Distributed UNO

Klik en typ de subtitel

Pieter Vanderhaegen

Wouter Deceuninck

Faculteit industriële ingenieurswetenschappen, technologiecampus Gent

2017-2018

2017-2018

Inhoud

[1 Beschrijving van de architectuur 7](#_Toc523353914)

[2 Beschrijving van de databank 8](#_Toc523353915)

[3 Ontwerpbeslissingen 9](#_Toc523353916)

[3.1 Consistency & replication 9](#_Toc523353917)

[3.2 Caching 9](#_Toc523353918)

[3.3 Security 9](#_Toc523353919)

[3.4 Recovery 9](#_Toc523353920)

[4 Api’s 10](#_Toc523353921)

[5 Reflectie 11](#_Toc523353922)

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.

**Druk <Ctrl + Alt + Shift + S> om het taakvenster met stijlen weer te geven**

# 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.

# 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.

# 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.

## 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.

## 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.

## Security

## Recovery

# Api’s

# Reflectie

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](http://www.kuleuven.be)