakket selectie traject zal zich richten op het vervangen/aanpassen van deze 4 servers.

3.5 Analyse gewenste situatie

Spill Productions gaf mij zeer veel vrijheid in het definiëren van de uiteindelijk gewenste situatie, omdat ze zelf al van veronderstelling zijn dat de huidige situatie echt niet meer kan. De volgende lijst met globale eisen zijn wel aangegeven:

- Lage TCO (Total Cost of Ownership)
    - Hierin is dan weer niet aangegeven hoeveel er exact uitgegeven mag worden, maar het idee is om de licentie kosten zo laag mogelijk te houden.
- Volledige ACID[1] transactie ondersteuning.
    - De ACID methode is een garantie dat transacties op de database worden uitgevoerd. Dit voorkomt dat er maar een halve transactie wordt uitgevoerd.
- Aanwezige support/upgrades voor de gekozen oplossing.
    - Dit is meer een garantie voor de toekomst dat het product ondersteund blijft, en een terugval mogelijkheid om hulp in te schakelen mochten er problemen ontstaan die systeembeheer zelf niet kan oplossen met betrekking tot de gekozen DBMS.
- Schaalbare oplossing (scalability).
    - Gezien de grote groei die Spill Productions meemaakt, moet de oplossing ook goed mee kunnen schalen. Er zijn nu ongeveer 40 miljoen unieke bezoekers op de websites van Spill Productions per maand (juni 2007) maar eind van dit jaar (december 2007) wordt het aantal bezoekers al geschat op rond de 60 miljoen per maand. Ondertussen zijn het al 60 miljoen bezoekers per maand geworden (sinds augustus 2007).
- Redundancy (availability).
    - De inhoud van de database moet nog beschikbaar zijn als een database server uitvalt wegens hardwareproblemen of connectiviteits problemen.

Zelfs na enkele interviews met dhr Teel werd nog niet duidelijk hoe groot het budget precies kon zijn. Er is wel aangegeven dat de TCO zo laag mogelijk moet zijn bij de gekozen oplossing. Spill productions staat voor elke prijs open, mits de kwaliteit / prijs verhouding goed zijn.

[1] **ACID** (**A**tomicity, **C**onsistency, **I**solation, **D**urability) zijn een aantal eigenschappen om te zorgen date en database transactie betrouwbaar wordt uitgevoerd In het context van databases, is een transactie een enkele logische operatie op de data in de database.

3.6 Te volgen strategie

Het vastellen van de te volgen strategie houdt in dat er gekeken moet worden naar een methode om de hoofd- en bijvragen uit het stukje hierboven op te lossen. De methode van pakketselectie ligt voor de hand, met name door de bijvragen “scalability” en “database clustering als alternatief voor replicatie”. Zonder deze vragen zou er ook nog gekeken kunnen worden naar optimalisatie van de huidige DBMS, en dan met name optimalisatie van de tabellen op de verschillende databases die er hier gedraaid worden.

Doordat deze 2 bijvragen wel gesteld zijn, is het logisch om voor een pakketselectie methode te kiezen om een nieuwe DBMS te vinden. In het kort houdt pakketselectie in het selecteren van een nieuw (software) pakket. Pakketselectie is vaak impliciet of expliciet onderdeel van een veranderings- of verbeteringstraject binnen een organisatie. Dit hoeft dus niet te houden dat er een ander pakket gebruikt moet worden, maar dat het huidige pakket alleen verbeteringen dient te ondergaan.

Tijdens het vaststellen van de opdracht omschrijving heb ik het nog met mevr Bechet-Tjoonk gehad over een projectbeheersingsmethode. Zij raadde mij aan om eens te kijken naar de projectbeheersingsmethode van Roel Grit. Ik heb deze methode op internet bekeken maar het was weinig toegevoegde waarde voor mij, mede omdat het neerkwam op de manier van project werken zoals ik het al gewend was om te doen van de Haagse Hogeschool. Verder is het projectmanagement voor een klein deel op de prince 2 methode gebaseerd, hoewel ik deze nooit gehad heb op de Haagse Hogeschool.

Omdat ik dit project alleen uitvoer, vind ik het niet bijster noodzakelijk om een strakke projectbeheersingsmethode te volgen. Mocht het team uit meerdere personen hebben bestaan, dan had ik waarschijnlijk wel de methode van Prince2 of Roel Grit in zijn geheel gebruikt.

3.7 Faciliteiten

Als faciliteiten heb ik meerdere machines tot mijn beschikbaar om test situaties op te creëren. Verder heb ik een machine op mijn desktop om documentatie op te typen. Op deze machine heb ik ook een VMware licentie gekregen om VMware desktop te draaien. Hierop kan ik allerlei configuraties op testen.

3.8 Projectorganisatie

De projectorganisatie laat zien wie welke verantwoordelijkheden binnen het project heeft. Aangezien ikzelf (de student) de uitvoerder van de opdracht ben vervul ik alle rollen met betrekking tot het project. De opdrachtgever heeft ook inmenging in het project maar dat valt meer onder controlerende taken dan uitvoerende taken.

De student heeft de volgende verantwoordelijkheden:

- Uitvoeren van het pakketselectie onderzoek.
- Documenteren van het pakketselectie onderzoek.
- Prototype opstellen en testen.
- Overdragen documentatie.
- Rapport met aanbeveling over de migratie.

De opdrachtgever heeft de volgende verantwoordelijkheden:

- Ondersteunt de student bij vragen over de opdracht
- Geeft de oplossingrichting aan
- Geeft opmerkingen / feedback over de aangeleverde documentatie

Wekelijks vindt een evaluatie gesprek plaats over hoe het project ervoor staat en of ik nog tegen problemen ben aangelopen.

3.9 Planning

De planning van het project komt er als volgt uit te zien:

Week	**Datum**	**Opdracht**
Week 1 tm 3	14-05-'07 tm 01-06-'07	Plan van aanpak, doornemen beschikbare documentatie over DBMS' en. Eisen/wensen samenstellen.
Week 4 tm 5	04-06-'07 tm 15-06-'07	analyse van database capaciteit door Spill, afbakening eisenpakket.
Week 6 tm 7	18-06-'07 tm 29-06-'07	verschillen tussen Opensource en (closed) commerciële DBMS'en beschrijven.
Week 8 tm 12	02-07-'07 tm 03-08-'07	Longlist en shortlist opstellen.
Week 13 tm 14	06-08-'07 tm 17-08-'07	Conclusie Longlist, mogelijke migratiemoeilijkheden.
Week 15 tm 17	20-08-'07 tm 07-09-'07	Conclusie opstellen, prototype, migratie aanbevelingen.
Week 18 tm 20	10-09-'07 tm 28-09-'07	Uitloop.

Deze planning beschrijft globaal de werkzaamheden en in welke periodes deze plaatsvinden.

4.0 Scalability en Availability

4.1 Inleiding

Ik heb gekozen om een apart hoofdstuk aan scalability en availability te wijden om hierin dieper in te gaan op de twee types van scalability en wat high availability inhoudt. Dit omdat deze twee onderwerpen van cruciaal belang zijn voor Spill Productions.

Scalability omdat ze momenteel een zeer grote groei ondergaan, zowel als in personeel als in bezoekers op hun game portals. Het is onbekend wanneer/of deze groei zal stagneren en daarom is het noodzakelijk dat de gekozen oplossing goed schaalbaar is.

High availability omdat Spill Productions diensten verleent over het internet, en de meeste inkomsten gegenereerd wordt via banners en ads op hun gameportals, kan Spill Productions het zich verloorloven dat deze systemen niet meer bereikbaar zijn voor bezoekers.

4.2 Scalability

Er zijn twee types scalability, namelijk “horizontal scaling” en “vertical scaling”, ook wel “scaling out” en “scaling up” genoemd. Deze twee types van scaling zijn wel degelijk van belang bij het opzetten van de testconfiguraties en de bijbehorende shortlist.

Vertical scaling
Vertical scaling houdt in dat er omhoog gescaled (geschaald) wordt. Dat wil zeggen, als men 1 server heeft, en men gaat deze hardware matig uitbreiden (bijvoorbeeld door meer geheugen erin te zetten, meer hardeschijf ruimte en een snellere processor) dat dit ”vertical scaling” wordt genoemd. Vertical scaling heeft zijn voor- en nadelen tegenover “horizontal scaling”, maar daar kom ik later op terug.

Voordelen:

- Server is gemakkelijk hardwarematig uit te breiden
- Geen softwarematige configuratie veranderingen hoeven plaats te vinden bij een uitbreiding van de hardware

Nadelen:

- Kosten, een nieuwere computer/hardware uitbreiding kost aanzienlijk wat geld
- men ‘verspilt’ processpower (meer hierover na het kopje over ”Horizontal scaling”)

Horizontal scaling
Horizontal scaling houdt in dat er in de breedte wordt gescaled. Dat wil zeggen dat wanneer een server niet meer voldoende capiciteit heeft dat er een 2e server naast wordt gezegd. In principe zal dit de capaciteit verdubbelen.

Voordelen:
- Goedkoper om meer capaciteit toe te voegen, aangezien het niet de nieuwste van de nieuwste hardware hoeft te zijn.
- Geen processpower verspilling

Nadelen:
- Er zullen softwarematige configuratie veranderingen plaats moeten vinden om de nieuwe machine toe te voegen aan de configuratie

Op de volgende pagina is nog een tabel met daarin de overzichten van horizontale scaling tegenover verticale scaling.

	Verticale scaling	Horizontale scaling
Scalability	Grote upgrade om capaciteit toe te voegen. Vervang bestaande SMP server met grotere SMP server. Voorbeeld: Vervang 12 SMP CPU machine met 24 SMP CPU machine.	Voeg capaciteit toe als dat nodig is. Verspreid de load gelijkwaardig over de servers. Voeg incrementeel 'goedkope'* servers toe. Voorbeeld: Vervang 12 CPU SMP server met 24 'goedkope'** servers.
Availability	Een hardware probleem kan gevolgen hebben voor de availability van de DBMS.	Hardware problemen zijn geïsoleerd tot 1 enkel systeem, en hebben (bijna) geen impact op de gehele DBMS.
Manageability	Eenvoudig om een enkele SMP machine te beheren.	Complexiteit neemt toe als het aantal servers toeneemt.
Affordability (hardware cost)	12 CPU SMP machine $600.000***	20 'goedkope' servers $75.000***

Tabel 3: Scaling types [1]

*) SMP betekent Symmetric Multi Processing, wat inhoudt dat er meerdere CPU's in een server zitten.
**) Met goedkoop worden geen kwalitatief slechte servers bedoeld, maar eventueel oudere, maar betrouwbare modellen van een server.
***) Deze cijfers zijn gebaseerd op een scale out artikel van MySQL, zie de referenties in bijlage VIII.

[1] MySQL Whitepaper: Guide to Cost-effective Database Scale-Out using MySQL

Processor power verspilling
Vertical scaling leidt tot process / geheugen 'verspilling'. Men zit bijvoorbeeld aan de maximum capaciteit van een server. Men besluit om een nieuwe, krachtigere server aan te schaffen die zeker nog een jaar mee kan gaan. Het zal dus nog een jaar duren voordat de nieuwe server aan zijn top zit. Ondertussen zit er 'teveel' processor power en geheugen in de server dat niks zit te doen. Dit wordt zo gedaan omdat men ook niet om de maand een nieuwe server wil aanschaffen.

Onderstaande figuur 1 geeft de processor/geheugen verspilling in paars en de load/throughput van een server in oranje weer.

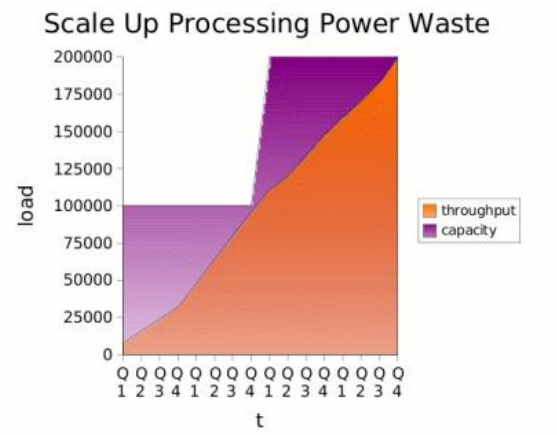


fig 1 Verticale scaling [1]

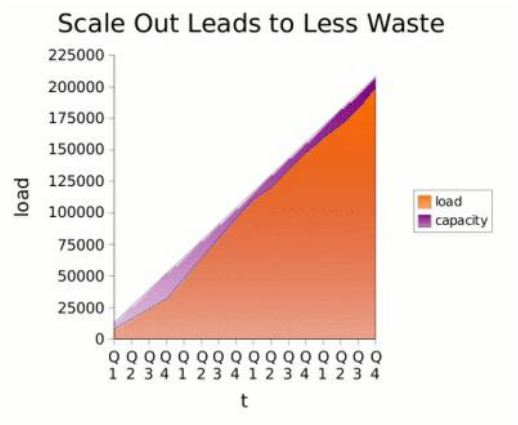


fig x Horizontale scaling [1]

Het is ook mogelijk om een combinatie te maken van horizontale en verticale scaling door bijvoorbeeld een cluster te maken van 4 machines (horizontale scaling) en wanneer deze niet meer voldoet, de 4 machines hardwarematig te upgraden (vertical scaling). Natuurlijk heeft deze optie ook dezelfde nadelen zoals deze op de vorige pagina zijn beschreven bij verticale scaling.

Na een gesprek met dhr Teel werd duidelijk dat de voorkeur van Spill Productions ligt in horizontale scaling, wat ik zelf ook de meest logische keuze vind, omdat deze optie zorgt voor de minste processor/geheugen verspilling. Tevens zorgt deze optie voor meer redundantie, omdat bij uitval van een server de capaciteit teruggaat, in plaats van helemaal wegvalt.

[1] http://jpipes.com/index.php?/archives/175-The-Ambiguously-Vague-Duo-Scale-Out-and-Scale-Up.html

4.3 High Availability

De vertaling van High Availability is hoge beschikbaarheid. Beschikbaarheid kan gedefinieerd worden als:

Continue toegang tot applicaties, met een voorspelbaar service-level.

Belangrijk hierbij is te beseffen dat dit geldt zoals de eindgebruiker dit beleeft: een systeem kan goed werken, de applicatie kan draaien, maar als een gebruiker er niet bij kan (door bijvoorbeeld een probleem in het netwerk) is de applicatie voor de gebruiker toch niet beschikbaar. Ook plotseling langer durende responstijden geven de gebruiker een gevoel van niet/mindere beschikbaarheid.

High availability is een belangrijke zaak voor Spill Productions, omdat zij een 'internet' bedrijf zijn, dienen hun websites ten alle tijde beschikbaar te zijn. 100% uptime is technisch niet haalbaar omdat veel zaken buiten het eigen beheer vallen. Denk hierbij aan kabels die kapot kunnen gaan omdat er graafwerkzaamheden ergens zijn, denk hierbij aan het feit dat Spill Productions geen internet provider is, en dus de internet lijnen ook weer moet huren bij een data center.

Toch wil Spill Productions de hoogst mogelijk uptime/bereikbaarheid van hun servers halen. Niet alleen omdat downtime inkomsten derving is, maar ook vanwege de volgende redenen:

- Verminderde productiviteit.
- Interne en externe ontevredenheid van klanten.
- Slecht imago.
- Claims.
- Verlies van bestaande klanten.

Er zijn 2 soorten downtime, namelijk ge*plande downtime* en *ongeplande downtime*. De oorzaken van *ongeplande downtime* zijn terug te brengen tot de 3 P's categorie, namelijk Product-People-Process.

- **P**roduct (20%)[1]
  Hierbij is te denken aan servers, storage, applicaties, databases, power/omgeving. 20% van unplanned downtime wordt veroorzaakt door de gebruikte producten.
- **P**eople(40%)[1]
  Eindgebruikers, systeembeheerders, service personeel, change management
  Risicofactoren bij mensen zijn: training, ervaring, coaching, screenen van externe mensen, de eindgebruiker, service personeel. Ook de organisatiestructuur is van belang: als er geen vervanging is voor iemand is dat een potentieel risico (Single Point of Failure in mensen)
- **P**rocess (40%)[1]
  Backup, "best practices", change management, problem management
  Bij processen moet er gedacht worden aan ITIL, maar ook aan procedures rondom backup/recovery. Met name recovery wordt nogal eens overgeslagen. Men maakt backups maar weet niet hoe deze terug te zetten zijn.

[1] http://www.nluug.nl/events/vj01/papers/muyzer.pdf

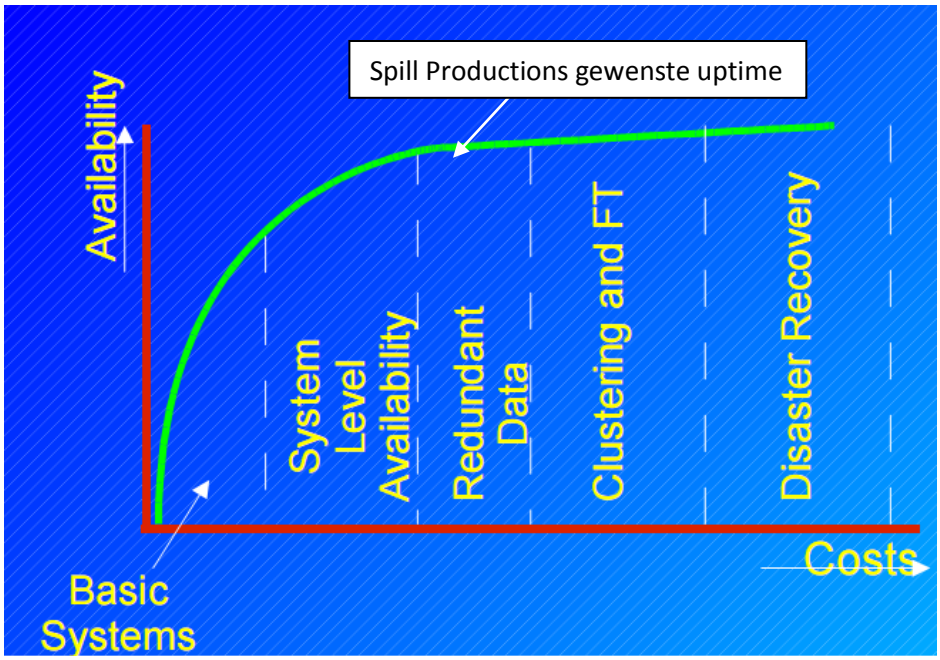
Hieronder volgt een beschikbaarheids matrix en het type configuratie wat voor de uptime zorgt [1].

Beschikbaarheid	Totale unplanned downtime per jaar	Totale unplanned downtime per week	Positionering
90%	> 1 maand	17 uur	Onbekend
99%	< 4 dagen	2 uur	Slecht
99,9%	< 9 uur	10 minuten	Standalone systemen
99,99%	~ 1 uur	1 minuut	Geclusterde systemen
99,999%	> 5 minuten	6 seconden	Uitdaging voor geclusterde systemen
99,9999%	~ 30 seconden	0,6 seconde	Fault tolerant system
99,99999%	~ 3 seconden		Vliegtuig computers

Tabel 4: uptime percentages [1]

Spill Productions streeft ernaar om 99,99% of 99,999% uptime te halen. Het nadeel van een zo'n hoog mogelijke uptime zijn de kosten. Spill Productions heeft geen budget aangegeven voor dit project, maar als stel regel kan genomen worden, hoe hoger de uptime hoe hoger de kosten. De grafiek op de volgende pagina geeft goed weer dat er na een bepaald punt de kosten niet meer in verhouding staan met de extra availability dat wordt geleverd.

[1] http://www.jiploo.com/blog/99999-or-five-nines-uptime/



grafiek 1: Hoe hoger de availability, hoe hoger de kosten. [1]

Zoals in bovenstaande grafiek te zien is, is dat de kosten extreem stijgen hoe groter de availability wordt. Op een gegeven moment, tussen Clustering met FT (Fault Tolerance) en Disaster Recovery, vind ik persoonlijk, dat de availability winst niet meer opweegt tegen kosten die hiervoor gemaakt moeten worden. Alhoewel Spill Productions wel heeft aangegeven dat er geen budget beschikbaar is, moet een budget wel in alle redelijkheid in acht worden genomen. Men kan bijvoorbeeld ook hardware van een paar miljoen dollar neer zetten om de komende groei goed op te vangen, maar dit is natuurlijk geen reële oplossing.

Bovenstaande hoofdstukken geven duidelijk weer wat er met de concepten Scalability en Availability bedoeld wordt, zodat hier verder in het procesverslag en in bijlage IV verslag geen misverstand meer over bestaat.

[1] http://www.nluug.nl/events/vj01/papers/muyzer.pdf

5.0 Pakketselectie methode definiëren

5.1 Inleiding

Dit hoofdstuk beschrijft mijn bevindingen over pakketselectie methodes en mijn motivatie over de geselecteerde pakketselectie methodes.

5.2 Oriëntatie op pakketselectie methodes

Wanneer men een pakketselectie methode wil toepassen, en hierop zoekt op internet, komen al gauw de methodes van de bedrijven Indora en KPMG als beste resultaat naar boven.

KPMG is een bedrijf dat een aantal jaar geleden een pakketselectie methode heeft ontwikkeld, gebaseerd op verschillende bronnen (zie referenties aan het einde van dit verslag). KPMG's methode heet officieel "Succesvol selecteren van logistieke standaardpakketten (Vroe97)". Dit document is helaas niet vrij beschikbaar door KPMG, maar de verkorte versie, genaamd "Pakketselectie: de begin- en de eindfase uitgelicht" is dat wel.

Indora is een bedrijf dat voornamelijk advies levert aan organisaties. Dit advies varieert van CRM, E-Business en marketing- en strategieontwikkeling. Indora heeft, net zoals KPMG, ook een pakketselectie methode ontwikkeld. Deze pakketselectie methode is gebaseerd op die van KPMG. Met name de eerste stap is heel gelijk. Het verschil zit hem grotendeels in dat die van Indora veel praktischer is ingesteld dan die van KPMG, die vooral veel theorie bevat.

Ik heb een combinatie van beide pakketselectie methodes toegepast voor de pakketselectie bij Spill Productions. Hierbij heb ik KPMG's methode als basis gebruikt, en hier op verder gebouwd met de Indora methode. Dat wil zeggen, de vragenlijsten van Indora als voorbeeld gebruikt om mijn eigen vragen te formuleren voor de longlist. Dit is nodig bij elke pakketselectie omdat er geen 'standaard' vragen bestaan en er onnoemelijk veel verschillende pakketten voor allerlei onderwerpen zijn. De vragen die Indora geeft moeten gebruikt worden als richtlijn om eigen vragen te formuleren.

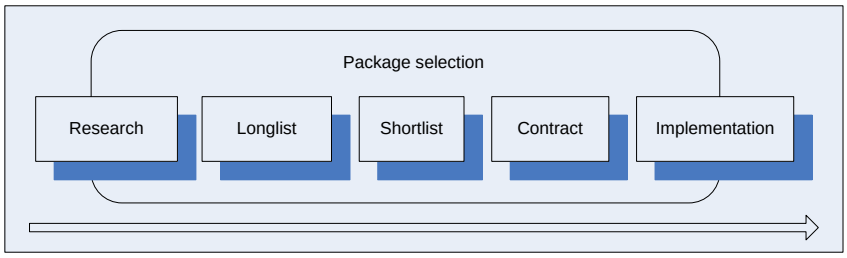


Fig 2: KPMG methode

De methode van KPMG kan naast die van Indora gelegd worden, waarbij de overeenkomsten duidelijk te zien zijn.

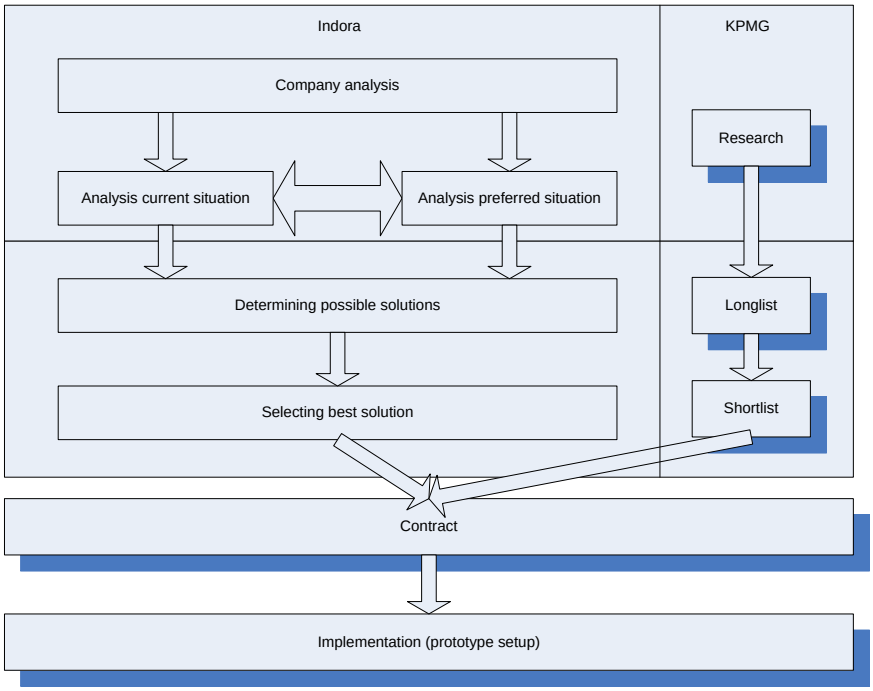


Fig 3: Indora en KPMG methode naast elkaar gezet

Het extended ISO model is goed terug te koppelen op de methodes van Indora en KPMG. Het extended iso model is een uitbreiding van het ISO model 9126 en verdeelt software kwaliteit in 6 categorieën, namelijk:

- Betrouwbaarheid (Reliability)
- Bruikbaarheid (Useability)
- Efficiëntie (Efficiency)
- Flexibiliteit (Portability)
- Effectiviteit (Functionality)
- Onderhoudbaarheid (Maintainability)

Door subcategoriëen zoals in fig 4 aan de bovenstaande categorieën te verbinden kunnen per categorie vragen geformuleerd worden die bij de longlist fase van pakketselectie gebruikt kunnen worden.

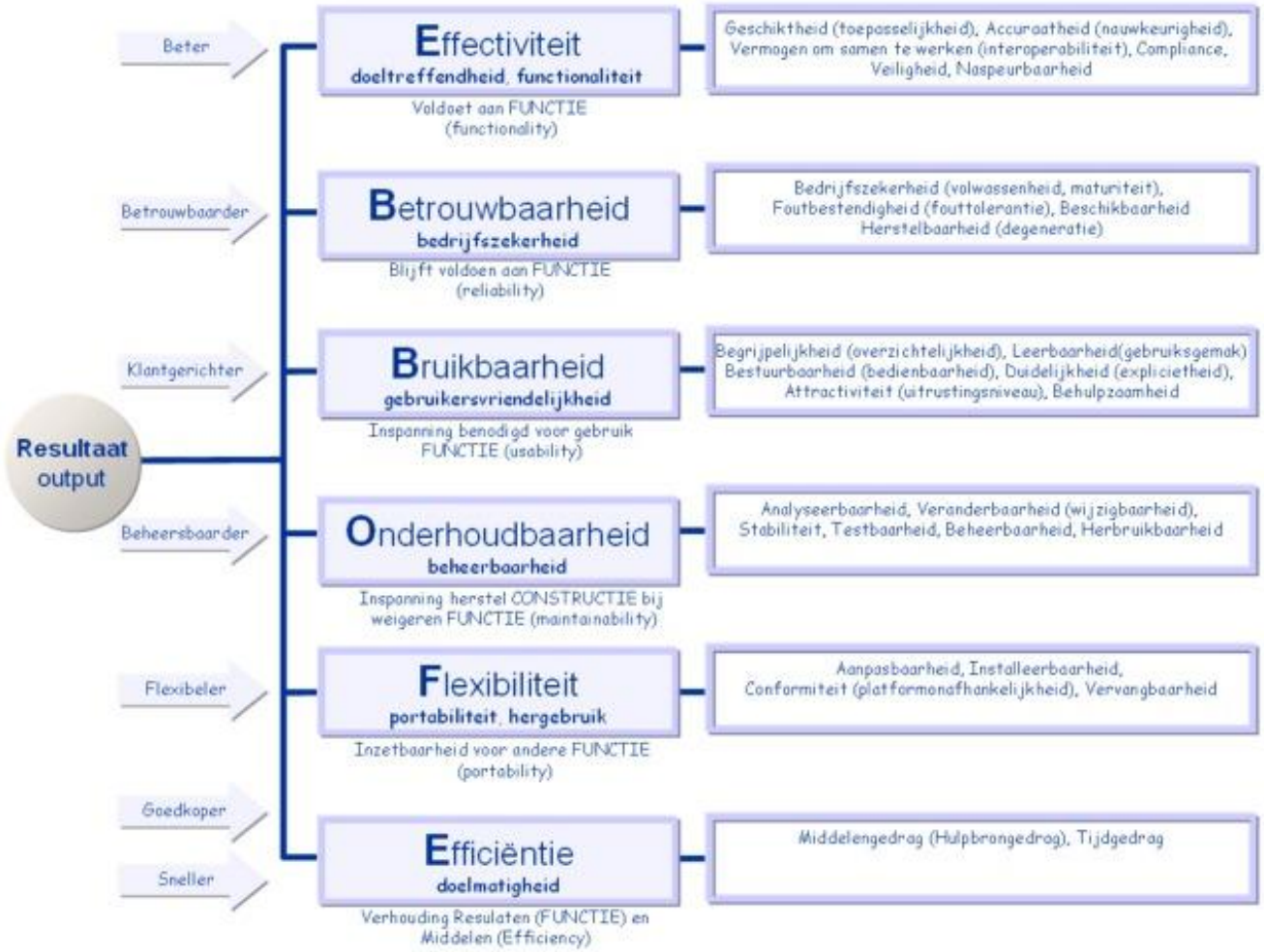


Fig 4: Het extended iso model 9126

Een andere veel gebruikte pakket selectie methode is die van Chau (beschreven in Elsevier in 1995 onder de titel van “Factors used in the selection of packaged software in small businesses: Views of owners and managers”). De methode van Chau werd gebruikt in de thesis “factoren die de keuze voor open source software bepalen” van Marco Theunissen. Hij gebruikt een checklist gebaseerd op die van Chau. Deze checklist viel voor mij af omdat de checklist zoals deze door Indora wordt voorgesteld veel uitgebreider en echt bedoeld is voor pakketselectie (hetzij open source of closed source). De checklist van Chau gaat dieper in op de verschillende types open source licenties. Desalniettemin zijn de groepen waarin Chau zijn vragen indeeld wel bruikbaar.

Chau’s vragenlijst is op te delen in 6 categorieën, namelijk:

- Technische criteria software
- Niet technische criteria software
- Technische criteria vendor
- Niet technische criteria vendor
- Opinies uit technische bronnen
- Opinies uit niet-technische bronnen

Voor de longlist maak ik gebruik van een combinatie van de KPMG/Indora methode, en voor de shortlist maak ik deels gebruik van Chau’s categorie vragen in combinatie met het extended ISO model 9126.

5.3 Verschillen Opensource en Commerciële database software

Alhoewel Spill Productions (nog) geen gebruik maakt van hele grote databases, vond ik het wel nodig om dit gedeelte ook op te nemen in het pakketselectie traject, omdat het in de toekomst misschien wel interessant is voor Spill Productions.

Ik heb het hoofdstuk in twee globale onderdelen ingedeeld, namelijk het verschil tussen licentie methodes, en het verschil in technische aard, wat de DBMS daadwerkelijk kan.

De voornaamste verschillen tussen de pakketten zijn dat de commerciële pakketten meer betrouwbaarheid hebben opgebouwd, omdat ze al langer bestaan. Oracle of DB2 (van IBM) bestaan bijvoorbeeld al bijna 30 jaar. Hierdoor hebben deze bedrijven veel ervaring opgebouwd met betrekking tot relationele databases. De commerciële databases ondersteunen ook vaker de minder gebruikte DBMS opties die nu nog in de kinderschoenen staan bij de opensource varianten.

Een paar voorbeelden van DBMS opties die nog nauwelijks ontwikkeld zijn in opensource varianten:

- Oracle, DB2 en Ingres kunnen met gemak database groottes van enkele terabytes aan terwijl MySQL hier al grote problemen mee heeft om snel dingen in op te zoeken.
- Synoniemen (aliasen voor tabellen, views) kan niet gedaan worden door MySQL of Postgresql, terwijl Oracle dit wel kan.
- Indexing, de meeste commerciele DBMS's ondersteunen meerdere types van indexing, zoals partial indexing[1] en bitmap indexing[2]. Het ligt wel in de verwachting dat de opensource DBMS's binnenkort ook meerdere indices gaan ondersteunen.
- Views, de commerciële DBMS's ondersteunen allemaal materialized views, terwijl geen van de opensource DBMS's dit ondersteunen. Materialized views zijn 'concrete' tabellen, in plaats van virtuele 'tijdelijke' tabellen. Dit verhoogt de efficiency aanzienlijk bij het aanroepen van de view.

[1] Partial indexing is indexing waarbij een conditie wordt toegevoegd zodat het alleen maar gedeeltes van de rijen in de tabellen laat zien.
[2] Bitmap indexing slaat het meeste van de data op als bitmap en beantwoordt queries via bitwise logical operations op de bitmap index.

Zoals eerder aangegeven, besteed ik ook een belangrijk deel van het pakketselectie traject in de type licenties waarin de Opensource DBMS'en verschillen met de commerciële DBMS'en. Als voorbeeld heb ik de grote licentie modellen gebruikt, zoals de GNU/GPL, BSD License en de licentie van Oracle. Ik beschrijf de verschillen tussen de licentie types en geef ook aan dat er hybride licenties bestaan die onder andere het bedrijf MySQL AB aanbiedt (deze zijn de eigenaars van de MySQL source code). Deze bieden hun software aan onder de GPL licentie, maar hebben ook aparte licenties waarin service en support aangeschaft kan worden. Hieronder een overzicht van de licentie modellen:

	Is allowed to mix with non free software	Changes in the code can be kept private	Can be licensed by anyone	Holds privileges for the original copyrightholder
General Public License (GPL)	No	No	No	No
Berkely System Distribution (BSD)	Yes	Yes	No	No
Mozilla Public License (MPL)	Yes	Yes	No	No

Tabel 5: Licentie schema [1]

5.4 Conclusie

Door in de eerste weken aandacht te besteden aan het definiëren van de opdracht en het uitzoeken van pakketselectie methodes kreeg ik meer inzicht in hoe pakketselectie werkte en op welke manier ik het beste pakket voor Spill productions kon uitkiezen.

Aan de hand van de longlist maak ik een scorelijst, waarbij ik de top 3 van die lijst meeneem naar het shortlist hoofdstuk in het pakketselectie traject. Door vraag 11 ben ik me ook gaan focussen op de type load die de 4 servers ondergaan. Deze load verschilt van databases die alleen maar lees opdrachten krijgen, en databases die gemixte lees en schrijf opdrachten krijgen. De rapporten van het programma *mysqlreport* komen daar heel veel bij van pas.

[1] Krishnamurthy, S. (2003), *A managerial overview of open source software*

6.0 Pakketselectie – Longlist

6.1 Inleiding

Dit hoofdstuk beschrijft de procesgang van het opstellen van de Shortlist voor pakketselectie. Hierin komt naar voren waarom ik voor de gekozen set van vragen heb gekozen, de beoordeling en de puntentelling van de selectie matrix en de uiteindelijk 3 DBMS keuzes.

6.2 Longlist

Mijn Longlist bestaat uit 8 DBMS's. Vraag is altijd, waar ligt de grens, omdat er ontzettend veel DBMS'en bestaan. Ik heb daarom bewust voor een even aantal gekozen zodat ik een helft van commerciële DBMS'sen en de andere helft uit opensource DBMS's kan laten bestaan.

Open source DBMS	Commercial DBMS
MySQL	Oracle
PostgreSQL	DB2
Firebird	SQL Anywhere
Ingres	Microsoft SQL Server

Tabel 6. Keuzelijst DBMS's voor de shortlist

De 8 DBMS's zijn gekozen aan de hand van wat ik op internet kon vinden wat 'populaire' DBMS's zijn. Mijn definitie van een populaire DBMS is, is dat wanneer het door grote aantallen gebruikers wordt gebruikt, en/of door een groot bedrijf (zoals Microsoft, IBM) is ontwikkeld. Er zijn nog wel meerdere kwaliteitscriteria om DBMS'en op te selecteren, maar dit leek mij een geschikte methode om de longlist te creëren. Het maakt niet uit welke methode ook gehanteerd wordt om tot een longlist te komen, er zal altijd de kans blijven bestaan dat potentiële kandidaten over het hoofd worden gezien, of dat minder goede kandidaten geselecteerd worden boven betere kandidaten.

Open source DBMS

- MySQL, dit is momenteel de populairste opensource DBMS. Deze is ook in de longlist gezet omdat Spill Productions nu al gebruik maakt van MySQL, en MySQL een zeer goede reputatie heeft als open source DBMS.
- PostgreSQL, dit is momenteel de nummer 2 opensource DBMS, zeer populair en qua performance gelijk aan MySQL.
- Firebird, dit is een 20 jaar oude database ontwikkeld door Borland, maar later vrijgegeven en opensource geworden.
- Ingres, ontwikkeld door de universiteit van Berkley zo'n 35 jaar geleden, bestaat nu nog steeds en is sinds 2004 opensource geworden. PostgreSQL is een spinoff van Ingress om enkele limitaties uit Ingres opgelost te zien worden. Mede omdat PostgreSQL een spinoff is van Ingres en een grote speler op de opensource DBMS markt, vond ik dat Ingres zelf ook in de longlist moest voorkomen sinds zij sinds een paar jaar zelf opensource zijn geworden.

Commerciële DBMS

- Oracle, dit is de grootste commerciële DBMS beschikbaar, wordt zeer veel gebruikt en heeft een goede reputatie.
- DB2, dit is de DBMS ontwikkeld door IBM. Ook een zeer oude DBMS.
- SQL Anywhere, dit is de DBMS van iAnywhere Solutions, die weer een onderdeel is van Sybase. Momenteel zijn er meer dan 10 miljoen SQL anywhere DBMS'en draaiend in de wereld. Zie meer hierover in het evaluatie gedeelte van het proces waarom deze keuze was gemaakt.
- Microsoft SQL Server, ook een zeer grote DBMS, origineel kocht microsoft de sourcecode over van Sybase om zich te mengen in de enterprise-level DBMS markt.

Met 'oude DBMS' wordt trouwens bedoeld dat deze al heel wat jaar bestaat en zichzelf bewezen heeft, en niet dat de software 'verouderd' is.

Na deze lijst van 8 DBMS's te hebben opgesteld, zijn er in overleg met dhr Teel 32 vragen uitgekomen die van belang zijn voor Spill Productions met betrekking tot de wensen/eisen voor hun toekomstige DBMS. Deze lijst vragen is deels gebaseerd op de hoofdvragen die de methode van Indora gebruikt, alsmede enkele interviews met de begeleider. Zie hoofdstuk 3.3 in dit verslag voor deze 12 vragen.

De rest van de vragen is opgebouwd door het extended ISO model te nemen en van elke categorie een aantal geschikte vragen te stellen. Meer hierover verderop in dit hoofdstuk.

Om een duidelijk beeld te krijgen van wat de 'beste' DBMS's zijn aan de hand van de longlist, is er een puntensysteem geïntroduceerd. In de methode van Indora wordt aangeraden om *knockout-criteria* te gebruiken. Ik heb niet gekozen voor een lange vragenlijst en enkele knockout criteria. Ik heb juist gekozen voor een lijst van vragen die allemaal enigszins van belang zijn voor Spill Productions. Aan de hand van deze vragenlijst is een puntensysteem gekoppeld waarin per vraag wordt aangegeven hoeveel een DBMS eraan voldoet.

Per vraag kan er een cijfer gegeven worden van 0 tot en met 3, waarbij 0 slecht is, en 3 uitstekend. De mate waarin een vraag positief beantwoord kan worden geeft ook gelijk de score aan, bijvoorbeeld:

```
1.7  Open source license
     Does the package has an open source license, is it free to copy
     and use on multiple servers?

     Oracle: 0
     MySQL:  3
```

MySQL scoort in dit voorbeeld 3 punten, waarbij Oracle 0 punten scoort. De hoogste score die haalbaar is, is 96 punten. Omdat deze score onmogelijk te halen is voor een DBMS (er bestaat immers geen perfect systeem). Is de lat gelegd op 80 punten of hoger voor een uitstekende DBMS, 70 tot en met 79 voor een goede DBMS, 60 tot en met 69 voor een middelmatige DBMS, en 59 of lager voor een slechte DBMS. Ik ben tot deze keuze gekomen na het invullen van de selectie matrix, waarbij de beste 85 punten had, en de slechtste 54 punten kreeg. Een mooie tussenweg hiertussen is 80 of meer voor een uitstekend systeem, en 59 of minder voor een slecht systeem.

De uiteindelijke selectiematrix is te vinden op in tabel 7 op pagina 39.

De keuze om een onderwerp een 0, 1, 2 of 3 score te geven is als volgt samengesteld.

0 punten:
- Wanneer een DBMS iets totaal niet ondersteund, of afwezigheid hiervan, dan wordt een score van 0 gegeven.

1 punt:
- Wanneer een DBMS zeer slechte ondersteuning voor iets heeft, bijvoorbeeld omdat het als beta geïmplementeerd is, of niet naar behoren werkt vanwege bugs, missende documentatie, dan krijgt het 1 punt.

2 punten:
- Wanneer een DBMS het wel ondersteunt, maar niet optimaal of dat het niet helemaal naar behoren werkt vanwege (kleine) bugs of incomplete documentatie, dan krijgt deze 2 punten toegewezen.

3 punten:
- Wanneer een DBMS de optie goed ondersteund krijgt deze 3 punten toegewezen.

De vragen van de selectiematrix op de volgende pagina zijn te herleiden uit het extended ISO model zoals te zien is in figuur 5 hieronder.

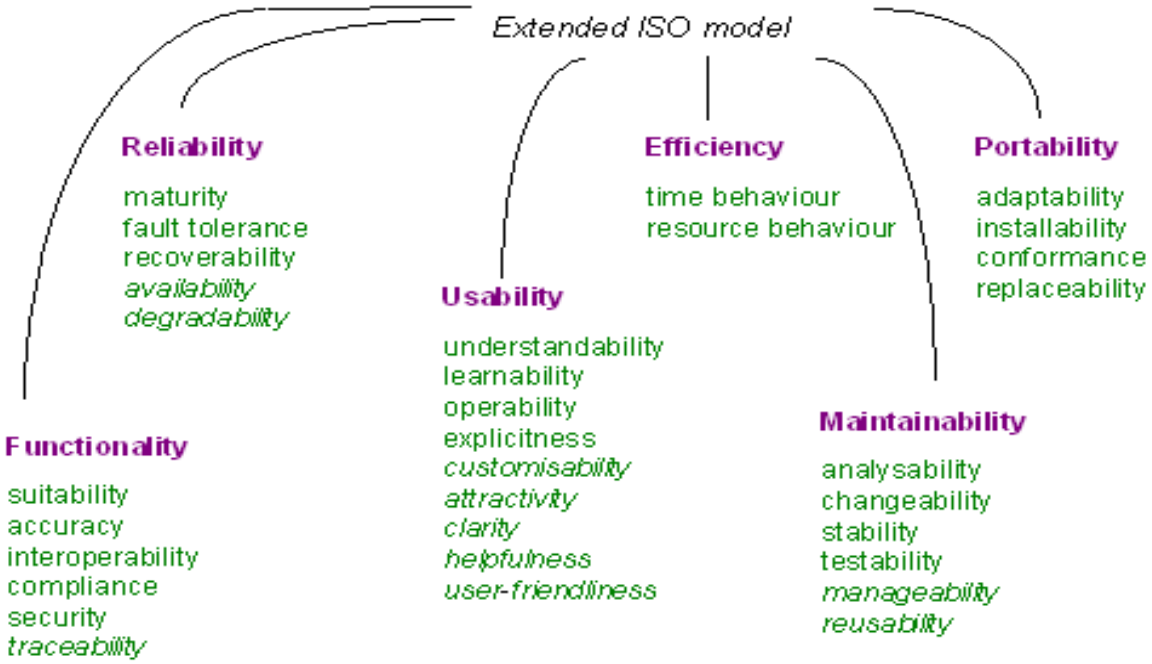


Fig 5: Extended ISO model

Om belangrijke en relevante elementen voor het software pakket in de selectiematrix te kunnen brengen, is uitgegaan van de kwaliteitseisen die door ISO zijn gedefinieerd (zie figuur 4 en figuur 5). ISO schrijft voor elk attribuut een protocol op ter bepaling van het kwaliteitsattribuut. Voor DBMS pakketten is, zonder installatie, nagenoeg niet na te gaan welke kwaliteitsattributen van toepassing zijn. Aangezien 8 DBMS'en installeren en dan de kwaliteitsattributen na te gaan, erg tijdrovend is waardoor het einddoel, namelijk een geschikt DBMS pakket vinden, in het geding zou komen.

Op de website van de leveranciers/ontwikkelaars van deze DBMS pakketten is vaak wel een lijst met functionele eigenschappen van het pakket te vinden. Daarin staan vaak features die overeenkomen met de door ISO opgestelde kwaliteitsattributen. Via forums en websites kon op deze manier ook feedback verzameld worden van personen die deze software pakketten al draaien.

Er is gekozen om de meeste attributen uit ISO over te nemen in de selectiematrix naast de specifieke wensen en eisen van Spill Productions. Hieronder het overzicht, de vraag nummers zijn terug te vinden in de matrix op de volgende pagina:

Reliability: vragen 1.6, 1.8, 2.3, 2.4, 2.5, 2.6, 5.1, 5.2
Useability: vragen 1.4, 1.9, 1.10, 1.13, 2.2, 2.7, 2.8, 3.1, 3.2, 3.3, 3.4
Efficiency: vragen 1.11
Portability: vragen 1.1, 1.5, 1.7, 1.12
Functionality: vragen 1.2, 1.3, 4.1, 4.2, 4.3
Maintainability: vragen 1.8, 1.13, 2.1, 2.2, 2.9, 3.5

Op deze manier is van elke vraag te herleiden dat deze gebaseerd is op het extended ISO model.

	0 = bad	1 = poor	2 = good	3 = excellent				
Score criteria:	50	60	70	80				
Longlist selection matrix								
Package	Oracle	DB2	SQLAnywhere	MS SQL	MySQL	PostgreSQL	Firebird	Ingres
1.0 Base criteria								
1.1 Usage of open standards	3	2	3	1	3	3	3	3
1.2 works on Linux	3	3	3	0	3	3	3	3
1.3 Basic security present	3	3	3	3	3	3	3	3
1.4 Userfriendly maintenance tools	3	3	2	3	3	3	3	3
1.5 Scalability options	3	1	2	1	3	2	3	2
1.6 Software reliability	3	2	3	2	3	3	3	3
1.7 Open Source license	0	0	0	0	3	3	3	3
1.8 Stability	3	3	3	2	3	3	3	3
1.9 Customizable software	1	0	0	0	3	3	3	3
1.10 Reusability	1	1	0	0	3	3	3	3
1.11 Hardware efficiency	3	3	3	1	2	3	2	3
1.12 Flexibility	2	0	3	1	3	3	3	3
1.13 available source code	0	0	0	0	3	3	3	3
2.0 Other								
2.1 Multilanguage	3	3	1	3	3	2	3	1
2.2 Documentation	3	3	2	2	3	2	3	3
2.3 DBMS maturity	3	3	2	3	2	3	1	3
2.4 Popularity	3	2	2	2	3	3	3	1
2.5 Success stories	3	2	3	2	3	3	2	0
2.6 Good reputation	3	3	2	1	3	3	2	2
2.7 DBMS useable without prior training	2	1	3	3	2	2	2	3
2.8 Available trainingprograms	3	3	3	3	3	0	0	2
2.9 Upgrading possiblity	3	3	3	3	3	3	3	3
3.0 Support								
3.1 Support available	3	3	2	3	3	1	1	2
3.2 Forum support available	2	2	1	2	3	3	3	2
3.3 on site support	3	2	0	0	0	3	0	3
3.4 1 hour telephone response	3	3	1	0	3	3	0	2
3.5 consultancy	3	3	1	2	3	3	1	3
4.0 Liability								
4.1 Supplier legal liable	0	0	0	0	0	0	0	0
4.2 SLA possibilities	3	3	1	2	2	0	0	2
4.3 linux indemnification	3	0	0	0	3	0	0	0
5.0 Spill Productions special requests								
5.1 Replication/backup automation	3	2	2	1	2	2	2	2
5.2 Scaleability / cluster	3	2	0	0	3	1	2	2
Package	Oracle	DB2	SQLAnywhere	MS SQL	MySQL	PostgreSQL	Firebird	Ingres
Score	80	64	54	46	85	75	66	74

tabel 7: longlist selectie matrix

6.3 Conclusie

Aan de hand van de selectie matrix, komen 3 DBMS'en naar voren met de meeste punten. Namelijk PostgreSQL, MySQL en Oracle.

Mijn keuze voor de shortlist valt op MySQL en PostgreSQL. De keuze om Oracle niet mee te nemen ligt voornamelijk op de hoge licentie kosten van Oracle. Er was wel aangegeven dat de financiële zijde van het pakket selectie onderzoek niet al te relevant is, maar de kosten van Oracle licenties zijn dermate hoog dat deze niet meer opweegt tegen eventuele performance winst boven open source varianten. In mijn optiek is Spill Productions nog niet zo groot dat dit wel verschil zal maken.

Een voorbeeld van de kosten van een Oracle licentie is recentelijk gegeven door een gezamenlijk test van Sun en PostgreSQL te vergelijken met een test van HP en Oracle. De kosten voor de hardware waren vrij gelijk (de hardware setup van Sun/PostgreSQL was $65.500 dollar en die HP/Oracle was ongeveer $74.000, zie [1] voor meer informatie). Daarnaast moest er voor de Oracle licenties $110.000 betaald worden. Deze HP/Oracle setup had wel 15% meer performance winst vergeleken met de Sun/PostgreSQL setup, maar was ook bijna 200% duurder[2]. Tevens had de Oracle setup ook iets betere hardware.

Ook het feit dat er veel opensource applicaties worden gebruikt bij Spill Productions heeft mijn keuze beïnvloed om Oracle af te laten vallen bij het shortlist gedeelte van het pakket selectie onderzoek.

De shortlist zal gemaakt worden door dieper in te gaan op de DBMS'en PostgreSQL en MySQL. Hierbij wordt eerst in het kort de geschiedenis van het DBMS beschreven, daarna wordt ingegaan over hoe het DBMS met scalability omgaat, en hoe dit op te zetten is. Hierna worden meerdere opstelling gemaakt en wordt de configuratie van deze opstellingen beschreven.

Benchmark tests voor deze configuraties zijn in de shortlist fase nog niet relevant om uit te voeren omdat deze installaties zich bevinden in een vmware omgeving op een desktop computer met hardware die niet in de buurt komt van de daadwerkelijke hardware waarop de DBMS zal gaan draaien.

Na deze resultaten met de opdrachtgever overlegd te hebben, zal beslists worden met welke DBMS een prototype rapport opgezet zal worden.

[1] http://blogs.ittoolbox.com/database/soup/archives/postgresql-publishes-first-real-benchmark-17470
[2] http://blogs.sun.com/tomdaly/date/20070618

7.0 Pakketselectie - Shortlist

7.1 Inleiding

Dit hoofdstuk beschrijft het shortlist gedeelte van het pakketselectie traject. Ik beschrijf per DBMS wat naar mijn mening de meest optimale DBMS configuratie is voor Spill productions. Er wordt nog geen conclusie getrokken nadat een DBMS configuratie is beschreven. De conclusies komen aan bod in het hoofdstuk “Advies”. De keuze waarom ik PostgreSQL en MySQL DBMS’en beschrijf is al gegeven in de conclusie van de longlist.

7.2 Keuze 1: MySQL

7.2.1 Inleiding

Dit hoofdstuk beschrijft MySQL. Hierin leg ik mijn keuzes uit voor een testopstelling voor Spill Productions. Ik ga hieronder mijn keuzes beschrijven van wat ik denk dat een ideale opstelling zou zijn voor Spill Productions met MySQL.

MySQL Community versus MySQL Enterprise

Ik heb een klein gedeelte besteed aan de verschillen tussen MySQL Community release en de MySQL Enterprise release. Dit hoofdstuk is te vinden in de bijlage “Package Selection Report” hoofdstuk “Package selection choice 1 – MySQL”. Ik geef hierin de verschillen weer tussen de 2 versies. De verschillen komen voornamelijk neer op de ondersteuning die MySQL AB biedt aan haar Enterprise klanten. De 2 versies zijn technisch gelijk. Aangezien MySQL sinds eind 2006 pas een Enterprise versie heeft vrijgegeven, staat dit nog in zijn kinderschoenen, en is er niet te garanderen dat er in de toekomst wel technische verschillen tussen deze 2 versies zullen komen. MySQL AB geeft hier zelf geen duidelijkheid over.

7.2.2 Scalability

Naast de 2 soorten scalability, namelijk vertical en horizontal scaling heeft MySQL zelf ook 2 types die gebruikt kunnen worden voor scalability en backup. Deze 2 types zijn replicatie en clustering. Replicatie kan ook weer onderverdeeld worden in 2 gedeeltes, namelijk database replicatie en file replicatie. Database replicatie komt voor in een master / slave setup, waarbij de data op de master wordt gerepliceerd naar de slave. Bij file replicatie komt DRBD om de hoek kijken, waarbij een filesysteem (een partitie meestal) wordt gerepliceerd naar een andere server. DRBD staat dan ook voor *Distributed Replicated Block Device,* en kan vergeleken worden met een RAID 1 configuratie over een netwerk in plaats van lokaal in 1 computer.

Requirements	MySQL Replication	MySQL Replication + Heartbeat	MySQL, Heartbeat + DRBD	MySQL Cluster
Availability				
Automated IP Fail Over	No	Yes	Yes	No
Automated DB Fail Over	No	No	Yes	Yes
Typical Fail Over Time	Varies	Varies	< 30 secs	< 3 secs
Auto Resynch of Data	No	No	Yes	Yes
Geographic Redundancy	Yes	Yes	MySQL Replication	MySQL Replication
Scalability				
Built-in Load Balancing	MySQL Replication	MySQL Replication	MySQL Replication	Yes
Read Intensive	Yes	Yes	MySQL Replication	Yes
Write Intensive	No	No	Possible	Yes
# of Nodes per Cluster	Master/Slave(s)	Master/Slave(s)	Active/Passive	255
# of Slaves	Dozens for Reads	Dozens for Reads	Dozens for Reads	Dozens for Reads

Tabel 8: Een overzicht van availability en scalability in MySQL. [1]

DRBD (Distributed Replicated Block Device) is een externe oplossing, en heeft niets te maken met het bedrijf MySQL AB. DRBD creëert een virtueel Block device dat verbonden is met het onderliggende fysieke device (meestal een harddisk partitie). Deze partitie wordt op deze manier gerepliceerd naar een andere server. Op deze manier is het exact vergelijkbaar met RAID 1.

Omdat het Block device, en niet de data wordt gerepliceerd is de data meer betrouwbaar op de machine waarnaar het gerepliceerd wordt. DRBD zorgt er ook voor data integriteit door alleen een schrijf operatie goed te keuren als deze ook goed is doorgevoerd op de machine waarnaar gerepliceerd wordt.

Heartbeat is ook een externe oplossing en staat los van MySQL AB. Het is geen data replicatie programma, maar let op de servers welke nog up zijn, en zorgt ervoor dat een actieve MySQL DRBD oplossing automatisch zal overschakelen naar de secondary in geval van een calamiteit. Heartbeat dient in combinatie met MySQL Replicatie en DRBD geïnstalleerd te worden om automatische failover te krijgen.

[1] MySQL and DRBD High Availability Architectures whitepaper

MySQL Clustering heeft als voordeel dat het intern al aan loadbalancing doet voor read en write requests, terwijl replicatie dit niet doet. Het is wel in te stellen voor replicatie maar dan werkt het alleen bij read requests. Men heeft dan bijvoorbeeld een master en meerdere slaves, en de read requests worden dan naar een slave gestuurd, en de write requests naar de master, die dit dan weer propageert naar de slaves. Een groot nadeel van een master met een aantal read slaves is dat elke applicatie bijgewerkt/aangepast moet worden om zijn read requests van een andere server te halen dan waar de write requests naar gestuurd worden.

Zoals al eerder genoemd is high availability een ontzettend belangrijk issue voor Spill productions, en clustering kan daar goed op in spelen.

Ik zal voordat ik de gekozen configuraties beschrijf, eerst de voor en nadelen van de 3 methodes (clustering, replicatie, DRBD) beschrijven, alsmede wat de ideale omstandigheden per methode zijn.

MySQL Replicatie zorgt ervoor dat statements en data van een MySQL server naar een andere MySQL gerepliceerd wordt. Zonder gebruik te maken van een complexe setup kan data van een master gerepliceerd worden naar meerdere slaves. De replicatie verloopt asynchroon en er is geen garantie dat data van de master gerepliceerd is naar een slave.

Voordelen

- MySQL replicatie is aanwezig op alle platformen waarop MySQL kan draaien en is niet operating system specifiek. Replicatie tussen een Linux master en microsoft slaves is zonder problemen direct mogelijk.
- Replicatie is asynchroon en kan daardoor gestart en gestopt worden wanneer men maar wil. Dit heeft als voordeel dat replicatie ook over langzamere connecties kan plaatsvinden.
- Data kan gerepliceerd worden naar een onbeperkt aantal slaves, zodat deze opstelling ideaal is in een situatie waarin er veel database reads zijn, maar weinig database writes. Hierdoor kan de read load verdeelt worden over meerdere slaves.

Nadelen

- Data kan alleen naar de master geschreven worden. Complexere setups waarbij masters naar elkaar repliceren zijn mogelijk in een ring configuratie. Zie bijlage “Package Selection Report” hoofdstuk “Replication”.
- Er is geen garantie dat de data tussen de master en de slaves consistent is op een willekeurig moment. Dit komt omdat replicatie asynchroon is, en daarom soms wacht bij kleine updates om deze door te voeren bij de slaves. Dit kan problemen geven bij applicaties waarbij data direct leesbaar moet zijn op slaves.

Bij voorkeur gebruikt bij

- Scale out oplossing waarbij een groot aantal reads plaatsvinden en weinig writes.
- Loggen van data, data analyses van live data. Doordat de data wordt gerepliceerd naar een slave kan er op de slave allerlei queries uitgevoerd worden op deze data zonder de master te storen in het huidige gebruik.
- Online backups.
- Offline backups.

MySQL Cluster is een synchrone oplossing waarbij meerdere MySQL processen database informatie delen. Waarbij het niet mogelijk is bij replicatie om naar elk proces data te schrijven, kan bij een cluster naar elke node data geschreven worden.

Voordelen

- Biedt meerdere read en write nodes voor data storage
- Zorgt voor automatische failover tussen nodes.
- De data op een node wordt direct gedistribueerd tussen de andere data nodes.

Nadelen

- Alleen beschikbaar op een aantal platformen
- Nodes in een cluster kunnen alleen via een LAN verbinding met elkaar gekoppeld zijn. Langere verbindingen en dus tragere verbindingen worden niet ondersteund.

Bij voorkeur gebruikt bij

- Applicaties die een hoge mate van high availability nodig hebben, zoals telecom bedrijven en banken.
- Applicaties die een even hoog aantal writes als reads nodig hebben.

DRBD (Distributed Replicated Block Device) kan bij elke DBMS oplossing gebruikt worden, omdat dit op Operating Systeem niveau werkt, en betrekking heeft tot data op de harde schijf, ongeacht welk programma deze data wegschrijft.

Voordelen

- Zorgt voor high availability en data integriteit tussen 2 servers tijdens hardware falen.
- Zorgt voor data integriteit door schrijf consistentie te garanderen door een write pas goed te keuren als deze ook gelukt is op de backup machine.

Nadelen

- Backup machines kunnen geen gebruik maken zolang data nog wordt gerepliceerd, en daardoor kan bijvoorbeeld een MySQL secondary node niet actief zijn op de backup machine.
- Ondersteunt geen horizontale scaling, omdat backup machines geen toegang hebben tot de data. Verticale scaling wordt wel ondersteund in dit geval.

Bij voorkeur gebruikt bij

- High availability situaties waarbij directe toegang tot huidige data niet nodig is, maar waarbij toegang tot actuele data wel nodig mocht er een hardware failure of een andere calamiteit zich voordoen.

7.2.3 Opstelling

Als testopstelling heb ik gebruik gemaakt van VMware 5.5. Omdat het al eerder duidelijk werd dat een cluster opstelling voor de hoogste uptime zou kunnen zorgen, is dat de gekozen configuratie geworden. Ik ben begonnen met het installeren van 1 template Centos 5.5 Linux besturingssysteem in VMware. Deze VMware opstelling bestaat uit:

- 128 mb RAM
- Harddisk (4 gigabyte)
- CD-Rom met een gemounte Centos iso
- 2 ethernet controllers (om het netwerk tussen de vmware machines gescheiden te houden van de connectie die de vmware machine met de host machine heeft).
- 1 virtuele processor

Dit zijn niet echt configuraties die bij een moderne server horen, maar op het moment dat ik deze test versie maakte, had Spill productions geen servers aanwezig om te testen. Hierdoor komt geen representatieve opstelling naar voren met betrekking tot benchmarking, en hoeveel transactions de cluster aankan. Wat deze test wel aantoont is een werkend configuratie, waarbij getest wordt wat er met de cluster gebeurt wanneer nodes uitvallen en toegevoegd worden, alsmede hoe de cluster omgaat met verschillende configuraties. Voor benchmarking tests kan [1] bekeken worden

De Centos installatie behelst een standaard installatie zonder enige extra's dan de default instellingen.

Deze machine werd 4x gekopieerd om een MySQL cluster te vormen waarbij gekozen kan worden tussen de verschillende replication node groups in de cluster

De gekozen MySQL versie is 5.1.19 geworden, en niet 6.0, omdat deze laatste nog teveel in beta niveau bevindt om een serieuze kandidaat te zijn voor een productie omgeving.

[1] www.ipa.go.jp/software/open/forum/north_asia/download/3-051dbmsc-7_e.pdf

Installatiestappen
De drie type pakketten die geïnstalleerd worden hebben de volgende rollen:

- MySQL(api) Node
  - Een client die data uit de database wil halen maakt verbinding met de api node.
- Data store Node
  - De data nodes zijn de database, alle databases en schemas worden hierin opgeslagen, en deze communiceren weer met de apin odes.
- Management Node
  - De management node wordt gebruikt voor het beheer van de cluster alsmede de configuratie van de cluster.

Ik heb gekozen om alle drie de pakketten op alle servers te installeren. In dit geval kan een server namelijk elke rol vervullen en is het makkelijker om verschillende types configuraties te testen.

Zie onderstaand figuur voor een schematisch overzicht van een typische MySQL cluster. Dit zijn tevens de minimale benodigheden voor een MySQL cluster.

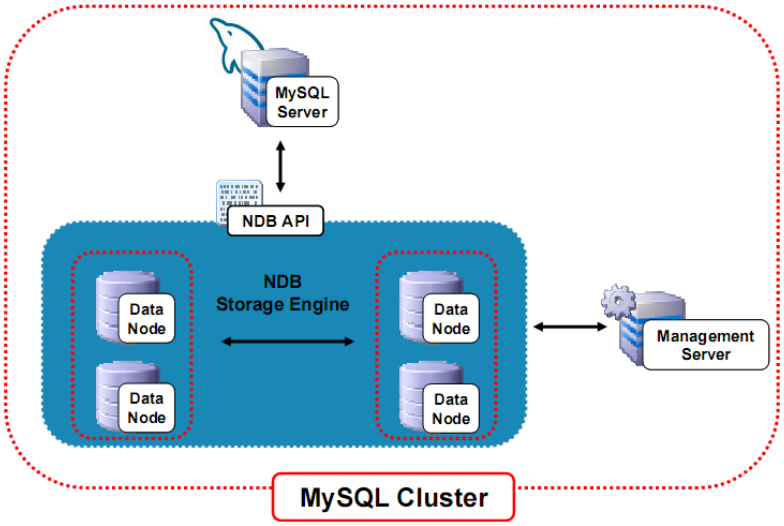


Fig 13: Een minimale MySQL cluster met NDB

Figuur 13 maakt gebruik van 2 nodegroups met 2 storage nodes in elke. Dit is niet de cluster configuratie die ik gekozen heb voor Spill Productions, omdat hier gebruik wordt gemaakt van een level 2 node replicatie. Bij het voorbeeld op de vorige pagina, zullen alleen de data nodes die bij elkaar in dezelfde groep zitten elkaar repliceren. Mocht nu door een calamiteit beide data node servers van 1 groep uitgeschakeld zijn, zal de cluster ook offline zijn. Dit is natuurlijk geen gewenste situatie. Ik kies er dan ook voor om level 4 node replicatie te gebruiken, die alle 4 de data node servers in dezelfde node group zal plaatsen. Hierdoor gaan de 4 data node servers onderling elkaar repliceren omdat ze in dezelfde groep zitten. Zie fig 14 hieronder.

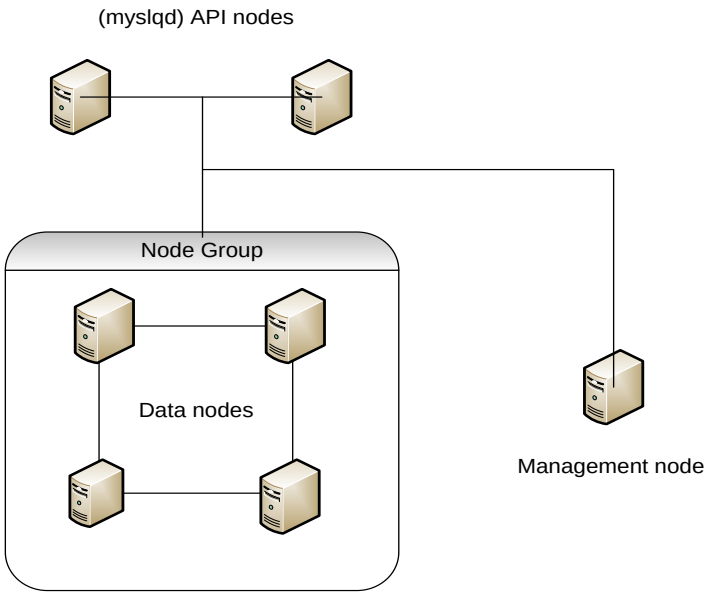


Fig 14: gewenste Spill productions cluster configuratie

Configuratie van de cluster
De cluster moet nu geconfigureerd worden, en we het is noodzakelijk om de management node als eerst te configureren, aangezien deze bij het opstarten van de cluster gebruikt wordt om te kijken welke andere servers bij de cluster horen, en welke replication node group gebruikt gaat worden.

De configuratie van de vorige pagina wordt uitgebreid behandeld in de bijlage “Package selection report” hoofdstuk “MySQL – Clustering”. Ik geef in dit gedeelte ook meer uitleg over de configuratie opties die mogelijk zijn maar die ik niet gebruikt heb.

7.2.4 Conclusie

Ik heb hierboven alleen de MySQL cluster variant beschreven, in de bijlagen is ook een multi master mysql configuratie beschreven. Dit is gedaan omdat daar vraag naar was vanuit Spill Productions. De keuze blijft voor MySQL Clustering, omdat al snel naar voren kwam dat een Multi Master setup ook een groot aantal extra problemen heeft tijdens operationele werking. Hierbij kan gedacht worden aan problemen met dubbele primary keys wanneer er gebruik gemaakt wordt van auto generating keys, data die verloren gaat op bepaalde momenten als een master uitvalt. Voor meer informatie hierover, zie de bijlage “Package selection report” hoofdstuk “package selection choice 1 – MySQL – Replication”.

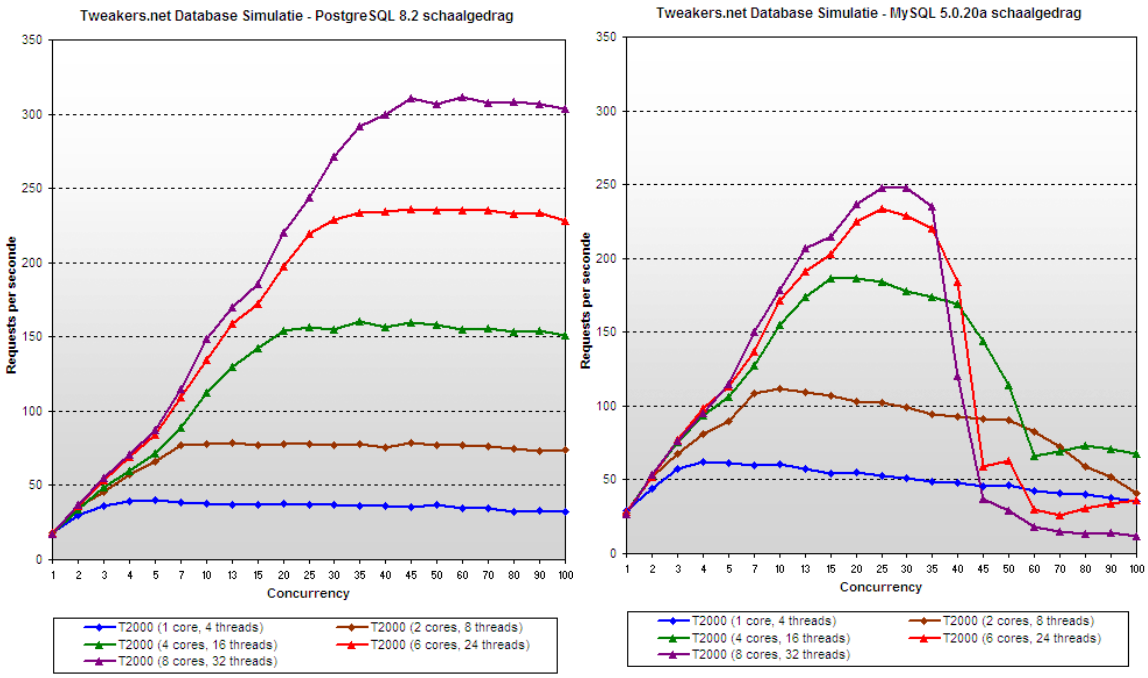
Er wordt in dat hoofdstuk een complexe configuratie beschreven van een circular Multi Master setup. Deze setup heeft een high availability, en is redelijk scalable, maar wordt toch ten zeerste afgeraden. Voornamelijk omdat de scalability gelimiteerd is tot ongeveer 10 servers, en het probleem met duplicate auto generated keys bestaat.

7.3 Keuze 2: PostgreSQL

7.3.1 Inleiding

Het tweede hoofdstuk zal gaan over PostgreSQL. Hierin leg ik mijn keuzes uit voor een voor postgreSQL configuratie Spill Productions. Net als in het vorige hoofdstuk zal ik ook nu nog geen oordeel vellen waarom wel of niet PostgreSQL te gebruiken.

7.3.2 Scalability



Grafiek 16: PostgreSQL en MySQL scaling gedrag.

Bovenstaande grafiek geeft goed het verschil in scaling weer tussen postgreSQL en MySQL op machines met meerdere cores. MySQL gaat er goed mee om totdat er een bepaald punt wordt bereikt waardoor er niet verder gescaled wordt en de performance aanzienlijk wordt verminderd. Meer cores en/of threads toevoegen zal geen enkel positief resultaat meer genereren, 8 cores geven zelfs een slechter resultaat na 50 concurrency (aantal users) dan een single core. Dit is een bekend probleem bij MySQL 5.1 en tot zover is er nog geen oplossing voor gevonden.

PostgreSQL heeft geen native support voor replicatie of clustering, dit wordt gedaan door derde partijen.

Mede door het tweakers.net onderzoek heb ik eerst de vertical scaling opstelling gemaakt met heartbeat en DRBD. Ook is mijn mening, omdat derde partijen de clustering mogelijkheden voor PostgreSQL maken, de ondersteuning en verder ontwikkeling niet gegarandeerd kan worden, en daarom een native oplossing, zoals vertical scaling, beter bij PostgreSQL past.

PostgreSQL heeft 4 types clustering/replicatie pakketten beschikbaar, namelijk:

- PGCluster
    - Multi-master synchrone replicatie.
- Slony-I
    - Master to Multi-slave
- Pgpool
    - Connection pooling frontend met synchrone replicatie
- PostgreSQL table comparator
    - Rsync voor PostgreSQL

Elke van deze pakketten heeft zijn eigen voor en nadelen, hieronder een korte samenvatting van wat elk pakket kan.

PGCluster
PGCluster ondersteunt Multi master configuraties, waardoor aan de eis van high availability en scalability goed voldaan kan worden. Dit komt door het Multi master gedeelte, omdat de servers masters van elkaar zijn en elkaar dus repliceren, is er een verhoogde availability in het geval van een calamiteit.

Omdat het Multi master is, kan er ook makkelijk een server toegevoegd worden, en hier komt het scalability verhaal om de hoek kijken. Een zeer groot nadeel van PGCluster is dat er nauwelijks documentatie beschikbaar is, en geen duidelijk beeld van een roadmap of wanneer nieuwe releases uitkomen. Ondersteuning op forums schiet ook ernstig tekort.

Slony-I

Slony-I is een Master to many slaves replicatie systeem, met ongeveer dezelfde limitaties die standaard MySQL replicatie heeft. Dat wil zeggen, geen automatische transitie van slave naar master als de master faalt, en geen detectie door de master als een slave faalt. Dit is wel weer op te lossen door een combinatie van Slony-I en Heartbeat.

pgpool

Dit pakket is een connection pooling replicatie methode. Deze methode is gelimiteerd tot 2 PostgreSQL servers en heeft geen automatische failover mocht er een server uitvallen. Een voordeel is dat dit pakket automatisch de transacties zal loadbalancen.

PG Comparator

PG Comparator is rsync voor postgreSQL. Deze manier van repliceren gebruikt een speciaal algoritme om dataverkeer voor replicatie tot een minimum te houden. Het gebruikte algoritme is delta compression/encoding[1]

Vergelijking

PGCluster en Slony-I zijn de geavanceerdste pakketten voor PostgreSQL. PGCluster Multi-master mogelijkheden zorgen dat dit pakket goed kan mee scalen met de veranderingen in Spill productions. PGCluster ondersteunt ook snelle recovery tijdens server failures. Slony-I is meer ontworpen om alleen als backup te dienen en ondersteund ook geen 'hot' switch over van master naar slave en vice versa.

PGPool is een simpelere versie van PGCluster/Slony-I, meer ontworpen voor loadbalancing en niet zozeer voor High Availability.

PG comparator is eigenlijk geen replicatie pakket, maar een synchronisatie pakket tussen twee tabellen. Dit pakket wordt vaak gebruikt om een slony-I server weer online te brengen als dat via de gewoonlijke methode niet meer lukt na een calamiteit.

[1] http://en.wikipedia.org/wiki/Delta_encoding

Voor Spillgroup heb ik gekozen voor Slony-I, omdat dit pakket de opties ondersteunde die belangrijk voor Spill Productions zijn, availability en scalability. Theoretisch is PGCluster beter dan Slony-I, maar er is voor PGCluster zeer gebrekkige documentatie te vinden was, een niet up to date website (laatste update stamt uit 2005). Via wat omwegen kon ik wel een recentere versie via de opensource website *sourceforge.net* vinden. Deze versie was wel recent, maar alle vragen omtrent CVS sources waren al maanden niet beantwoord. Dit alles gaf mij een zeer onprofessionele en niet erg betrouwbare indruk van PGCluster, en daarom heb ik voor Slony-I gekozen

Persoonlijk vind ik het wel een minpunt dat PostgreSQL afhankelijk is van derde partijen om replicatie mogelijkheden aan te bieden. Hierdoor wordt door PostgreSQL zelf geen support aangeboden, en blijft het onzeker of de replicatie mogelijkheid die gekozen wordt over bijvoorbeeld een jaar nog wel beschikbaar is.

7.3.3 Opstelling

Ik heb gekozen om 1 gecombineerde opstellingen te maken, namelijk een opstelling van PostgreSQL met Slony-I gecombineerd met Heartbeat en DRBD. De Slony-I opstelling is een master met multi read slaves, en de heartbeat + DRBD opstelling is voor de high availability. Voor de configuratie files van de heartbeat en DRBD opstelling kan gekeken worden in Bijlage I “Package selection report” hoofdstuk “Package selection choice 2 – PostgreSQL”.

De opstelling zijn gemaakt in VMware 5.5 en bestaat uit de volgende configuraties:

- 128 mb RAM
- Harddisk (4 gigabyte)
- CD-Rom met een gemounte Centos iso
- 2 ethernet controllers (om het netwerk tussen de vmware machines gescheiden te houden van de connectie die de vmware machine met de host machine heeft).
- 1 virtuele processor

Er werden 3 van deze vmware machines aangemaakt. Deze kregen de namen Node0, Node1 en Node2.

Node0 en Node1 worden gebruikt als Primary en Secondary voor de Heartbeat/DRBD configuratie, en node2 zal gebruikt worden als connectie test en als client voor node0 en node1.

DRBD setup en het split-brain probleem
De DRBD configuratie is heel recht toe recht aan, waarbij een resource group aangegeven moet worden met daarin de ip' s en partities van de primary en secondary nodes. Ik heb de resource group postgres genoemd om aan te geven dat de configuratie voor PostgreSQL gebruikt gaat worden. Het lastigste gedeelte was om split brain situaties te kunnen opvangen. Ik heb na een aantal tests waarbij ik verschillende split brain situaties simuleerde voor de volgende configuratie in figuur 15 gekozen.

```
net {
       max-buffers 2048;
       ko-count 4;

       after-sb-0pri discard-older-image;
       after-sb-1pri consensus;
       after-sb-2pri call-pri-lost-after-sb;

       rr-conflict call-pri-lost;

}
```

Fig 15: DRBD split brain configuratie

De afkorting in de configuratie file after-sb-0pri staat voor After splitbrain 0 primaries. Oftewel, als er na een split brain geen primary node meer bestaat, maar alleen 2 secondary nodes, dan wordt gekeken naar welke node de meeste recente data heeft, en die wordt dan primary.

Wanneer er na een splitbrain 1 primary node is (after-sb-1pri), heb ik gekozen voor "consensus". Dit houdt in dat er gekeken wordt naar de huidige secondary, en als deze ouder is zoals dan de primary, deze overschreven wordt door de primary. Als de secondary nieuwer is, dan wordt de huidige primary gedisconnect en gaat als standalone verder. Als dit voorkomt is het belangrijk dat een administrator verifieert welke node de meeste recente data heeft en deze handmatig terugzet.

Wanneer er na een splitbrain 2 primary nodes zijn (after-sb-2pri) wordt er net als bij after-sb-0pri gekeken naar welke de meeste recente data heeft. Hierna wordt de server met de minst recente data herstart zodat deze weer 'fris' is en tijdens de boot de rol gesynched zal worden met de overgebleven primary server. Deze variant van split brain komt het meeste voor, voornamelijk wanneer bijvoorbeeld de huidige primary netwerk connectiviteit verliest (bijv wanneer de netwerk kabels eruit getrokken worden). De secondary zal dan herkennen dat de primary niet meer aanwezig is en zelf primary worden. De originele primary node is zich van geen kwaad bewust en zal ook primary blijven. Wanneer netwerk connectiviteit herstelt wordt dan zullen de 2 primary nodes elkaar niet accepteren en wordt de machine met de minst recente data herstart. In bijna alle gevallen zal dit de originele primary node zijn.

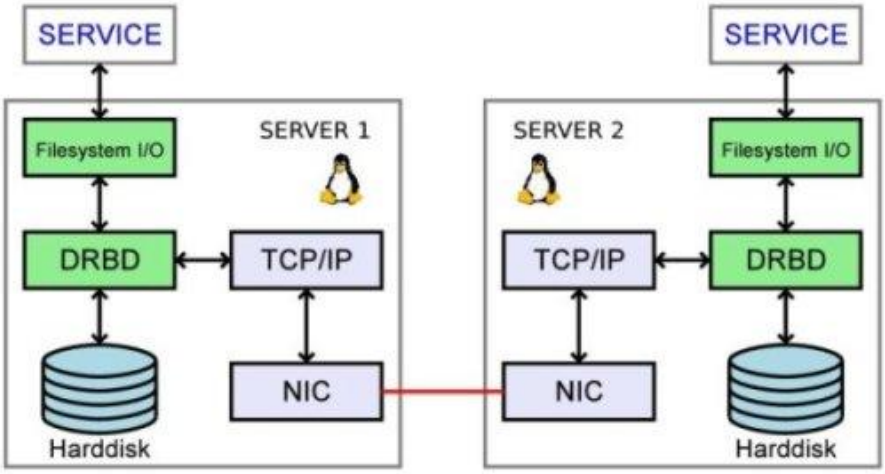


Fig 17: Een schematisch overzicht van een DRBD opstelling

Heartbeat setup
Het programma heartbeat zorgt ervoor dat de IP addressen en DRBD worden omgezet naar een een backup server bij uitval.

De heartbeat setup is ook een recht toe recht aan configuratie zonder bijzondere instellingen. Zie de bijlage “Package selection report” hoofdstuk “Package selection choice 2 – PostgreSQL” voor de gekozen configuratie.

Een setup optie die wel enig denkwerk vereist is de *auto_failback* optie, of die nou on of off moet. Bij off zal er niet automatisch teruggeschakeld worden naar de primary server mocht de secondary detecteren dat deze weer actief is. Bij on wordt dit wel gedaan. Het probleem bij automatisch terugschakelen kan tot dataverlies lijden als er bijvoorbeeld bij DRBD inconsistente data is gevonden. Ook kan het tot het ‘ server flapping’ leiden. Server flapping houdt in dat de primary bijvoorbeeld wel weer up komt, maar door problemen weer binnen enkele minuten offline gaat. Hierdoor moet de secondary server steeds switchen tussen primary en secondary modus. Dit komt niet de availability ten goede omdat het enkele seconden steeds duurt voordat ip’s gewisseld worden tussen servers. Ik heb voor Spill Productions gekozen om *auto_failback* niet aan te zetten en dat een server administrator eerst moet verifiëren of de data goed is op de secondary node en of de ‘oude’ primary node geschikt is om zijn rol weer terug te nemen

Een nadeel hiervan is dat natuurlijk wanneer de secondary ook uitvalt, beide servers offline zijn en de service niet meer aangeboden kan worden aan de clients.

PostgreSQL setup
De PostgreSQL was een standaard setup gecompileerd van de source codes die gedownload zijn van www.postgresql.org. Hierna heb ik gelijk Slony-I geïnstalleerd. Ik heb ervoor gekozen om de configuratie van PostgreSQL zo veel mogelijk te fine tunen om het zo optimaal mogelijk te laten werken met de heartbeat en DRBD combinatie. Zie de bijlage “Package selection report” hoofdstuk “Package selection choice 2 – PostgreSQL” voor de gekozen configuratie.

Slony setup
Ook de slony setup is vrij eenvoudig uit te voeren. Hierna worden 2 test databases aangemaakt op de master server, en wordt via *slonik* (een onderdeel van slony) aangegeven welke databases gerepliceerd moeten worden. Als dit gedaan is wordt *slon* op de master **en** op de slave gestart. Dit zorgt ervoor dat er replicatie synchronisatie plaatsvindt.

7.3.4 Conclusie

PostgreSQL is veruit beter op grotere servers, met name het aantal CPU's en CPU cores maakt een groot verschil tegenover MySQL. Hierdoor is PostgreSQL sterkste punt voornamelijk vertical scaling. 2 PostgreSQL servers met een heartbeat en DRBD setup is volgens mij een uitstekende setup voor Spill Productions. Een ander voordeel van PostgreSQL is dat ze 'echt' SQL-92 compliant zijn. Dit houdt in dat ze zich veel strikter houden aan de originele specificaties zoals deze beschreven zijn in de ISO en ANSI SQL specificaties [1].

[1] http://en.wikipedia.org/wiki/SQL

8.0 Pakketselectie – Conclusie

Ik ging uit van 3 shortlist keuzes, maar door het gebrek aan hardware, en de hoge Oracle licentie kosten, en de huidige applicaties die al op MySQL draaien, heb ik de shortlist keuze ingekort naar 2 keuzes, namelijk MySQL en PostgreSQL.

Mijn keuze gaat uit naar MySQL voor Spill Productions om de volgende redenen:

- Huidige applicaties (zoals openads) draaien al, en zijn gemaakt voor MySQL.
- Goede replicatie en clustering opties.
- Zeer uitgebreide support en advies ondersteuning vanuit MySQL AB.

De reden dat ik niet voor PostgreSQL heb gekozen is om de volgende redenen:

- De configuratie is zeer complex, zelfs voor simpele opstellingen.
- Verschillende applicaties zullen herschreven / veranderd moeten worden om werkend te krijgen onder PostgreSQL.
- Zeer matige ondersteuning voor horizontale replicatie, maar wel zeer goede ondersteuning voor verticale replicatie.
- Replicatie en clustering wordt alleen ondersteund door third parties, in plaats van het PostgreSQL development team. Hierdoor is continuïteit van het project niet gegarandeerd.

Alhoewel PostgreSQL iets sneller is in gelijke hardware configuraties dan MySQL, ben ik van mening dat de boven genoemde nadelen zwaarder wegen dan de voordelen van PostgreSQL. De reden voor de scaling problemen die Spill Productions nu heeft komt meer door het feit dat MySQL niet optimaal is geconfigureerd voor scalability en availability. Mijn suggestie is daarom om door te gaan met MySQL, maar dat de huidige DBMS servers opnieuw worden geconfigureerd om beter om te gaan met scaling en tegelijkertijd high available te blijven. Dit kan gerealiseerd worden door om te schakelen naar MySQL cluster, of om een MySQL Master met multiple read slaves, waarbij de master high available wordt gehouden door middel van heartbeat en DRBD.

De uiteindelijke keus voor MySQL Cluster of MySQL master met meerdere read slaves zal gemaakt moeten worden door Spill Productions. Dit komt mede omdat een cluster configuratie meer machines inneemt in een datacenter. Dit kan een probleem zijn met de huidige problemen die de meeste datacenters momenteel ondervinden wegens te weinig stroom capaciteit. Voor meer informatie zie bijlage VII.

9.0 Evaluatie

9.1 Inleiding

Dit hoofdstuk beschrijf ik de evaluatie van het proces van pakket selectie en implementatie, en daarna de producten die ik gemaakt heb tijdens het pakket selectie proces.

9.2 Evaluatie van het proces

Bij het doorgelopen proces zijn er 5 hoofdfasen te herkennen, namelijk:

- Voorbereiding
- Pakketselectie methode definiëren
- Pakketselectie - longlist
- Pakketselectie - shortlist
- Pakketselectie - conclusie

De *voorbereidingsfase* was een heel belangrijke fase voor het project. Hierin werd namelijk de opdracht gedefinieerd en ook werd hierin het plan van aanpak gemaakt. In deze fase heb ik heb ik via interviews duidelijkheid gekregen wat het project inhield voor de opdrachtgever. Tevens heb ik in deze fase ook doorgevraagd naar de huidige situatie, en wat een gewenste situatie voor de opdrachtgever is. Hierna heb ik deze fase ook gebruikt om informatie in te winnen over de huidige gang van zaken met betrekking tot DBMS' en zoals Spill Productions deze momenteel gebruikt.

Hierna heb ik globaal een te volgen project strategie uitgestippeld die ik de rest van het project heb gevolgd. Hierin heb ik ook de keuzes gemaakt om geen projectbeheersingsmethode toe te passen.

De fase *Pakketselectie methode definiëren* gaat over het kiezen van een pakketselectie methode. Hierin beschrijf ik de verschillende bestaande pakketselectie methodes en geef ik aan hoe ik deze heb toegepast op dit project.

De fase *pakketselectie – longlist* geeft aan welke stappen ik genomen heb om tot een longlist te komen, en beschrijf ik waarom ik deze pakketten gekozen heb. Hierna heb ik een matrix met een puntensysteem, om hierna een overzicht te krijgen welk mogelijkerwijs het beste zou zijn voor Spill Productions.

De fase *pakketselectie – shortlist* gaat dieper in op de twee gekozen pakketten, hierbij worden verschillende configuraties en test opstellingen gemaakt om zo een beter inzicht in het pakket te geven en tevens welke configuratie optimaal is voor Spill Productions. Ik heb ervoor gekozen om voor elk pakket twee verschillende configuraties neer te zetten om zo een duidelijker beeld te geven van de mogelijkheden van het pakket.

In de fase *pakketselectie – conclusie* geef ik mijn uiteindelijke keuze voor het pakket dat mij het meest geschikt lijkt voor Spill Productions. Hierin geef ik ook aan waarom het overgebleven pakket niet geschikt is.

Ik heb me wel verkeken op het opstellen van de test configuraties van de verschillende DBMS' en. Ik had van te voren hier veel minder tijd voor ingepland en hierdoor het configureren onderschat. Omdat ik bij sommige stappen van het pakketselectie traject teveel tijd had ingecalculeerd, hield ik hierbij weer tijd over om te gebruiken bij het opstellen van de verschillende configuraties.

9.3 Evaluatie van het product

Naar aanleiding van het project is er als product een pakketselectie rapport opgeleverd. Tevens zijn er als bijproducten diverse configuraties van diverse DBMS'en gemaakt en gedocumenteerd. Bij het configureren van de verschillende oplossingen ben ik niet tot noemenswaardige problemen gekomen, alhoewel alles wel nieuw was, kon ik er snel aan wennen. De configuratie van DRBD was het lastigst, omdat ik hier tegen het split brain probleem aanliep. MySQL clustering heeft ook last van split brain situaties, maar heeft hier zelf een interne oplossing voor gemaakt.

Bij mijn longlist selectie van 8 DBMS'en heb ik per ongeluk gekozen voor de SQL Anywhere variant van Sybase, en over het hoofd gezien dat Sybase zelf ook een enterprise editie heeft genaamd Adaptive server enterprise. Omdat SQL Anywhere gemaakt wordt door een dochteronderneming van Sybase, genaamd iAnywhere, viel mij deze fout niet op tijdens het pakket selectie traject. Na het tweede gesprek met de docenten wees een van de docenten mij hierop, maar het was al op het einde van het project en daarom te laat om deze fout nog te corrigeren. Dit pakket had tevens de laagste score, maar dat is ook begrijpelijk aangezien het hier ging om een desktop pakket, en niet een pakket dat in server en hosting omgevingen gebruikt moet worden.

Het was een wens van Spill Productions om de documentatie op te leveren in het Engels. Het was even wennen in het begin, maar ik raakte snel gewend aan het typen van engelse documenten. In Spill Productions wordt intern ook veel Engels gesproken en gemaild.