Distributed Systems (H0N08A)

Report: Distributed Cloud Applications

Jules Verbessem *(r0957436)*
Gert-Jan Gillis *(r0674083)*

December 7, 2023

11. How does using a NoSQL database, such as Cloud Firestore, affect scalability and availability? NoSQL databases zijn ontworpen voor horizontal scalability. Ze kunnen eenvoudig scalen door meer nodes aan de database toe te voegen. Dit is cruciaal voor het verwerken van grote hoeveelheden data en hoge traffic loads. Data kopies worden verdeeld over verschillende nodes, dit zorgt voor een hoge availability. Door de horizontal scalability van een NoSQL database bekomen we een hoge scalability en availability. Die twee gaan hand in hand.

12. How have you structured your data model (i.e. the entities and their relationships), and why in this way? We hebben een trains-collectie en die heeft een collectie van seats. Als je een ticket hebt aangemaakt, wordt dit bijgehouden in de bijhorende seat. Wanneer een booking is gemaakt worden er ook tickets toegevoegd aan de booking. Dit zorgt er wel voor dat dezelfde ticket-data 2 keer is opgeslagen. Maar het aantal keer lezen/schrijven is veel effeciënter op deze manier. Voor onze crash recovery implementatie is het belangrijk om te weten welke seats al tickets hebben. Als we dat moeten gaan opzoeken in de booking collection zouden dit veel onnodige leesopdrachten zijn.

NoSQL Databases zijn ontworpen om een grote hoeveelheid documenten te verwerken. Je hoeft je dus geen zorgen te maken over het schrijven van veel documenten. Om bijvoorbeeld een enkele zitplaats op te zoeken, is het efficiënter om over veel kleine documenten met 1 enkele zitplaats te loopen, dan te itereren over een groot document van alle zitplaatsen.

Ons datamodel is uitbreidbaar, want er kunnen verschillende treinen worden toegevoegd aan de trains-collectie die dan allemaal hun eigen seats-collectie hebben.

13. Compared to a relational database, what sort of query limitations have you faced when using the Cloud Firestore? Bij een gewone relationele database kan je rechtstreeks bij de table "Ticket" vragen of die een relatie heeft met een record uit de table "Seat". Bij schemaless NoSQL database moet je werken met geneste collecties. Waardoor je door collecties moet gaan om uiteindelijk iets te vinden. Bijvoorbeeld om een ticket te zoeken van een bepaalde seat moet je: trains/seats/ticket. Je kan geen gejoinde tabellen opvragen in schemaless NoSQL, wat in een relationele databank wel kan. Dit hebben we niet moeten gebruiken maar is een zeer krachtige feature.

14. How does your implementation of transactional behaviour for Level 2 compare with the all-or-nothing semantics required in Level 1 ? Wij hebben ervoor gekozen om de transactional behaviour niet te gebruiken. Want onze all-or-nothing semantics van level 1 met feedback werken even goed. Als we de transactional behaviour zouden willen gebruiken zouden we een groot aantal acties die niets te maken hebben met Firestore ook binnen in de transaction moeten steken. Dit leek ons niet de bedoeling van de transacties implementatie van Firestore. Deels omdat onze all-or-nothing semantics hetzelfde gedrag nabootst.

15. How have you implemented your feedback channel? What triggers the feedback channel to be used, how does the client use the feedback channel and what information is sent? Wanneer alle tickets geboekt zijn en de data is opgeslagen in Firestore, dan wordt de EmailController aangesproken om een email te verzenden. De informatie die verzonden wordt is te zien op de afbeelding onderaan: email success (figure 1). Wanneer er ergens een probleem is met het boeken van een ticket, begint het rollback process. Erna zal de EmailController worden aangesproken en wordt er een email verzonden. De informatie die verzonden wordt is te zien in de afbeelding onderaan: email failure (figure 2)

16. Did you make any changes to the Google App Engine configuration to improve scalability? If so, which changes did you make and why? If not, why was it not necessary and what does the default configuration achieve? Wij hadden gemerkt dat wanneer we allebei de applicatie testen er een heel deel instanties starten. We hadden besloten om de parameters: "target-cpu-utilization en target-throughput-utilization allebei op 0.7 te zetten. Maar hierdoor ondervonden we dat de applicatie een stuk trager werd. Dus hebben we besloten om toch maar de 2 parameters op de orginele 0.9 te zetten.

17. What are the benefits of running an application or a service (1) as a web service instead of natively and (2) in the cloud instead of on your own server? 1) De applicatie laten werken als een webservice in plaats van een applicatie voor één bepaalde OS laat toe om maar één applicatie te bouwen voor verschillende apparaten. Wat een pak minder development tijd ineemt. Ook maakt de webservice het gebruiksrvriendelijker voor de eindgebruiker omdat hij/zij aan onze applicatie kan van op al zijn/haar verschillende apparaten. 2) Door in de cloud te werken verbeter je de scalability van de applicatie. Om de service op een eigen webserver te draaien moet je webserver sterk genoeg zijn om de "high loads" aan te kunnen. De webserver zal trouwens 24/7 energie gebruiken om deze "high load" kracht te kunnen aangeven. Terwijl de cloud enkel maar aanrekent wat de gebruikers gebruiken. Ook heeft de cloud veel betere availability. Als de lokale web server down is dan is de applicatie niet meer toegankelijk. Zo'n situatie zou nooit voorkomen bij de cloud.

18. What are the pitfalls when migrating a locally developed application to a real-world cloud platform? What are the restrictions of Google Cloud Platform in this regard? Om lokaal nieuwe functionaliteit te testen of een bug te debuggen moet je enkel de applicatie aanpassen. Als je wilt debuggen in een cloud platform moet je eerst de applicatie "packagen" en erna deployen. Dit samen kan al snel enkele minuten duren. Daarom moet je beter nadenken over welke acties je uiteindelijk gebruikt om te debuggen. Het wordt moeilijker om een "trail-end-error" debug methode te gebruiken. Door te werken met een Google Cloud Platform komen er ook extra technische complexiteiten bij. De complexiteit is zeker nodig voor een real-world application. Maar enkel door ervaring worden de technische aspecten makkelijker te begrijpen.

19. How extensive is the tie-in of your application with Google Cloud Platform? Which changes do you deem necessary for migrating to another cloud provider? Om van Google Cloud Platform naar een andere cloud provider over te gaan moeten we volgende zaken veranderen:

- Voor het verifiëren van de JWT token moeten we de public key maken op basis van een ander certificaat
- We moeten in plaats van PubSub te gebruiken, overgaan naar een andere manier van indirecte communciatie. Bijvoorbeeld RabbitMQ
- Voor het versturen van emails zouden we in de nieuwe cloud provider ofwel ook SendGrid moeten kunnen gebruiken of een nieuwe email provider gebruiken

- Onze NoSQL database in Firestore zou moeten nageboud kunnen worden in een andere NoSQL database deze moet niet perse van alternatieve cloud provider zijn.

Hierbovenop moet je de nieuwe UI en functionaliteit van de nieuwe cloud provider nog aanleren. Het is dus ten strengste aangeraden om eerst een grondige studie te doen over welke cloud provider je best gebruikt, want het is redelijk moeilijk om van de ene naar de andere over te gaan. Maar het is wel mogelijk.

20. What is the estimated monthly cost if you expect 1000 customers every day? How would this cost scale when your customer base would grow to one million users a day? Can you think of some ways to reduce this cost? Het kost voor 0.036857 euro per uur per instantie. Als we ervanuit gaan dat iedere gebruiker een eigen instantie heeft en dat iedere gebruiker die instantie gebruikt voor max een half uur per dag: dan hebben we 15000 uren per maand. Totale kost per maand komt dan neer op: 552.86 euro per maand. Als onze applicatie door een miljoen gebruikers per maand gebruikt wordt, dan kost dit ons 552 855 euro per maand. Deze kosten zijn dan nog zonder de kosten voor de opslag van Firestore. Toen we de "Compute Engine workload estimate" gebruikten om de workload kost te berkenen kwamen we te weten dat de kosten per gebruiker (dus per instantie) niet verminderen wanneer er meer instanties runnen. Zie afbeeldingen 1KUsers en 1MUsers.

Door niet te snel en dus te veel instanties aan te maken bij workload, verminder je de totale kost. Maar dit maakt de applicatie ook trager. Door minder gegevens op te vragen en gegevens te gaan cachen in de frontend, wordt de workload op de instanties verminderd. Wat op zijn beurt ook de kost vermindert.

21. Include the App Engine URL where your application is deployed. Make sure that you keep your application deployed until after you have received your grades. https://distributedsystems-597e7.ew.r.appspot.com/ De inlog gegevens voor de applicatie zijn: jules.verbessem@student.kuleuven.be testdemo

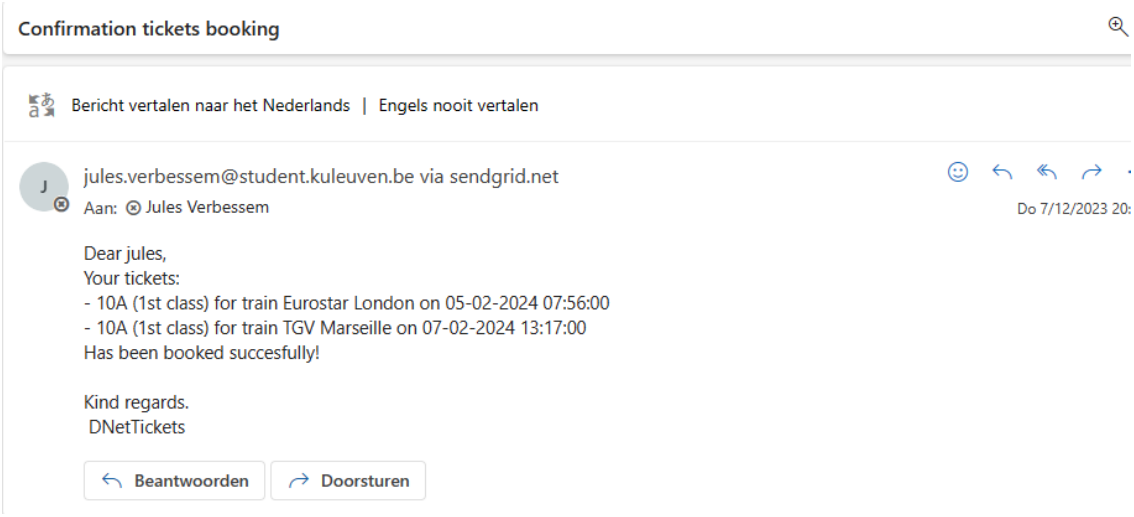

Figure 1: Email success

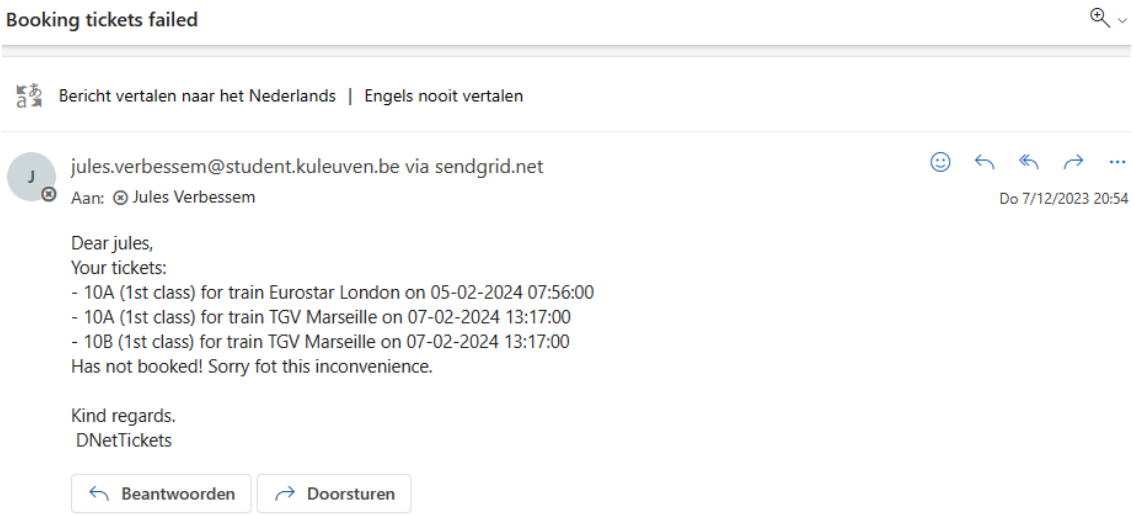

Figure 2: Email failure

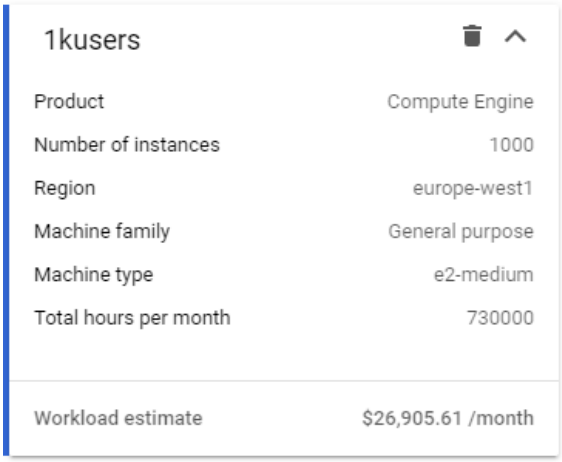

Figure 3: 1KUsers

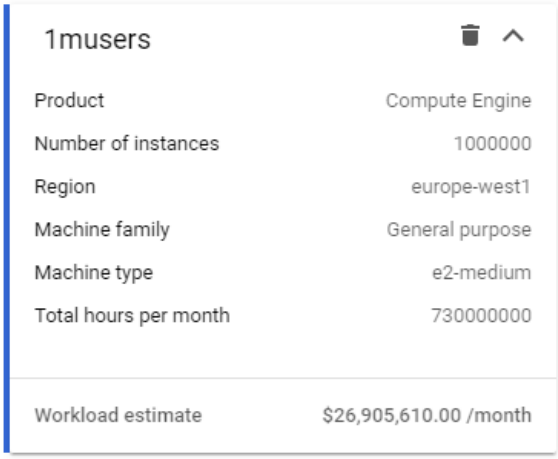

Figure 4: 1MUsers