KU LEUVEN

FACULTEIT INDUSTRIËLE INGENIEURSWETENSCHAPPEN,
TECHNOLOGIECAMPUS GENT

2017-2018

Distributed UNO

Klik en typ de subtitel

Pieter Vanderhaegen

Wouter Deceuninck

2017-2018

Inhoud

Inleiding

Dit is een verslag van de UNO-opdracht voor het vak gedistribueerde systemen. De opdracht bestaat eruit om een gedistribueerd systeem te creëren dat spelers toelaat om UNO te spelen. De spelers moeten zich eerst registeren bij de server door een login en paswoord in te geven. Eenmaal aangemeld komt de speler in een lobby terecht die het aantal reeds aangemaakte spellen bevat. De client kan dan deelnemen aan bestaand spel of nieuwe spellen aanmaken. De spellen kunnen zowel met 2, 3 als 4 spelers gespeeld worden. Als communicatietechnologie dient gebruik gemaakt te worden van Java RMI. Ook moet rekening gehouden worden met de gevraagde architectuur met databank servers, applicatie servers, dispatcher en clients.

1 Beschrijving van de architectuur

Voor het ontwerp van onze gedistribueerde UNO-applicatie maken we gebruik van een dispatcher. Deze dispatcher heeft als taak om verschillende gebruikers toe te wijzen aan verschillende applicatieservers. Hierbij moet natuurlijk rekening gehouden worden met de belasting van deze verschillende applicatieservers. Om dit te realiseren zal de dispatcher dus bijhouden hoeveel servers er online zijn en ook hoeveel spellen er op elke server worden gespeeld. Wanneer er dan een nieuwe client zicht registreert zal deze terechtkomen bij de dispatcher, die de client doorstuurt naar een geschikte applicatieserver om nadien een spel te kunnen starten.

Naast deze dispatcher en de verschillende applicatieservers maken we ook gebruik van enkele databankservers. Elk van deze databankservers is verbonden met een lokale kopie van de databank en dus verantwoordelijk voor het databankverkeer. Door het gebruik van meerdere databankservers moeten we ook rekening houden met consistentie en replicatie van de gegevens. De verschillende replica's van

De dispatcher zal ook bijhouden welke applicatieservers verbonden zijn met welke databankservers. Zo wordt het ook mogelijk om de servers gelijk te verdelen over de verschillende databanken, waardoor het verkeer per databank verminderd wordt.

Zoals eerder vermeld is recovery mogelijk bij onze applicatie, indien er een error zou plaatsvinden of een server uitvallen, is de dispatcher ook het eerste aanspreekpunt voor de client. De client stuurt bij error dus een request naar de dispatcher, die op zijn beurt zal proberen om de connectie te herstellen of een nieuwe server op te starten of toe te wijzen aan de client.

De applicatieservers worden op de eerste plaats gebruikt voor de authenticatie van nieuwe clients. Daarnaast bevatten ze ook de logica om een UNO-spel te spelen en de game info door te sturen naar andere database servers waarmee mogelijks andere applicatieservers verbonden zijn.

2 Beschrijving van de databank

Voor het opstellen van onze databank hebben we gebruikt gemaakt van SQLite. Dit is een snel en betrouwbaar databasemanagementsysteem dat gebruikt maakt van SQL. De opbouw van onze databank wordt hierna beschreven.

- *Users*

De eerste tabel bevat de nodige info voor het registreren en inloggen van gebruikers. We houden eigenschappen voor identificatie van de gebruikers bij zoals username, de hash waarde van zijn paswoord, een token en een timestamp.

- *Game*

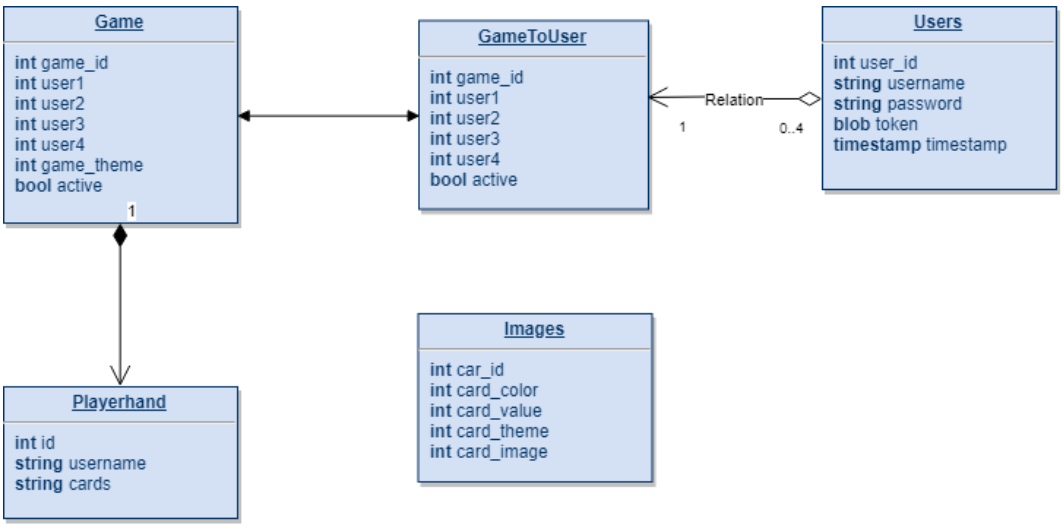
De Game tabel wordt gebruikt om informatie voor het spelen van een spel bij te houden. De tabel bevat info om een bepaald spel op te zoeken, zoals een id en een naam. Om het mogelijk te maken met verschillende soorten kaarten te spelen, houden we ook een card_theme bij. Door deze parameter te veranderen zal het uitzicht van de kaarten tijdens het spel ook veranderen. Ook houden we de ServerPort bij, deze wordt gebruikt om een spel te koppelen aan een bepaalde applicatieserver.

- *GameToUser*

Deze tabel wordt gebruikt om na te gaan welke spellen er actief zijn en welke spelers deelnemen aan deze spellen.

- *PlayerHand*

Per spel houden we ook een tabel bij die de kaarten van elke speler bijhoudt. Zo kunnen we ook een spel op elk moment herstellen omdat we op elk moment kunnen opvragen welke speler welke kaarten heeft.



3 Ontwerpbeslissingen

Tijdens het implementeren van de opdracht werden we geconfronteerd met verschillende ontwerpdilemma's. Hierna bespreken we kort welke keuzes wij gemaakt hebben op de verschillende domeinen.

3.1 Consistency & replication

In onze applicatie worden er 4 databankserver gebruikt, met elk hun eigen lokale databank. Deze databankservers worden opgestart en beheerd door de dispatcher. Bij het toewijzen van applicatieservers aan deze databankservers zal de dispatcher rekening houden met de belasting van elk van deze databankservers. De databankservers met het minst aantal verbonden applicatieservers krijgt dus de volgende nieuwe applicatieserver toegewezen.

Om een consistente gedistribueerde databank te realiseren zorgen we ervoor dat alle write operaties naar lokale databanken worden gepropageerd naar de replica's van andere databankservers. Omdat er voor elk spel een andere tabel bijgehouden wordt, zullen er veel schrijfoperaties plaatsvinden, waardoor we deze dus ook vaak zullen moeten propageren. Maar om dit te versnellen maken we voor query's naar onze databank gebruik van preparedstatements. Omdat de grootte van de tabel beperkt blijft tot 1 spel ondervinden we ook geen vertragingen door de write operaties naar andere replica's.

3.2 Caching

Om de bestanden voor grafische voorstelling van de kaarten niet voortduren te moeten betrekken in de write operaties van het spel, worden deze enkel initieel gedownload bij de client en daar gecached. Zo kunnen deze snel opgevraagd worden zonder dat er eventuele vertragingen door de query's van de databank plaatsvinden. De functionaliteit bevindt zich verder voornamelijk op de server waardoor de client zich vooral beperkt het doorsturen naar en ontvangen van de server.

3.3 Security

Op vlak van security moesten we in de eerste plaats zorgen voor een degelijke authenticatie en autorisatie. Wanneer nieuwe clients zich registeren dienen ze een unieke username en een geheim paswoord in te geven. Dit passwoord wordt eerst gecontroleerd of dit wel lang genoeg is om veilig te zijn. Hier kunnen makkelijk nog andere extra controles toegevoegd worden. Wanneer een gebruiker zich registreert zal zijn ingegeven wachtwoord gehashd worden met behulp Bcrypt, een implementatie van het OpenBSD Blowfish encryptiealgoritme. Bij het inloggen en registeren van gebruikers wordt ook een gesigneerde token meegegeven, deze kan gebruikt worden om de sessie bij te houden van de client.

Door een onderscheid te maken tussen een User object en een Player object is het niet mogelijk dat er betrouwbare informatie gestolen wordt uit je Player object waarmee je deelneemt aan een spel.

3.4 Recovery

Doordat we onze replica's updaten bij alle schrijfoperaties naar een databank van een database server, is het op elk moment mogelijk om een server weer te herstellen naar zijn vorige staat. Hiervoor dienen we de dispatcher aan te spreken, die verbonden is met zowel de applicatieservers als de databankservers.

Client Recovery

Wanneer een spel onderbroken wordt door connectieproblemen en er niet meer verder gespeeld kan worden, zal de client een boodschap ontvangen en zal het spel gestopt worden. Het spel wordt op inactief gezet en het dataverkeer naar de database server wordt gestopt.

Database Recovery

Doordat de verschillende replica's altijd consistent gehouden worden is het geen probleem om van de ene database server over te schakelen op de anderen. Om te voorkomen dat de databaseserver zal uitvallen en het spelverloop te versnellen wordt er een thread gestart voor het propageren van de writes naar de replica's. Op die manier kan het spel verder gespeeld worden terwijl de andere database servers geüpdatet worden.

4 Api’s

Zie JavaDoc op https://github.com/Pieterism/UNO

5 Reflectie

Als laatste kijken we terug op hoe we deze opdracht gerealiseerd hebben. We evalueren de positieve en minder positieve punten van onze implementatie. Ook bedenken we enkele mogelijk uitbreidingen die we mits meer tijd zouden kunnen implementeren.

Sterktes

Een van de positieve punten van onze implementatie is dat het gaat over een autonoom systeem. Het systeem is, eenmaal opgestart, instaat om zichzelf te herstellen. Clients worden bij problemen teruggestuurd naar de lobby om een nieuw spel te starten en databankservers kunnen zonder dataverlies hersteld worden.

Zwaktes

Zelf hadden we graag de consistency van de databankservers verder uitgewerkt. Momenteel wordt reeds alle nodige data bijgehouden, maar een volledige recovery van een applicatieserver is nog niet mogelijk. We zouden dit realiseren door uit de GameToUser-tabel op te vragen welke spellen nog op actief staan en dus niet correct zijn afgesloten. Eenmaal we deze spellen weten, kunnen we via query's in de databank ook de kaarten per speler ophalen.

Mogelijke uitbreidingen

Mogelijk maken om een spel dat onderbroken is door een crash van een client verder te spelen wanneer de client binnen een bepaalde tijdspanne zich terug kan aanmelden bij het systeem. Hiervoor kunnen we gebruik maken van de tokens.

Ook zou het interessant zijn om het mogelijk te maken om tegen een bot te spelen. Waardoor je ook alleen kan spelen op verschillende moeilijkheidsgraden.

Het toevoegen van eigen regels per spel lijkt ons ook een leuke uitdaging.

AFDELING
Straat nr bus 0000
3000 LEUVEN, België
tel. + 32 16 00 00 00
fax + 32 16 00 00 00
@kuleuven.be
www.kuleuven.be