De databasearchitectuur van Panradio
—een zelflerende gedistribueerde muziekspeler—

H.J.W. van Heerde, M.M. Fokkinga
Universiteit Twente, Faculteit EWI
h.j.w.vanheerde@student.utwente.nl

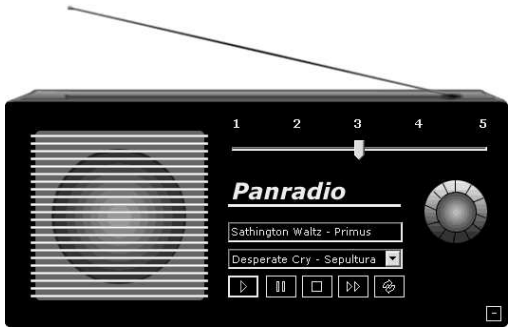
21 januari 2005

1 Inleiding

Muziekspelers bestaan er al heel lang, in allerlei vormen. In de begintijd was het voor de gewone consument alleen mogelijk om door anderen samengestelde en/of gecomponeerde muziek af te spelen. Vooral sinds de intrede van cassettebandjes werd het voor mensen mogelijk om de muziek van hun eigen smaak te beheren en te verzamelen. Het digitale tijdperk heeft dit fenomeen uitgebreid en versterkt. Door middel van een computer met cd-brander werd het voor iedereen mogelijk eigen cd's samen te stellen met een perfecte kwaliteit én precies die muziek die men leuk vindt op één muziekdrager. Kortom, het werd steeds eenvoudiger om cd's samen te stellen precies toegespitst op de smaak van elk individu apart. Bij cd's bleef het niet: de intrede van MP3's in combinatie met breedband internet heeft er voor gezorgd dat iedereen (legaal dan wel illegaal) toegang heeft tot een immense collectie van oude en nieuwe muziek.

Hier komt een probleem om de hoek kijken dat aangemerkt zou kunnen worden als een luxeprobleem: hoe filter ik uit die immense collectie muziek, die al dan niet makkelijk toegankelijk is, precies die muziek die past bij mijn smaak? En daarnaast: hoe leer ik nieuwe muziek kennen als ik niet weet waar ik moet zoeken in de vaak onoverzichtelijke berg informatie die het internet mij te bieden heeft? Een oplossing is natuurlijk om naar de radio te luisteren; er is een groot aanbod aan zenders en minstens één ervan zal toch wel aansluiten op mijn smaak. Maar, dat is nooit precies mijn smaak, en, wat veel hinderlijker is: de muziek wordt verstoord door file-informatie, gebabbel van dj's en reclame.

Zou het niet eenvoudiger zijn als de muziekspeler zélf zou bepalen welke muziek het afspeelt, aangenomen dat die muziek dan ook voldoet aan mijn smaak, zonder dat ik op zoek hoef te gaan naar die muziek maar toch nieuwe artiesten leer kennen? *Panradio* is zo'n systeem. Panradio biedt de gebruiker muziek aan via een simpele applicatie; de grafische interface is afgebeeld in Figuur 1. De gebruiker kan vervolgens kiezen om het nummer af te spelen, of een nieuw nummer kiezen. Ook kan de gebruiker een waardering aan het nummer geven, of (gegeven een lijst met voorstellen) een ander nummer 'aanvragen'.



Figuur 1: *Gebruikersinterface van de Panradio*

Wat houden de eigenschappen 'zelflerend' en 'gedistribueerd' nu precies in? Het zelflerende zit hem in het feit dat de speler het gedrag van de gebruiker analyseert bij het afspelen van muziek. Deze muziek herbergt bepaalde kenmerken zoals artiest en genre. De speler houdt bij welke muziek de gebruiker wel of niet mooi vindt, en zoekt gegeven deze informatie nieuwe nummers die overeenkomen met het smaakprofiel van de gebruiker. Het gedistribueerde zit hem in het feit dat de speler kan kiezen uit een grote collectie muziek en de

*Panradio [1] is uitgevoerd als 'vrij project' in de opleiding Technische Informatica door H.J.W. van Heerde (database- en serveraspecten) samen met R.M. Smelik (zelflerende aspecten), onder begeleiding van H.J.A. op den Akker en M.M. Fokkinga.

daarbij behorende informatie die, transparant voor de gebruiker, beschikbaar is op het internet.

Hoofdstuk 2 gaat in op de gedistribueerde aspecten van het achterliggende 'onzichtbare' systeem van Panradio: de informatievergaring, opslag en verspreiding van informatie, en de distributie van de muziek. Hierbij concentreren we ons in Hoofdstuk 3 op de systeemprestaties: het aantal gebruikers dat het systeem aan kan. Volledigheidshalve lichten we de zelflerende aspecten van Panradio ook nog kort toe in Hoofdstuk 4. Om verwarring te voorkomen gebruiken we vanaf nu het woord *song* in plaats van *nummer*.

2 Architectuur

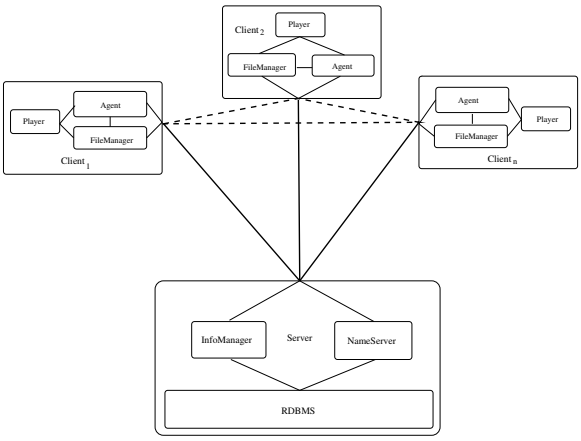
Zoals aangegeven neemt Panradio de gebruiker werk uit handen. Panradio 'beheert' de totale muziekcollectie en kiest daaruit de songs. Zo'n muziekcollectie komt niet zomaar uit de lucht vallen. Deze enorme collectie moet dan ook gevormd worden uit de grote verzameling kleine collecties van de gezamenlijke gebruikers van Panradio. Bekende peer-to-peer netwerken zoals het vroegere Napster en Kazaa, maken al geruime tijd gebruik van dit principe. Bij Panradio is dit niet anders, met één groot verschil: voor de gebruikers van Panradio is dit geheel transparant. Dat wil zeggen, gebruikers hoeven niet te weten dat de afgespeelde muziek van een bepaalde andere gebruiker komt. Althans, hij wordt daar niet constant mee opgezadeld en hoeft niet actief naar die bestanden op andere computers te zoeken.

Om dit doel te bereiken wordt elke gebruiker, en vooral zijn computer, een client van het systeem met locaal zijn eigen muziekcollectie met de daarbij horende meta-informatie. Deze informatie wordt samengevoegd bij de al bestaande informatie van andere gebruikers, waardoor er een grote vergaarbak van informatie óver muziek ontstaat, met daarbij de locaties waar de bijbehorende muziek te vinden is bij de afzonderlijke gebruikers.

2.1 Overzicht van de componenten

Figuur 2 geeft de globale architectuur van het Panradio systeem schematisch weer. Het systeem bevat één centrale server (welke in een later stadium uitgebreid kan worden naar meerdere servers) en een aantal clients. Deze clients bestaan elk weer uit drie componenten:

Agent: zorgt voor het kiezen van songs gebaseerd op een *smaakmodel* van de gebruiker.



Figuur 2: *Architectuur van Panradio. De lijnen tussen de componenten duiden communicatie aan.*

FileManager: zorgt voor het downloaden van de songs die gekozen worden door de Agent. Het downloaden gebeurt via een *peer-to-peer* [3] netwerk, waarbij er contact gezocht wordt met de FileManager van een gebruiker die de gezochte song in zijn locale muziekcollectie heeft. Daarnaast indexeert de FileManager de locale muziekverzameling van de gebruiker.

Player: speelt de gedownloade bestanden af, en geeft een interface voor de gebruikerfeedback.

De clients staan in verbinding met een centrale server, die de informatie over de gezamenlijke totale muziekcollectie bevat. Deze collectie is opgeslagen in een relationele database. We spreken gemakshalve over een 'database van songs', terwijl er in feite in de database alleen informatie *over* songs is opgeslagen; de songs zelf staan bij de gebruikers op de computer. De database wordt beheerd door twee componenten:

InfoManager: accepteert *SongQuery's* van de Agents en retourneert de gevraagde songs uit de database. Een SongQuery is opgebouwd uit *features* van songs (zoals artiest, genre, et cetera). Daarnaast bevat de SongQuery een filter waarin staat welke features de songs *niet* mogen hebben. Ook het aantal songs kan gespecificeerd worden in de query.

NameServer: zorgt ervoor dat de geïndexeerde songs in de database terecht komen. Het heeft hierbij de taak om er voor te zorgen dat er geen *dubbele*

songs in de database komen, songs die een kleine afwijking in de meta-informatie bevatten maar qua inhoud gelijk zijn. Daarnaast kunnen FileManagers locaties van songs opvragen. De NameServer probeert hierbij de meest gunstige locatie van een song terug te geven (songs kunnen meerdere locaties hebben).

2.2 Indexatie van de meta-informatie

Met indexatie wordt in deze context bedoeld het vergaren van informatie over individuele muziekbestanden en over de opslag van deze muziek. Het vergaren van de meta-informatie (informatie over muziek) gebeurt aan de hand van een *content-descriptor*, in de huidige implementatie is dit de *ID3v1-tag* [2]. Het voordeel van het gebruik van deze, primitieve, content-descriptor is de eenvoud: mits goed ingevuld levert het zonder technische rompslomp genoeg informatie op. Deze informatie bestaat onder andere uit genre, artiest, album en jaar van uitgave. Dit zijn tevens ook de features die gebruikt worden om de smaak van de gebruiker te bepalen (waarover in Hoofdstuk 4 meer). Het nadeel van het gebruik van de ID3v1-tags is dat controle op de correctheid van de ingevulde informatie moeilijk is. De tags worden ingevuld door de eigenaren van de MP3's, en kunnen zodoende spelfouten bevatten, of onvolledig of zelfs incorrect zijn. Panradio kan uitgebreid worden met een onbeperkt aantal andere features, waaronder features op een lager niveau die geïndexeerd kunnen worden door analyse van de inhoud van de MP3's.

Indien een nieuwe gebruiker gebruik wenst te maken van Panradio, dan wordt zijn locale muziekcollectie gescand op aanwezigheid van ID3v1-tags. Indien zo'n tag aanwezig is wordt deze ingelezen, waarna de informatie alsmede de fysieke locatie van het bestand opgeslagen wordt in een hashtabel. Deze hashtabel wordt gebruikt om wijzigingen in de locale collectie eenvoudig te detecteren. De geïndexeerde informatie wordt naar een centrale server gestuurd waar de data in een relationele database wordt opgeslagen.

2.3 Toegang tot en opslag van data

Zoals eerder aangegeven is de server verantwoordelijk voor de opslag van, en de toegang tot de geïndexeerde songs. Voor de opslag maakt de server gebruikt van een *relationeel* databasesysteem; op dit moment is dat Postgresql 7.4 [4].

Om op een overzichtelijke manier met data in de database om te gaan (en om op een overzichtelijke manier data in de database te kunnen plaatsen) zijn de relaties in het relationele model afgebeeld op objecten in het object-georiënteerde model. Figuur 3 laat via een ER-diagram een klein deel van de database zien, en hoe de relaties afgebeeld worden op objecten. Zo wordt een abstractielaag gecreëerd tussen serverfunctionaliteit en de gegevensopslag, waardoor het voor het systeem zelf niet van belang is hoe de daadwerkelijke opslag van data geregeld is.

Een groot voordeel van de zojuist beschreven abstractielaag is dat de programmeur zelf niet na hoeft te denken over de te gebruiken SQL query's om informatie uit de database te halen maar methoden zoals `getTitle()` kan aanroepen. Het vergemakkelijkt het programmeren waardoor de kans op fouten kleiner is. Het houdt echter ook in dat de query's minder efficiënt opgesteld zullen worden. Om bijvoorbeeld van een song de titel en de artiest op te halen zou in het relationele model de volgende query uitgevoerd kunnen worden (zie voor het schema Figuur 3):

```
select s.title, p.name
from song s, performs ps, performer p
where s.song_id = ps.song_id
and ps.performer_id = p.performer_id
and s.song_id = ?
```

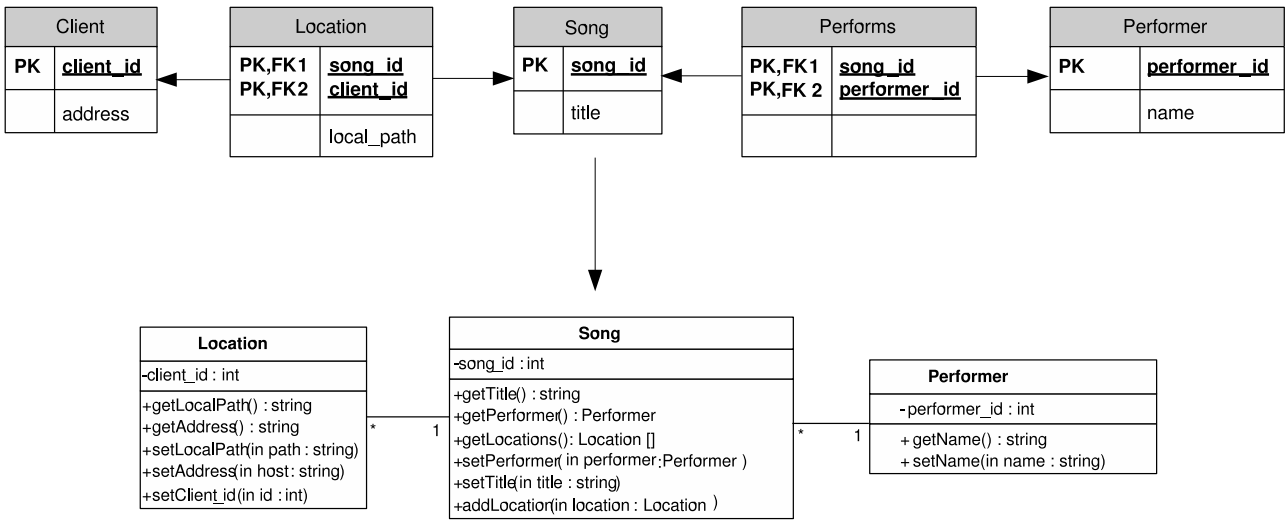
Echter, doordat er ten gevolge van de afbeelding op het object-georiënteerde model gebruik gemaakt wordt van het object `Song`, met twee methoden `getTitle()` en `getPerformer()`, zullen er nu twee SQL query's uitgevoerd worden:

```
select s.title
from song s
where s.song_id = ?
```

en

```
select p.name
from song s, performs ps, performer p
where s.song_id = ps.song_id
and ps.performer_id = p.performer_id
and s.song_id = ?
```

De gekozen aanpak leidt er dus toe dat er meer query's uitgevoerd worden op de database dan nodig is, en dat de optimalisatietechnieken van het DBMS niet ten volle benut worden. Dit kan vooral bij een groter wordende database (in de orde van enkele tienduizenden songs of nog veel meer) tot problemen leiden, wat de schaalbaarheid (één van de eisen van het systeem) niet ten goede komt. Om toch tot acceptabele querytijden te komen is het ontwerp (gerealiseerd in een prototype) uitgebreid



Figuur 3: *Van relationeel naar object-geörienteerd.*

geëvalueerd door middel van tests, en zijn aan de hand daarvan zo goed mogelijk indices op attributen van de verschillende relaties gespecificeerd.

3 Evaluatie van het db-ontwerp

Door een toename van het aantal gebruikers zal de database groeien met als gevolg dat executietijden ook toenemen, tot uiteindelijk een onacceptabele duur. Om te voorspellen hoeveel gebruikers een server aan kan met het huidig ontwerp, is er een model nodig van de prestaties van de server. Uit zo'n model kunnen ook verbeteringen voor het ontwerp afgeleid worden waardoor er méér gebruikers mogelijk zijn.

Dit hoofdstuk laat zien hoe zo'n model is opgesteld door de prestaties van het Panradio-prototype te meten in een aantal tests, en hoe dat model vervolgens gebruikt is. De tests meten de executietijden van het indexeren van nieuwe songs, het opvragen van songs en het opsporen van locaties van songs, als functie van de databasegrootte. De tests zijn niet alleen nodig voor het opstellen van een model, maar ook om te beslissen welke maatregelen (zoals het aanleggen van indices op diverse attributen) daadwerkelijke optimalisaties zijn en welke dat niet niet zijn. Het is zonder experimenten haast onmogelijk te voorspellen wat het effect is van welke maatregelen.

3.1 Tests

De tests zijn allen uitgevoerd op een database draaiend op een server met een AMD Athlon XP 2500+ processor, een 80 GB Samsung HD met 7200 RPM, 256 MB DDR geheugen en een Ashrock moederbord. De tests zijn zoveel mogelijk uitgevoerd op tijden dat de server niet belast wordt door andere processen. Zoals gezegd maakt Panradio gebruik van het Postgresql 7.4 DBMS.

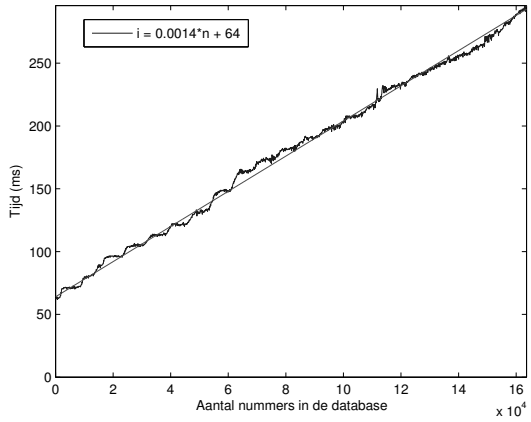
Bij de tests worden er kunstmatige, niet-bestaande *Songs*, *Performers*, *Genres*, *Albums* en *Years* gegenereerd. Hierbij wordt ervoor gezorgd dat er per Performer twintig Songs zijn. Elke Song krijgt één locatie.

Indexatie. Om het indexeren van songs te modelleren zijn er series van 100 songs gegenereerd en naar de server gestuurd. Figuur 4 laat de executietijden zien zoals die behaald zijn na het toepassen van enkele optimalisaties (die dus mede dankzij de tests gevonden zijn). Deze optimalisaties houden met name in het aanleggen van indices (*btrees*); de details hierover staan in het projectverslag [1].

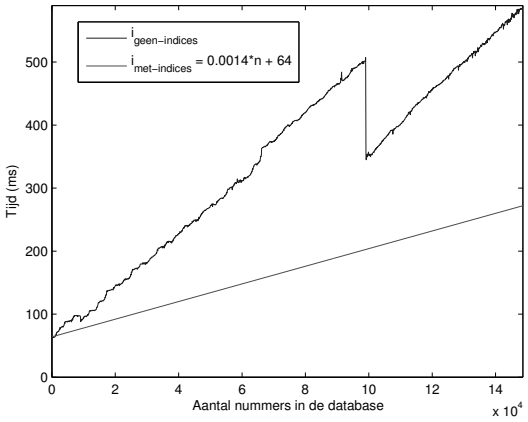
De grafiek laat zien dat de executietijd van indexatie i, gemeten in milliseconden, lineair afhangt van het aantal songs n in de database:

$$i = 0.0014 * n + 64$$

Figuur 5 laat de executietijden van indexatie zien, zónder optimalisaties toegepast op de database. Ter vergelijking is ook de functie i weergegeven. Te zien is hoe



Figuur 4: *Executietijd van indexatie als functie van het aantal songs in de database, nadat optimalisatie is toegepast. De rechte lijn geeft de benadering van deze executietijden weer.*



Figuur 5: *Executietijd van indexatie als functie van het aantal songs in de database, zónder optimalisatie. Duidelijk te zien is de verandering in executietijd op het moment dat het DBMS zelf een nieuw queryplan kiest. De rechte lijn is de benadering van de executietijden mét optimalisatie (uit Figuur 4).*

optimalisatie de executietijden drastisch verkort. Ook is te zien hoe het DBMS zelf probeert een goede query executie strategie te bedenken, en deze aanpast als een bepaalde grens (100 000 tuples) is bereikt.

Het opvragen van songs. Voor het bepalen van de gemiddelde executietijd van het opvragen van een song, worden er verschillende, voor het systeem representatieve, SongQuery's gegenereerd en naar de InfoManager gestuurd. Figuur 6 laat de executietijden bij oplopende aantallen songs in de database zien na optimalisatie. De *gemiddelde* querytijd q hangt als volgt af van het aantal songs n in de database:

$$q = 0.0066 * n + 30$$

Figuur 7 laat zien wat er gebeurt indien er geen optimalisaties worden toegepast: bij 120 000 songs in de database is de executietijd zonder extra indices ongeveer 20 seconden, mét indices minder dan 1 seconde. Bij grotere n wordt de verhouding snel groter: bij 500 000 songs is de executietijd zonder extra indices ongeveer 4.5 minuut, mét indices slechts 1.5 seconde.

Het opvragen van locaties. De test voor het vinden van de functie voor de executietijd van een locatieopvraag gaat analoog aan de hiervoor beschreven tests. Er wordt een vraag naar een locatie van een willekeurig gekozen song naar de NameServer gestuurd. De executietijd l van een locatieopvraag blijkt als volgt af te hangen van het aantal songs n in de database:

$$l = 0.0023 * n + 48$$

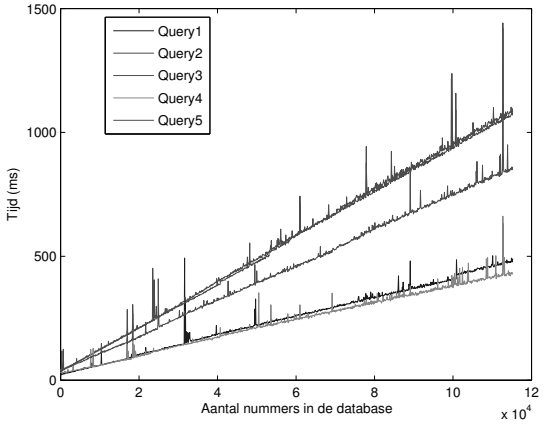
3.2 Model

Uit de tests is gebleken dat met name geschikt gekozen indices de executietijden aanzienlijk verbeteren. Daarbij zijn er goede benaderingen gevonden van de verschillende executietijden als functie van het aantal songs in de database. Het doel is nu om het gedrag van de totale service (InfoManager + NameServer) te modelleren.

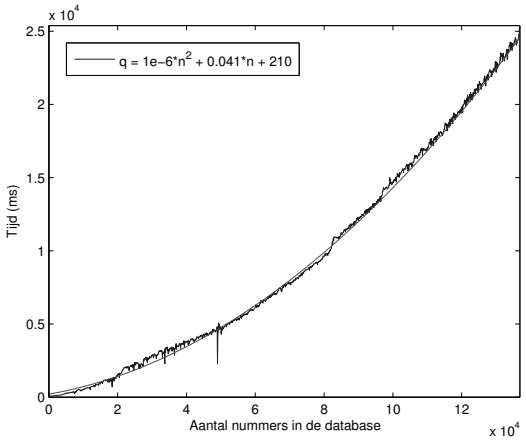
Naast de al genoemde n, i, q en l, worden ook de volgende variabelen onderscheiden:

- g: het aantal *g*ebruikers van Panradio,
- a: het aantal song-*a*anvragen per tijdseenheid,
- v: de totale *v*erwerkingstijd van een aanvraag,
- p: de *p*restatie, i.e., het aantal verwerkte songs per tijdseenheid: $p = 1/v$,
- A: de fractie van de gebruikers die *actief* zijn (alleen zij veroorzaken aanvragen van songs),
- S: de gemiddelde tijdsduur van een *s*ong,
- F: een *f*actor in 0..1 die van de Agents afhangt,
- R: de verhouding (*r*atio) tussen aantallen SongQuery's en locatieopvragingen.

De met een kleine letter geschreven variabelen hangen af van het aantal gebruikers, g. De variabelen zijn als



Figuur 6: *Querytijd als functie van de databasegrootte, voor verschillende type query's, ná optimalisatie.*



Figuur 7: *Querytijd als functie van de databasegrootte, zónder optimalisatie.*

volgt aan elkaar gerelateerd (met milliseconde als tijdeenheid):

- We nemen aan dat elke gebruiker 1000 verschillende songs heeft, en dat deze verschillen van de songs van de andere gebruikers. Dus voor het aantal songs in de database geldt: $n = 1000 * g$.

  Deze aanname is discutabel: het is de vraag of bij grote gebruikersaantallen iedere nieuwe gebruiker wéér 1000 nieuwe songs inbrengt. Maar met het verstrijken van de tijd zullen er wél steeds nieuwe songs op de markt komen en door gebruikers ingebracht worden. Bij gebrek aan beter houden we het op de gedane aanname.

- Het aantal song-aanvragen per tijdseenheid, a, is recht-evenredig met het aantal actieve gebruikers $A * g$ en omgekeerd evenredig met de tijdsduur $F * S$ die een song wordt afgespeeld; we stellen dat gemiddeld genomen een gebruiker een song afbreekt nadat fractie F ervan is afgespeeld. Dus $a = A * g/(F * S)$. Betere Agents zorgen ervoor dat de gespeelde songs meer in de smaak van de gebruiker vallen en de gebruiker dus minder songs afbreekt zodat factor F dichter bij 1 ligt.

- De totale verwerkingstijd van een aanvraag, v, is per definitie de som van de verwerkingstijden van de InfoManager en de verwerkingstijden van de NameServer voor het traject tussen het zoeken naar een geschikte song en het leveren van een locatie voor de song. We nemen ter vereenvoudiging aan dat het zoeken van songs (SongQuery's), het opvragen van locaties en de indexaties niet tegelijkertijd worden gedaan. In werkelijkheid gaat dit deels wel tegelijkertijd. We nemen voorts aan dat na iedere SongQuery en iedere locatieopvraag een indexatie plaats vindt (zodat er volgens deze modellering altijd "tijd genoeg is voor indexaties"). Omdat er per locatieopvraag R SongQuery's zijn, is de totale verwerkingstijd van een songaanvraag, v, gelijk aan: $R * (q + i)$ (voor de verwerking van de SongQuery's plus opvolgende indexaties) plus $l + i$ (voor de locatieopvraag en de erop volgende indexatie).

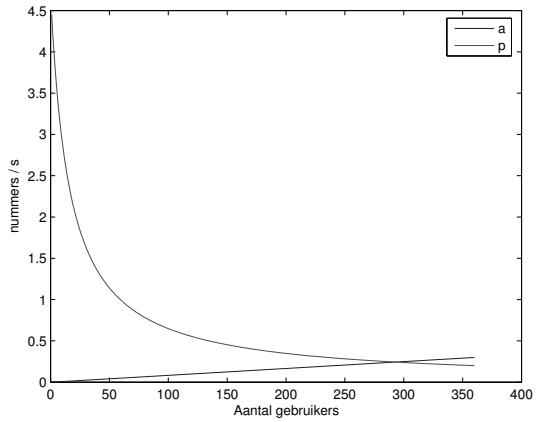
Samengevat leveren deze overwegingen en aannamen, tezamen met de tests, dit model:

$$\begin{aligned} i &= 0.0014 * n + 64 \\ q &= 0.0066 * n + 30 \\ l &= 0.0023 * n + 48 \\ n &= 1000 * g \\ v &= R * q + l + (1 + R) * i \\ a &= A * g \ / \ (F * S) \\ p &= 1/v \end{aligned}$$

Uiteraard kan het model verfijnd worden met meer aspecten en gedetailleerdere aannamen.

3.3 Gebruik van het model

Een belangrijke prestatie-eis aan Panradio is dat een gebruiker nooit hoeft te wachten op een volgende song.



Figuur 8: *De prestatie p en aanvraagfrequentie a als functie van het gebruikersaantal g, bij $A = 20\%$, $S = 4\,\text{min}$, $R = 1.2$, $F = 1$. Tot en met 292 gebruikers (dus $292 * A = 58$ actieve gebruikers) is aan de eis $p > a$ voldaan.*

Dit betekent dat een aanvraag uitgevoerd moet zijn binnen de tijd dat de voorafgaande song afgespeeld is. In termen van het model is hieraan *gemiddeld genomen* voldaan wanneer:

$$p \quad \geq \quad a$$

Figuur 8 laat p en a zien als functie van het aantal gebruikers g, wanneer er 20% gebruikers actief zijn (A), de gemiddelde lengte van een song 4 minuten is (S), en $R = 1.2$ en $F = 1$. Bij gebruikersaantallen tot en met 292 is aan de eis voldaan; van deze 292 gebruikers kunnen er dus 20%, ongeveer 58 stuks, tegelijkertijd actief zijn, zonder dat een van hen moet wachten wanneer een nieuwe song afgespeeld gaat worden. Op dezelfde manier kunnen andere, optimistischer en pessimistischer, scenario's doorgerekend worden. Uit de figuur blijkt ook dat bij grote gebruikersaantallen de grafieken voor p en a "nogal evenwijdig" lopen, en dus dat kleine wijzigingen van sommige parameters grote veranderingen kunnen hebben op het punt waar p en a elkaar snijden: het aantal gebruikers dat het systeem aan kan.

Het model laat zien dat de volgende aspecten van grote invloed zijn op de schaalbaarheid van het Panradio-systeem.

A. De richtingscoëfficiënten van i, q en l geven aan hoe groot de toename in tijd is bij een toename in het aantal songs in de database. Hoe kleiner deze coëfficiënten, hoe kleiner die toenamen en hoe korter de executietijd bij grotere aantallen songs in de database. Een verkleining van de coëfficienten heeft tot gevolg dat er meer gebruikers van het systeem gebruik kunnen maken.

   Bijvoorbeeld, Sectie 2.3 beschrijft dat en hoe query's worden samengesteld. De manier waarop dat gebeurt bepaalt mede de coëfficiënt van q.

B. Indien het aantal songs in de database lager is, dan is de prestatie hoger. Dit suggereert om de database te distribueren over verscheidene servers.

C. De verhouding R tussen aantallen SongQuery's en locatieopvragingen speelt een belangrijke factor in het aantal gebruikers. Hoe meer SongQuery's er per songaanvraag uitgevoerd moeten worden, hoe groter de belasting.

D. Het percentage A van gebruikers dat actief is, is van grote invloed op het aantal gebruikers. Door verschillende waarden voor A te kiezen ($A * g$ is het aantal actieve gebruikers) kunnen verschillende gebruikersscenario's doorgerekend worden.

Punten A en B beïnvloeden de prestatie aan de server zijde. Punt C is, net als factor F, afhankelijk van de Agents: indien de Agents zorgvuldig om gaan met resultaten, zijn er minder query's nodig wat gunstig is voor de belasting van het systeem. Punt D is een statistische waarde die moeilijk beïnvloed kan worden.

Naar aanleiding van punt B is er, om aan de schaalbaarheidseis te voldoen, ook weer met deze modelmatige aanpak een ontwerp gemaakt voor een distributie van de database over meerdere servers. Dit ontwerp is terug te vinden in het projectverslag [1] en wordt hier niet behandeld.

4 Zelflerende aspecten

Het uiteindelijke doel van Panradio is dat de gebruiker muziek voorgeschoteld krijgt die gebaseerd is op de smaak van de gebruiker. Hiervoor is de Agent verantwoordelijk. Om het plaatje van het Panradio-systeem compleet te maken, volgt hier nog een korte beschrijving van de zelflerende aspecten. Zoals aangegeven bij de beschrijving van het model, heeft de mate waarin de Agent zijn werk doet invloed op de belasting van de server.

De Agent speelt de rol van radio-dj. Hij bepaalt welke muziek voorgeschoteld wordt aan de Panradio-gebruiker. De Agent speelt dus een cruciale rol voor de positieve of negatieve ervaring die de gebruiker met Panradio zal hebben. De gebruiker maakt niet direct zijn muziekwensen aan de Agent bekend, de Agent is zelfs volledig verborgen voor de gebruiker. Door het gedrag van de gebruiker te observeren probeert de Agent te achterhalen of zijn gemaakte keuzen geschikt zijn, of dat hij zijn gedrag moet aanpassen. Vooraf trainen is dus niet nodig, de Agent zal naarmate het programma langer gebruikt wordt zijn gedrag verder verfijnen en beter in staat zijn in de keuze voor muziek de daadwerkelijke smaak van de gebruiker te volgen.

Het leermechanisme is geïnspireerd op *Reinforcement Learning* [5] en is goed te vergelijken met *Evaluative Feedback* [6]. Aan de hand van een modellering van de smaak van de gebruiker zal de Agent met behulp van de InfoManager zoeken naar muziek die aansluit op dat model. Door middel van interactie met de gebruiker (deze kan de voorgeschotelde song afbreken of een waardering geven) wordt dit model ververst. Het smaakmodel bevat de elementen Song, Performer, en Genre. Per element is een waardering en het aantal aanpassingen van de waardering van het element opgeslagen. De waardering is een waarde tussen de 0.0 en de 1.0 (0.0 slechtst, 1.0 best). Songs, Performers en Genres die niet in het model voorkomen, hebben standaard de waardering 0.5. De Agent heeft initieel een leeg smaakmodel.

Om ervoor te zorgen dat de Agent ook nieuwe muziek verkent (die dus niet noodzakelijkerwijs hoeft te voldoen aan het smaakmodel), bevat de Agent een *problem generator*. Zonder een actieve problem generator blijft de Agent een klein aantal geschikte songs herhalen, dat wil zeggen hij blijft steken in een locaal minimum. De suggesties kunnen op twee manieren tot stand komen:

- Willekeurige suggesties: het kiezen van een song met als voorwaarde dat het nog niet gespeeld is (de song, de artiest of het genre).
- Suggesties van andere Agents: bij andere Agents met vergelijkbare smaak om suggesties vragen.

Een formele beschrijving van het verversen van het smaakmodel gebaseerd op beloningen staat in het projectverslag [1].

5 Afsluitende opmerkingen

Om het database-ontwerp van het prototype van Panradio te evalueren, en enig houvast te hebben bij verbeteringen aan het ontwerp, is er een model van het ontwerp gewenst. Zo'n model is gemaakt. Het model is zo eenvoudig dat het in de koffiekamer beredeneerd kan worden en op de achterkant van een envelop uiteengezet. Om goede benaderingen te vinden voor sommige executietijden die in het model voorkomen, zijn uitgebreide tests gedaan. Die tests zijn sowieso ook nodig om te bepalen welke maatregelen de prestaties verbeteren. Met name het toepassen van geschikt gekozen indices zorgt ervoor dat de server meer gebruikers aan kan. Het aantal gebruikers is dé maatstaf waaraan de prestatie van het systeem afgemeten wordt.

Er is inmiddels een ontwerp gemaakt voor de distributie van de data over meerdere databases. Verder onderzoek zal moeten uitwijzen of het ontwerp schaalbaar is zodat niet alleen onze weinige huisgenoten, of de vele campusleden, maar ook enorme aantallen internetters gebruik kunnen maken van Panradio!

Referenties

[1] H.J.W. van Heerde, R.M. Smelik, Panradio verslag, 2004, `http://panradio.vanheerde.net/verslag.pdf`

[2] id3.org, ID3 made easy, `http://www.id3.org/id3v1.html`

[3] Wikipedia, peer-to-peer, `http://en.wikipedia.org/wiki/Peer-to-peer`

[4] Postgresql website, `http://www.postgres.org`

[5] Stuart J. Russell, Peter Norvig, Artificial Intelligence, a modern approach, 2nd edition, Pearson Education, Upper Saddle River, New Jersey, 2003.

[6] Richard S. Sutton, Andrew G. Barto, Reinforcement Learning, an introduction, MIT Press, Cambridge (MA), 1998.