Wat is de meest performante manier om data te ontvangen en te verzenden op het web: Een vergelijkende studie tussen gRPC met Protocol Buffers en REST met JSON en proof-of-concept

Onderzoeksvoorstel Bachelorproef 2020-2021

Sven Pepermans[1]

Samenvatting
In deze bachelorproef zal onderzoek gedaan worden naar de verschillen in performantie, tooling en usability tussen google Remote Procedure Call (gRPC) met Protocol Buffers en Representational State Transfer (REST) met JavaScript Object Notation (JSON). Eerst zal er onderzocht worden wat gRPC, Protocol Buffers, REST en JSON zijn en hoe deze werken. In het tweede deel wordt aan de hand van een proof-of-concept onderzocht welke manier van data verzenden en ontvangen het meest performant is en welke de betere usability heeft.
De relevantie van dit onderzoek is te vinden in de digitalisering van bedrijven, alsmaar meer data wordt verzonden en ontvangen via het web aan de hand van webapplicaties zoals ASP.NET Core apps. Dit maakt de nood aan een performante manier van data te verzenden en ontvangen des te noodzakelijker. Door het onderzoeken van het performantste dataformaat kan er binnen bedrijven bespaard worden op resources zoals processing power. Alsook zal een performant dataformaat de user experience verbeteren door de hogere doorvoersnelheid. Van deze studie wordt verwacht dat gRPC een hogere snelheid en een lager CPU gebruik zal hebben.

Sleutelwoorden
Webapplicatieontwikkeling. gRPC — JSON — API

Co-promotor

Contact: [1] sven.pepermans@student.hogent.be;

Inhoudsopgave

1 **Introductie** 1
2 **State-of-the-art** 1
3 **Methodologie** 2
4 **Verwachte resultaten** 2
5 **Verwachte conclusies** 2
Referenties 2

1. Introductie

Zonder dat het te weten of te beseffen maakt de internetgebruiker continue gebruik van dataformaten en protocollen. Deze komen in verschillende soorten, XML, JSON, Protocol Buffers, GraphQL, ... Deze data formaten zorgen aan de hand van een achterliggende implementatie in een Application Programming Interface (API) voor dat mensen hun vrienden hun nieuwe Facebook of Instagrampost kunnen zien in een internetbrowser of op een app. APIs maken het meest gebruik van XML en JSON, indien toch een dominerend formaat gekozen moet worden zal dit JSON zijn. JSON is veel sneller om te lezen en te schrijven dan XML.

Echter wil dit niet zeggen dat JSON het snelste data formaat is. In deze Bachelorproef zal onderzocht worden of Protocol Buffers geïmplementeerd aan de hand van gRPC eventueel een even performant of zelfs performanter alternatief kan zijn als REST API dat JSON gebruikt. De bijhorende onderzoeksvragen zullen dan ook als volgt zijn:

- Is gRPC met ProtocolBuffers sneller dan een REST API met JSON?
- Is gRPC met ProtocolBuffers efficiënter dan een REST API met JSON?

2. State-of-the-art

gRPC is open source Remote Procedure Call (RPC) framework en is in 2015 ontstaan uit zijn voorganger Stubby, deze was een single general-purpose RPC infrastructuur (gRPC Authors, g.d.). Deze technologie laat het toe voor een programma om procedures te starten op andere computers in, eventueel, verschillende adresruimten aan de hand van een smal comminucatiekanaal (Nelson, 1981). gRPC zelf is geen dataformaat zoals JSON maar kan gebruik maken van verschillende data formaten, standaard is dit Protocol buffers, ook wel Protobufs genoemd.

JSON en Protocol Buffers hebben zowel gelijkenissen als grote verschillen, beide zijn een language-neutral dataformaat zo blijkt uit de documentatie van Protocol Buffers (Google, g.d.) en JSON (ECMA, 2017). Het grootste, visuele verschil voor de gebruiker is te vinden in de structuur. JSON is een collectie van naam/value paren of een geordende lijst van waarden, daarintegen maken Protobufs gebruik van een soort model, er moet maar eenmaal gedefinieerd worden hoe de data gestructureerd zal zijn, daarna kan gebruik gemaakt worden van gegenereerde code om gemakkelijk data van en naar een variëteit van data streams te lezen en schrijven. Dit kan allemaal geprogrammeerd worden in heel wat verschillende programmeertalen.

gRPC wordt reeds gebruikt door verschillende grote bedrijven zoals Cisco en Netflix. Cisco gebruikt gRPC als het ideale, uniforme transportprotocol voor modelgestuurde configuratie en telemetrie. Dit komt dankzij de ondersteuning voor hoogwaardige bidirectionele streaming, op TLS gebaseerde beveiliging en het brede scala aan programmeertalen. Netflix koos dan weer voor gRPC omdat ze belang hechtte aan de architectonische kennis in de IDL (proto) dat een onderdeel is van gRPC en aan de, van deze proto-afgeleide, codegeneratie. Daarnaast speelde ook de cross-language compatibility en codegeneratie in gRPC een belangrijke rol bij de keuze voor gRPC bij Netflix (Foundation, 2018).

Eerder werd er nog geen officieel onderzoek uitgevoerd naar dit onderwerp, echter zijn wel online artikels te vinden die gRPC met REST gaan vergelijken aan de hand van een benchmark. Eén zo een onderzoek is uitgevoerd door Fernando (2019). Dit onderzoek was een benchmark tussen gRPC met Protocol Buffers en REST met JSON concludeerd dat gRPC sneller was dan REST behalve bij het streamen van data, hier is gRPC iets trager als REST.

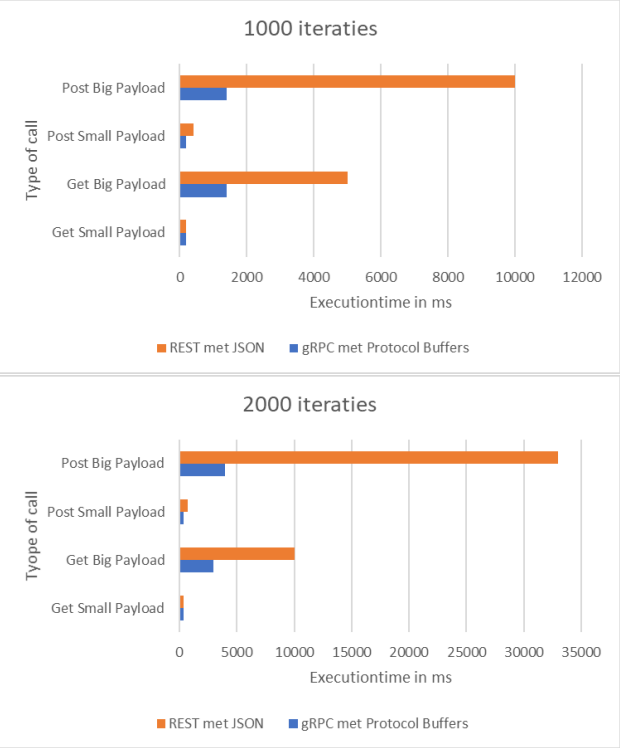
Het onderzoek dat in deze Bachelorproef zal uitgevoerd worden is gebaseerd op dat van Ruwan Fernando, hiermee wordt bedoeld dat deze proof-of-concept geprogrammeerd zal worden in C# en dat de verschillende impelementaties met elkaar vergeleken zullen worden aan de hand van een benchmark. Het verschil met Fernando R. zijn onderzoek kan men vinden in de verwerking van de data, in dit onderzoek zal de data uit een aanhangende databank gehaald worden. Alsook zal in dit onderzoek een tienvoud van het aantal iteraties doen.

3. Methodologie

Het onderzoek zal gevoerd worden aan de hand van simulaties en experimenten in een Proof-of-Content (PoC). Er zullen 2 identieke APIs opgesteld worden in C#, een .NET Core REST API voor JSON, en .NET Core API voor gRPC. Aan de hand van een benchmark tool, dewelke nog niet specifiek gekozen is, zullen er 2 verschillende categorieën aan GET en POST calls uitgevoerd worden. Big payload calls, hier zal substantief meer data opgehaald en verzonden worden dan bij de Small payload calls. Daarnaast zal het experiment twee maal uitgevoerd worden, een eerste keer met 1000 iteraties en een tweede maal met 2000 iteraties, dit om inaccurate metingen van kleine uitvoertijden te verminderen. Eens de resultaten van de experimenten verzameld zijn zullen we deze vergelijken en conclusies trekken.

4. Verwachte resultaten

Bij de grote payloads wordt een duidelijk verschil in voordeel van gRPC en Protocol Buffers verwacht, echter wordt er wel ook verwacht dat REST API met JSON in de kleine payloads nog degelijk zal presteren en misschien zelf nog licht overwegend beter zal zijn dan gRPC en Protocol Buffers.



5. Verwachte conclusies

Dit onderzoek moet doen blijken of gRPC en Protocol Buffers sneller en efficiënter zijn dan een REST API met JSON. In de PoC zal duidelijk worden dat gRPC en Protocol Buffers een sneller en efficiënter alternatief zijn voor JSON. Dit resultaat zou eventueel verklaard kunnen worden doordat JSON een ouder dataformaat is en dus niet geoptimaliseerd is voor de huidige technologieën terwijl gRPC zeer recent ontworpen is om optimaal met de huidige technologieën om te gaan.

Referenties

ECMA. (2017). *The JSON Data Interchange Syntax*. ECMA International. Verkregen 16 december 2020, van https://www.ecma-international.org/publications/files/ECMA-ST-ARCH/ECMA-404%201st%20edition%20October%202013.pdf

Fernando, R. (2019, april 3). *Evaluating Performance of REST vs. gRPC*. https://medium.com/@EmperorRXF/evaluating-performance-of-rest-vs-grpc-1b8bdf0b22da

Foundation, C. N. C. (2018, december 4). *Netflix: Increasing developer productivity and defeating the thundering herd with gRPC*. https://www.cncf.io/case-studies/netflix/

Google. (g.d.). *Protocol Buffers*. Google. https://developers.google.com/protocol-buffers

gRPC Authors. (g.d.). *gRPC, A high-performance, open source universal RPC framework*. https://grpc.io/

Nelson, B. J. (1981). *Remote Procedure Call* (proefschrift). Carnegie-Mellon University. https://ia801900.us.archive.org/22/items/bitsavers_xeroxparctoteProcedureCall_14151614/CSL-81-9_Remote_Procedure_Call.pdf