JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 33 (2024) 323–330
Received: 23 August 2024
Accepted: 14 October 2024

Analysis of the performance and scalability of microservices depending on the communication technology

Analiza wydajności i skalowalności mikrousług w zależności od technologii komunikacji

Patryk Iwanowski, Jan Jarmoszewicz*, Małgorzata Plechawska-Wójcik

Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract
The aim of this study is to compare the performance and scalability of microservices based on the communication technology used. The research is conducted on four systems, each consisting of three microservices, utilizing REST and gRPC technologies, as well as two communication schemes between the services. The systems are tested on two experimental setups with different system resources, considering metrics such as response time, memory usage, and CPU usage. The obtained results indicate that applications based on gRPC are generally more efficient and scalable but require more CPU resources, whereas REST-based applications require more RAM resources.

Keywords: gRPC; REST; performance of microservices applications; scalability of microservices applications

Streszczenie
Celem niniejszej pracy jest porównanie wydajności i skalowalności mikrousług w zależności od technologii komunikacji. Badania bazują na czterech systemach, z których każdy składa się z trzech mikroserwisów, opartych odpowiednio na technologii REST oraz gRPC, a także dwóch schematach komunikacji między serwisami. Systemy testowane są na dwóch stanowiskach badawczych o różnych zasobach systemowych oraz uwzględniają metryki, tj. czas odpowiedzi, użycie pamięci czy użycie procesora. Uzyskane wyniki pozwalają stwierdzić, że aplikacje oparte na gRPC są w większości przypadków wydajniejsze i bardziej skalowalne, a zarazem wymagają większej ilości zasobów procesora w przeciwieństwie do REST, który natomiast wymaga większej ilości zasobów pamięci RAM.

Słowa kluczowe: gRPC; REST; wydajność aplikacji mikroserwisowych; skalowalność aplikacji mikroserwisowych

*Corresponding author
Email address: **jan.jarmoszewicz@pollub.edu.pl** (J. Jarmoszewicz)

Published under Creative Common License (CC BY 4.0 Int.)

1. Wstęp

W obliczu wszechobecnej architektury mikrousługowej w systemach informatycznych, istnieje potrzeba określenia wydajności i skalowalności serwisów wykorzystujących różne technologie komunikacji. Problem ten dotyczy wielu architektów i administratorów systemów, którzy muszą podjąć decyzję dotyczącą wyboru najbardziej odpowiedniej technologii komunikacji dla ich potrzeb. W związku z istnieniem takiego problemu należy podjąć odpowiednie kroki i przeprowadzić badania, które pozwolą dostarczyć szczegółowych informacji na temat tego, która z technologii komunikacji dominuje w konkretnych środowiskach i systemach.

Technologie komunikacji różnią się sposobem przesyłania danych między usługami, a ich wybór może znacząco wpływać na aspekty takie jak wydajność, skalowalność, elastyczność i złożoność implementacji. Głównym celem artykułu jest przeprowadzenie głębokiej analizy porównawczej wydajności i skalowalności mikrousług opierających się na komunikacji synchronicznej. Szczególną uwagę skupiono na porównaniu gRPC, reprezentującego protokół RPC (ang. Remote Procedure Call) oraz REST (ang. Representational State Transfer), który bazuje na architekturze odnośników i zapytań HTTP.

Wyniki tego badania mogą mieć istotne znaczenie dla profesjonalistów w dziedzinie inżynierii oprogramowania, architektury systemów oraz zarządzania projektami, gdyż dostarczają praktycznych wskazówek dotyczących wyboru optymalnej technologii komunikacji w zależności od wymagań i celów systemu opartego na mikrousługach. W rezultacie artykuł ma na celu wzbogacenie wiedzy na temat projektowania, wdrażania i utrzymania systemów opartych na architekturze mikrousług, przyczyniając się do efektywnego rozwoju nowoczesnych aplikacji o znaczącej skali i złożoności. Przedstawione poniżej wyniki badań nie wyczerpują w pełni tematu doboru odpowiedniej technologii komunikacji mikroserwisowej, lecz dostarczą istotnego wglądu w kluczowe elementy podczas projektowania mikrousług oraz stanowią wstęp do bardziej szczegółowych badań nad tym zagadnieniem.

2. Przegląd literatury

Architektura mikrousługowa na stałe zakorzeniła się w produkcji dzisiejszych systemów informatycznych. Firmy stosują różne technologie komunikacji między mikroserwisami w swoich projektach, w zależności od potrzeb ich konsumentów i deweloperów. Obecne środowisko naukowe udostępnia wiele publikacji naukowych poświęconych analizie wydajności, skalowalności oraz przedstawienia wad i zalet praktycznie każdej z nich.

Analiza wybranych artykułów pozwala na głębsze zrozumienie zagadnienia, a także zapoznanie się z potencjalnymi sposobami przeprowadzania badań. Kolejnym jej atutem jest możliwość porównania otrzymanych wyników z wynikami innych naukowców w celu weryfikacji postawionych hipotez, a także zrozumienia ewentualnych błędów lub nieścisłości powstałych w procesie badawczym.

Pierwszym z nich jest praca [1] dotycząca benchmarku mikroserwisów, w której przedstawione zostało podejście umożliwiające zbudowanie aplikacji opartej na mikrousługach, jednocześnie umożliwiającej pomiar parametrów określających jakość komunikacji. Całość rozwiązania opiera się na interfejsie API REST wystawianym przez mikroserwis w celu mapowania na niego określonych przez użytkownika operacji.

Kolejną pracą, którą wzięto pod uwagę jest artykuł [2] dotyczący analizy wydajności mikroserwisów pod kątem wzorca projektowego. Autorzy tego artykułu oparli się na przedstawieniu wyników wydajnościowych takich jak czas odpowiedzi mikrousług w zależności od podejścia przy projektowaniu aplikacji. Wyniki pracy wskazały, że żaden z opisanych wzorców projektowych nie uzyskuje znaczących różnic nad innymi pod względem wydajności, a wybranie konkretnego wzorca powinno być przemyślane pod kątem przeznaczenia aplikacji.

W pracy [3] dotyczącej porównania wydajności protokołów komunikacyjnych między mikroserwisami, autorzy skupili się głównie na trzech z nich, tj. REST, gRPC oraz Thrift, dla których sprawdzali takie metryki jak pamięć, zużycie procesora oraz czas odpowiedzi. Ich badania wykazały, że Thrift i gRPC osiągają niższe czasy odpowiedzi od REST, co oznacza, że są lepszymi środkami komunikacji pomiędzy serwisami.

Kolejna publikacja [4] opisuje degradacje wydajności aplikacji zależnie od wykonywanych przez mikroserwisy zadań. Autorzy pracy wytworzyli matrycę kalkulacji, która wskazuje połączenia pomiędzy mikrousługami z największą tendencją do degradacji pod względem wydajności. Rezultatem ich badań było zaobserwowanie największej korelacji pomiędzy serwisem zliczającym dzienne podsumowanie faktur, a serwisem zwracającym listę faktur dla danej kasy fiskalnej.

Autorzy pracy [5] opisują podejście generowania testów obciążeniowych w celu automatycznej oceny skalowalności konfiguracji mikrousług. Podejście to pozwala na podejmowanie decyzji dotyczącej wyboru konfiguracji pod względem wydajności systemu czy wykrywaniu anomalii. Wyniki ich badań dowiodły, że istnieje potrzeba dokładnej oceny wpływu zwiększenia zasobów na wydajność mikrousługi. Dodatkowo zaobserwowali, że oceniana metryka oparta na domenie nie jest wprost proporcjonalna do liczby zasobów.

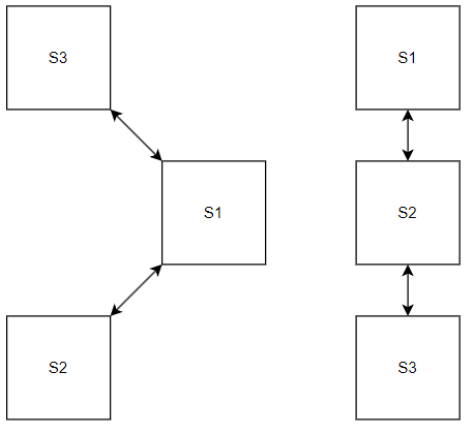
W publikacji [6] zostały porównane najbardziej rozpowszechnione technologie komunikacji pomiędzy usługami. Autorzy przeprowadzili badania w standardzie przemysłowym w celu uzyskania informacji o czasie opóźnień. Wyniki testów wykazały, że protokół gRPC działa najlepiej pod względem czasu odpowiedzi oraz przepustowości, w porównaniu do HTTP i WebSocket.

3. Cel i zakres badań

Celem pracy jest porównanie wydajności i skalowalności aplikacji poprzez weryfikację czasu odpowiedzi, przepustowości, stabilności i zużycia zasobów systemowych przy zastosowaniu technologii komunikacji gRPC oraz REST, a także uzyskanie odpowiedzi na następujące pytania:

1. Czy i w jakim stopniu gRPC posiada znaczną przewagę wydajności nad REST.
2. Czy i jaki typ przesyłanych danych ma znaczenie w zestawieniu wyników porównawczych pomiędzy badanymi technikami komunikacji.
3. Czy i w jakim stopniu specyfikacja komputerowa wpływa na czas realizacji przesyłania danych, oraz zużycie zasobów systemowych w przypadku poszczególnych technologii komunikacji
4. Czy i w jakim stopniu schemat systemu opartego na mikrousługach ma znaczenie w przypadku wcześniej wymienionych aspektów.

Do wykonania badań wykorzystywane są cztery systemy w dwóch różnych schematach (Rysunek 1), tj. gwiazdy oraz liniowym (w dalszej części artykułu określanych mianem topologii), każdy składający się z trzech mikroserwisów komunikujących się ze sobą za pomocą technologii gRPC oraz REST. Każdy z systemów posiada identyczne funkcjonalności względem porównywanych technologii i topologii oraz korzysta z bazy danych PostgreSQL.



Rysunek 1: Schematy aplikacji typu gwiazda (lewy) oraz liniowy (prawy).

Hipotezy badawcze:

H1. gRPC osiąga wyższą przepustowość niż REST, ze względu na korzystanie z formatu binarnego.
H2. Nie ma istotnej różnicy w skalowalności pomiędzy gRPC oraz REST.
H3. Nie ma istotnej różnicy w zużyciu zasobów systemowych pomiędzy gRPC oraz REST.

4. Metody badań

Do przeprowadzania badań wykorzystywane są następujące technologie: Spring Boot – implementacja aplikacji, gRPC oraz REST – komunikacja między serwisami, Gatling – testy obciążeniowe, Prometheus oraz Grafana – monitoring aplikacji, PostgreSQL – utrwalanie danych, Docker – konteneryzacja bazy danych oraz monitoringu.

4.1. Plan badań

Analiza wydajności i skalowalności technologii komunikacji realizowana jest na podstawie poniższego planu:

1. Utworzenie dwóch systemów w dwóch różnych topologiach serwisów, tj. gwiazdy oraz liniowej, z których każdy składa się z trzech aplikacji napisanych z użyciem szkieletu programistycznego Spring Boot, wykorzystujących technologię gRPC oraz REST do komunikacji, a także bazy danych PostgreSQL.
2. Wykonanie testów obciążeniowych za pomocą wtyczek Gatling na dwóch stanowiskach badawczych o różnych zasobach systemowych.
3. Porównanie oraz opracowanie wyników na podstawie dwóch źródeł:
   a) zbiorowego wyniku z aplikacji Gatling.
   b) graficznego przedstawienia wyników uzyskanych z Prometheus'a za pomocą aplikacji Grafana.
4. Przedstawienie wniosków.

4.2. Przygotowanie scenariuszy badawczych

Do przeprowadzenia badania wydajności i skalowalności wykorzystywane są dwa scenariusze badawcze przedstawione w Tabeli 1 i Tabeli 2. Pierwszy scenariusz opisujący standardowe operacje CRUD na danych o małym rozmiarze (około 1 kB) oraz drugi opisujący wysyłanie i pobieranie danych o dużym rozmiarze (875 kB). Należy nadmienić również istniejące w tych scenariuszach różnice, a mianowicie scenariusz nr 1 wykonywany jest na nieokreślonej liczbie użytkowników, a dokładniej w danej jednostce czasu uruchamiana jest skonfigurowana liczba użytkowników. Natomiast scenariusz nr 2, po wstępnych badaniach, ze względu na rozmiar danych oraz chęć uzyskania miarodajnych wyników, wykonywany jest na określonej liczbie użytkowników w czasie (nowi użytkownicy dodawani są tylko w momencie, gdy poprzedni skończą swoje zadanie, tak aby utrzymywać ich stałą liczbę), co w dalszej części artykułu przekłada się na procentowy stosunek poprawnych odpowiedzi.

Tabela 1: Scenariusz testowy numer 1

Scenariusz nr 1	
Akcja	Opis
1	Pobranie wszystkich rekordów
2	Dodanie pojedynczego rekordu
3	Edycja pojedynczego rekordu
4	Usunięcie pojedynczego rekordu

Tabela 2: Scenariusz testowy numer 2

Scenariusz nr 2	
Akcja	Opis
1	Przesłanie zdjęcia
2	Pobranie zdjęcia

Proces testowania przeprowadzany jest przy użyciu wtyczek Gatling, które pozwalają na wykonanie następujących symulacji dla ww. scenariuszy:

- Dla scenariusza nr 1:
  1. Wykonanie scenariusza dziesięciokrotnie przez wirtualnego użytkownika. Liczba użytkowników na sekundę oscyluje w liczbie 300. Symulacja testowa trwa 60 sekund.
  2. Wykonanie scenariusza pięciokrotnie przez wirtualnego użytkownika. Liczba użytkowników na sekundę oscyluje w liczbie 150. Symulacja testowa trwa 60 sekund.
  3. Wykonanie scenariusza jednokrotnie przez wirtualnego użytkownika. Liczba użytkowników na sekundę zwiększa się stopniowo od 100 do 1200. Symulacja testowa trwa 60 sekund.
- Dla scenariusza nr 2:
  1. Wykonanie scenariusza dwukrotnie przez wirtualnego użytkownika. Liczba aktywnych użytkowników oscyluje w liczbie 100. Symulacja testowa trwa 60 sekund.
  2. Wykonanie scenariusza jednokrotnie przez wirtualnego użytkownika. Liczba aktywnych użytkowników oscyluje w liczbie 50. Symulacja testowa trwa 60 sekund.
  3. Wykonanie scenariusza jednokrotnie przez wirtualnego użytkownika. Liczba aktywnych użytkowników zwiększa się stopniowo od 2 do 100. Symulacja testowa trwa 60 sekund.

Dla obu scenariuszy symulacje nr 1 i nr 2 wykonywane są trzydziestokrotnie, a symulacja nr 3 dziesięciokrotnie, odpowiednio dla czterech opisanych wcześniej systemów. Dodatkowo w dalszej części artykułu dla uproszczenia każda z symulacji nazywana jest zgodnie z określonym schematem np. 5/150/60.

4.3. Opis stanowisk badawczych

Serie testów przeprowadzane są na dwóch stanowiskach badawczych (Tabela 3) o różnej specyfikacji, w celu określenia różnic w zachowaniu aplikacji w zależności od dostępnych zasobów systemowych, a także uzyskania informacji odnośnie skalowalności serwisów w zależności od poszczególnych technologii komunikacji. Dodatkowo w Tabeli 4, przedstawione zostały wersje wykorzystanych w procesie badawczym technologii i bibliotek.

Tabela 3: Parametry stanowisk badawczych

Stanowisko nr 1	
Procesor	Intel Core i7-13700F, 16 rdzeni
System operacyjny	Windows 11 Home 22H2 64-bit
Pamięć RAM	32 GB
Stanowisko nr 2	
Procesor	Intel Core i5-11400H, 6 rdzeni
System operacyjny	Windows 10 Home 22H2 64-bit
Pamięć RAM	16 GB

Tabela 4: Wersje użytych technologii

Technologie	
Java	17
Spring Boot	2.7.18
Spring Cloud Starter OpenFeign	3.1.9
gRPC Spring Boot Starter	2.15.0
gRPC API	1.58.0
PostgreSQL	16
Gatling	3.9.5
Prometheus	2.50.1
Grafana	10.0.12
Docker	25.0.3

5. Analiza wyników badań

W celu porównania wydajności oraz skalowalności technologii gRPC oraz REST, każda z symulacji uwzględnia pomiar następujących metryk oraz statystyk:

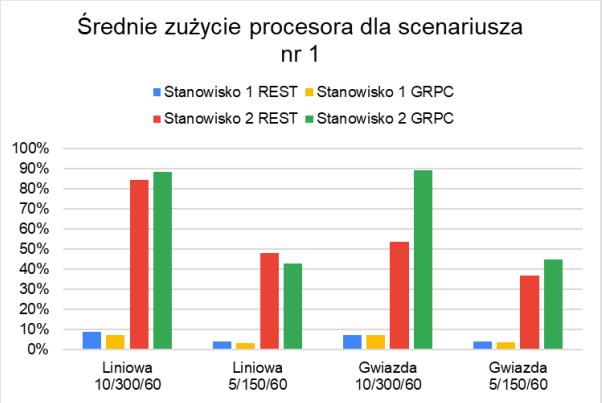
- średnie użycie procesora,
- średni czas wykonania scenariusza,
- średni czas odpowiedzi na żądania pomiędzy serwisami,
- średnie użycie pamięci RAM przez aplikację znajdującą się na wejściu każdego z czterech systemów,
- liczbę otrzymywanych żądań na sekundę,
- procent pozytywnych odpowiedzi z aplikacji,
- średnia liczba aktywnych użytkowników w momencie pierwszych problemów z przetwarzaniem kolejnych żądań.

Ostateczne wnioski określane są na podstawie powyższych wyników zestawionych ze sobą dla odpowiednich technologii, topologii oraz stanowisk badawczych.

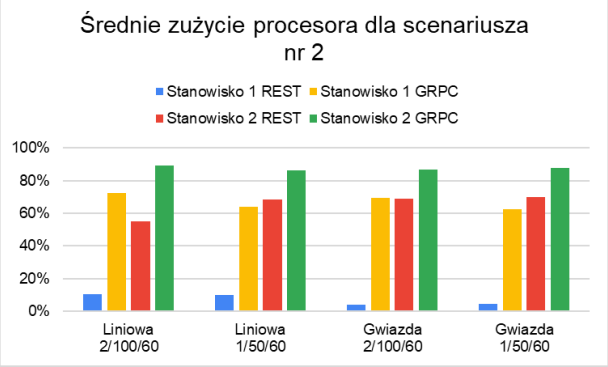
5.1. Analiza średniego użycia procesora

Na podstawie wyników procentowego użycia procesora dla scenariusza nr 1, zawartych na Rysunku 2, można zauważyć dwie odmienne charakterystyki poziomu użycia względem REST i gRPC dla poszczególnych stanowisk badawczych. W przypadku stanowiska nr 1, zarówno dla symulacji 5/150/60 jak i 10/300/60 w topologii liniowej, widoczne jest od 18% do 22% mniejsze zużycie procesora w przypadku korzystania z gRPC w stosunku do REST. Natomiast w przypadku stanowiska nr 2, podobny schemat widoczny jest tylko w symulacji 5/150/60 w systemie zbudowanym zgodnie z topologią liniową, w której średnie zużycie procesora dla gRPC jest o 11% mniejsze od REST. Analizując wyniki pozostałych symulacji dla tego stanowiska badawczego widać znacznie większe zużycie procesora przy wykorzystaniu technologii gRPC, które w najgorszym przypadku, dla symulacji 10/300/60 w systemie o topologii gwiazdy, jest większe od REST o 66%.

Przechodząc do analizy wyników dla scenariusza nr 2 widoczne jest znacznie większe ogólne zużycie zasobów procesora dla obu stanowisk badawczych. Praktycznie w każdym przypadku zużycie procesora jest większe dla aplikacji korzystających z technologii gRPC. Szczególną różnicę widać tutaj na stanowisku nr 1, dla którego gRPC wykorzystuje około 6 razy więcej zasobów w stosunku do REST, wykonującego identyczne zadanie. Taki wynik oznacza, że gRPC intensywniej wykorzystuje dostępne zasoby systemowe, co może wynikać z procesu serializacji pliku o dużym rozmiarze. W przypadku stanowiska nr 2 sytuacja charakteryzuje się podobnie jak w przypadku scenariusza nr 1, tj. zużycie procesora jest większe w przypadku korzystania z technologii gRPC, a różnica w stosunku do REST sięga nawet 62%.



Rysunek 2: Średnie zużycie procesora dla scenariusza nr 1.



Rysunek 3: Średnie zużycie procesora dla scenariusza nr 2.

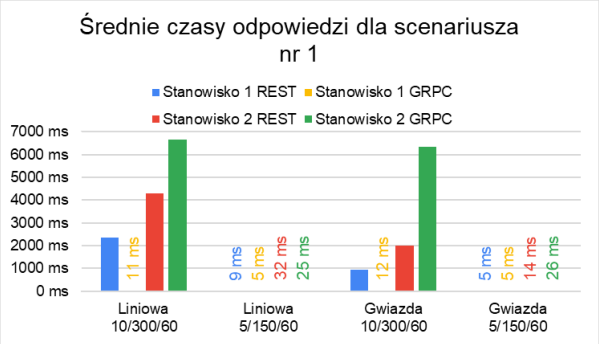
5.2. Analiza średniego czasu odpowiedzi na żądanie

Proces badawczy uwzględnia obliczanie trzech różnych czasów odpowiedzi, a mianowicie: czasu odpowiedzi systemu na żądania wysyłane za pomocą Gatling, czasu odpowiedzi serwisu nr 2 do serwisu nr 1 oraz czasu odpowiedzi serwisu nr 3 do serwisu nr 1 bądź nr 2 (zależnie od topologii systemu). Analiza tych czasów podzielona jest na dwa etapy: analizę całkowitego średniego czasu odpowiedzi oraz analizę średniego czasu odpowiedzi pomiędzy serwisami.

5.2.1. Analiza całkowitego średniego czasu odpowiedzi

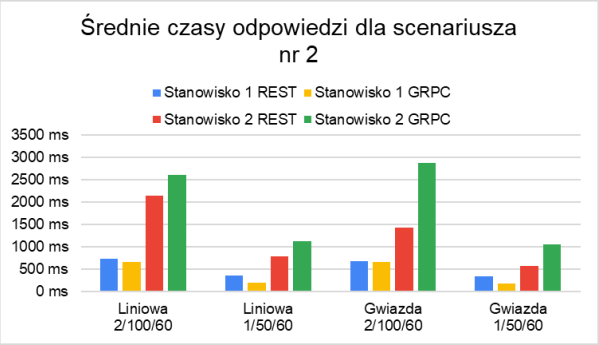
Analizując średni czas odpowiedzi dla scenariusza nr 1 zauważalne są różne wnioski. Mianowicie w przypadku stanowiska nr 1, w każdym z przeprowadzonych testów, gRPC wypada tak samo lub lepiej w stosunku do swojego przeciwnika. Największa różnica w czasach zauważalna jest w przypadku symulacji 10/300/60, w których przekracza ona ponad 1000 ms dla topologii gwiazdy oraz

2000 ms dla topologii liniowej. Spowodowane jest to przepustowością obu tych technologii, a mianowicie trudnościami REST z bezproblemowym obsłużeniem tej samej liczby żądań co gRPC, co szczegółowo opisane jest w jednym z kolejnych podrozdziałów. W związku z tym, w przypadku symulacji 10/300/60 pod uwagę wzięta jest również mediana, która dla REST w topologii liniowej wynosi 11 ms oraz 5 ms w topologii gwiazdy, natomiast dla gRPC wartości te wynoszą odpowiednio 5 ms oraz 4 ms, co tak jak w przypadku symulacji 5/150/60 świadczy o jego przewadze. Odmienna sytuacja widoczna jest w przypadku stanowiska nr 2, dla którego mediana czasu odpowiedzi technologii REST w topologii liniowej wynosi 1484 ms oraz 605 ms w topologii gwiazdy, tymczasem dla gRPC wartości czasów wynoszą odpowiednio 3933 ms oraz 3604 ms.



Rysunek 4: Średnie czasy odpowiedzi dla scenariusza nr 1.

Przechodząc do analizy średnich czasów scenariusza nr 2 sytuacja kształtuje się podobnie jak w przypadku scenariusza nr 1. Technologia gRPC na stanowisku nr 1 osiąga niższe średnie czasy odpowiedzi, osiągające nawet 50% czasu odpowiedzi REST, czego nie można powiedzieć o czasach otrzymanych na stanowisku nr 2, które dla gRPC są nawet dwukrotnie większe niż czasy otrzymane dla REST.



Rysunek 5: Średnie czasy odpowiedzi dla scenariusza nr 2.

5.2.2. Analiza średniego czasu odpowiedzi pomiędzy serwisami

Przeprowadzając analizę średnich czasów odpowiedzi pomiędzy poszczególnymi serwisami należy w inny sposób podzielić dane. Analizując metryki, takie jak zużycie procesora czy całkowity czas odpowiedzi na żądanie, wskazana jest ocena pod względem wykonywanych scenariuszy. Natomiast w przypadku średnich czasów odpowiedzi serwisów należy rozdzielić dane na podstawie konkretnych połączeń, czyli połączenia serwisu nr 1 z serwisem nr 2 oraz zależnie od topologii, serwisu nr 1 z serwisem nr 3 bądź serwisu nr 2 z serwisem nr 3. W dalszej części rozdziału konkretne połączenia określane są skrótowo używając następującego schematu, tj. połączenie serwisu nr 1 z serwisem nr 2, określane jest jako S1->S2.

Analizując czasy odpowiedzi dla połączenia S1->S2 można zauważyć, że dla stanowiska nr 1 niższe czasy odpowiedzi uzyskała technologia gRPC, podczas wykonywania obu scenariuszy testowych. Dla scenariusza nr 1 gRPC w symulacji 10/300/60 w topologii liniowej osiąga czas siedmiokrotnie niższy niż czas dla technologii REST, wynoszący 6 ms, natomiast dla symulacji 5/150/60 czas ten jest dwukrotnie niższy i wynosi 3 ms. W dwóch przypadkach ta tendencja jest złamana, a mianowicie, dla scenariusza nr 1 podczas wykonywania symulacji 10/300/60 oraz 5/150/60 dla topologii gwiazdy, technologia REST osiąga czas niższy, którego różnica nie przekracza 2 ms. W przypadku stanowiska nr 2, w każdej przeprowadzonej symulacji, REST osiąga niższe czasy odpowiedzi. Dla scenariusza nr 1 REST w dwóch przypadkach osiąga kilkunastokrotnie niższe wyniki od gRPC. Jedynie dla symulacji 5/150/60, w obu topologiach, czasy odpowiedzi pomiędzy technologiami różnią się o kilka milisekund. Wyniki dla scenariusza nr 2 charakteryzują się tą samą tendencją niższych czasów dla technologii REST, lecz z mniejszą różnicą niż dla scenariusza nr 1. W trzech przypadkach różnice te wynoszą od 100 ms do 250 ms, a w symulacji 2/100/60 dla topologii gwiazdy różnica wynosi ponad 1150 ms.

Tabela 5: Średnie czasy odpowiedzi dla serwisów S1->S2

		Liniowa 10/300/60 (ms)	Liniowa 5/150/60 (ms)	Gwiazda 10/300/60 (ms)	Gwiazda 5/150/60 (ms)
Scenariusz 1	Stan. 1 REST	45,26	6,28	5,00	1,87
	Stan. 1 GRPC	5,90	3,09	6,36	2,62
	Stan. 2 REST	41,22	19,21	17,57	6,32
	Stan. 2 GRPC	7282,1	16,3	7169,7	11,8
		Liniowa 2/100/60 (ms)	Liniowa 1/50/60 (ms)	Gwiazda 2/100/60 (ms)	Gwiazda 1/50/60 (ms)
Scenariusz 2	Stan. 1 REST	701,20	345,37	668,00	329,17
	Stan. 1 GRPC	458,47	113,10	430,07	88,16
	Stan. 2 REST	2189,00	772,97	1396,3	551,88
	Stan. 2 GRPC	2281,9	870,2	2527,9	795,1

Na podstawie wyników dla połączenia serwisów S1/S2->S3, przedstawionych w Tabeli 6, można stwierdzić, że dla scenariusza nr 1 technologia gRPC osiąga niższy czas odpowiedzi dla każdego stanowiska badawczego. W przypadku obu stanowisk czasy odpowiedzi serwisu dla technologii gRPC w topologii liniowej wynoszą od 200 ns do 700 ns. Scenariusz nr 2 cechuje się inną charakterystyką, a dokładniej, dla stanowiska nr 1 technologia gRPC osiąga niższe czasy odpowiedzi tylko w topologii liniowej. W pozostałych przypadkach testowych technologia REST uzyskuje niższe czasy odpowiedzi.

Tabela 6: Średnie czasy odpowiedzi dla serwisów S1/S2->S3

		Liniowa 10/300/60 (ms)	Liniowa 5/150/60 (ms)	Gwiazda 10/300/60 (ms)	Gwiazda 5/150/60 (ms)
Scenariusz 1	Stan. 1 REST	6,70	2,75	16,24	1,16
	Stan. 1 GRPC	0,0002	0,0003	1,00	0,90
	Stan. 2 REST	16,07	10,49	31,54	2,02
	Stan. 2 GRPC	0,0007	0,0006	20,8	1,7
		Liniowa 2/100/60 (ms)	Liniowa 1/50/60 (ms)	Gwiazda 2/100/60 (ms)	Gwiazda 1/50/60 (ms)
Scenariusz 2	Stan. 1 REST	687,00	332,60	4,74	4,70
	Stan. 1 GRPC	456,20	109,82	18,69	9,59
	Stan. 2 REST	2157,67	762,53	6,89	6,50
	Stan. 2 GRPC	2277,0	858,6	32,4	26,0

5.3. Analiza przepustowości

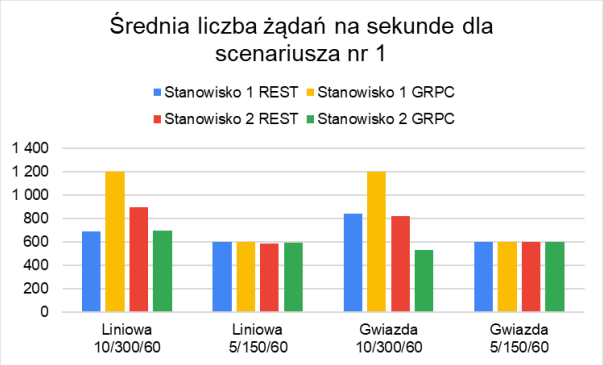
Przed analizą wyników średniej liczby żądań na sekundę należy nadmienić, że procent liczby poprawnych odpowiedzi dla obu stanowisk jest większy dla technologii gRPC i zaprezentowany jest w Tabeli 7.

Tabela 7: Procentowy stosunek poprawnych odpowiedzi

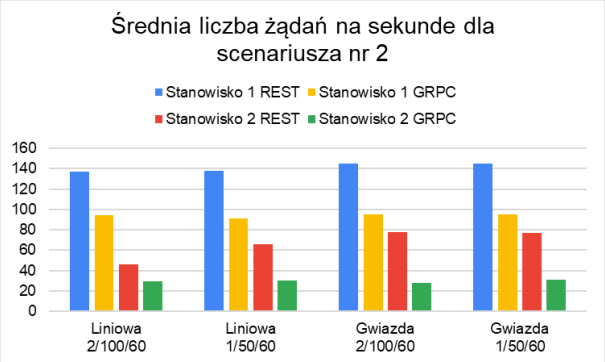
		Liniowa 10/300/60	Liniowa 5/150/60	Gwiazda 10/300/60	Gwiazda 5/150/60
Scenariusz 1	Stan. 1 REST	73%	100%	85%	100%
	Stan. 1 GRPC	100%	100%	100%	100%
	Stan. 2 REST	53%	100%	49%	100%
	Stan. 2 GRPC	86%	100%	88%	100%
		Liniowa 2/100/60	Liniowa 1/50/60	Gwiazda 2/100/60	Gwiazda 1/50/60
Scenariusz 2	Stan. 1 REST	100%	100%	100%	100%
	Stan. 1 GRPC	100%	100%	100%	100%
	Stan. 2 REST	100%	100%	100%	100%
	Stan. 2 GRPC	100%	100%	100%	100%

Analizując wyniki średniej liczby żądań na sekundę dla scenariusza nr 1 oraz stanowiska badawczego nr 1, widocznych na Rysunku 6, można stwierdzić, że w przypadku symulacji mniej obciążających, tj. 5/150/60, obie technologie wypadają podobnie i osiągają przepustowość 600 żądań na sekundę, czyli maksymalną osiągalną liczbę dla tej symulacji. Przechodząc do symulacji 10/300/60 widać zauważalne różnice pomiędzy przepustowością między REST i gRPC. Aplikacje korzystające z gRPC osiągają maksymalny możliwy wynik dla tej symulacji wynoszący 1200 żądań na sekundę, co stanowi prawie 200% liczby żądań uzyskiwanych przez aplikacje korzystające z REST dla topologii liniowej oraz 150% dla topologii gwiazdy. W przypadku stanowiska nr 2 rezultaty dla symulacji 10/300/60 wskazują na przewagę aplikacji korzystających z REST, które dla topologii liniowej oraz gwiazdy wykonują odpowiednio o 30% i 55% większą liczbę żądań na sekundę niż aplikacje korzystające z gRPC. Na podstawie rezultatów dla symulacji 5/150/60, gRPC osiąga takie same wyniki jak REST w przypadku obu topologii.

Wyniki dla scenariusza nr 2 charakteryzują się natomiast nieco inną tendencją dla obu stanowisk niż w przypadku scenariusza nr 1, a mianowicie, oba stanowiska dla wszystkich przeprowadzanych symulacji wskazują na przewagę REST nad gRPC. Zestawiając ze sobą wyniki osiągane przez gRPC oraz czas odpowiedzi dla scenariusza nr 2 i obu stanowisk badawczych można dojść do wniosku, że w przypadku stanowiska nr 2 wyniki wydają się racjonalne porównując czas odpowiedzi z liczbą żądań na sekundę, a mianowicie - większy czas odpowiedzi oraz mniejsza liczba żądań na sekundę. Identyczne zestawienie nieco inaczej prezentuje się w przypadku stanowiska nr 1, dla którego aplikacje korzystające z gRPC wykonują mniej żądań na sekundę pomimo nawet dwukrotnie niższego czasu odpowiedzi dla systemu w topologii gwiazdy. Biorąc pod uwagę dodatkowo zużycie procesora, można stwierdzić, że wyniki te mogą być spowodowane procesem serializacji i deserializacji formatu Protobuf.



Rysunek 6: Średnia liczba żądań na sekundę dla scenariusza nr 1.



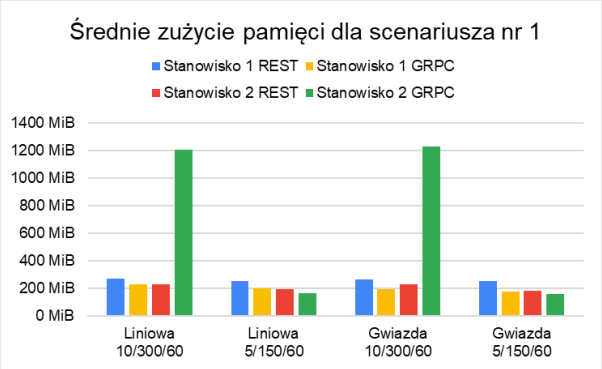
Rysunek 7: Średnia liczba żądań na sekundę dla scenariusza nr 2.

5.4. Analiza zużycia pamięci

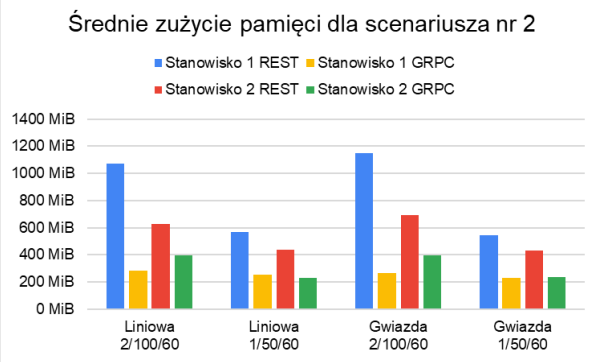
Na podstawie otrzymanych wyników dla obu topologii oraz stanowiska badawczego nr 1, widać niewielkie różnice w zużyciu pamięci pomiędzy REST i gRPC w przypadku scenariusza nr 1 (Rysunek 8). W obu przypadkach średnia pamięć waha się pomiędzy 175-270 MiB, lecz w każdej z przeprowadzonych symulacji gRPC osiąga średnio o 25% niższe wyniki.

W przypadku scenariusza nr 2 zauważalna jest przewaga gRPC, który podczas przesyłania i pobierania plików o większym rozmiarze wykorzystuje średnio około 250 MiB pamięci RAM, czyli około od 23% do 44% pamięci wykorzystywanej przez REST (Rysunek 9). W przypadku stanowiska nr 2 wyniki dla gRPC również wskazują na jego przewagę dla scenariusza nr 2, w którym średnie zużycie pamięci dla systemów wykorzystujących gRPC wynosi średnio od 40% do 60% zużycia pamięci systemów korzystających z REST. Znaczne

różnice występują natomiast w przypadku scenariusza nr 1, a konkretnie w symulacji 10/300/60, w której systemy oparte na gRPC osiągają od pięciokrotnie do sześciokrotnie większe wartości zużycia pamięci RAM niż systemy oparte na REST, wynoszących średnio 1.2 GB.



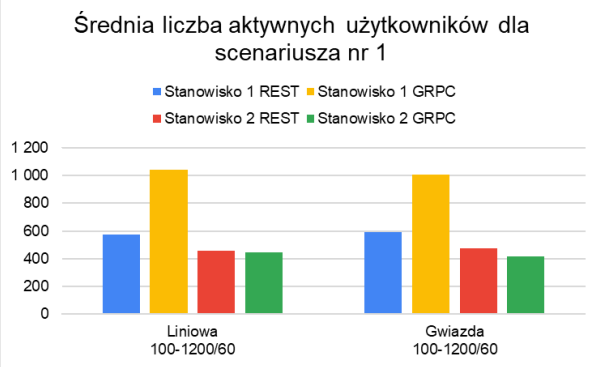
Rysunek 8: Średnie zużycie pamięci dla scenariusza nr 1.



Rysunek 9: Średnie zużycie pamięci dla scenariusza nr 2.

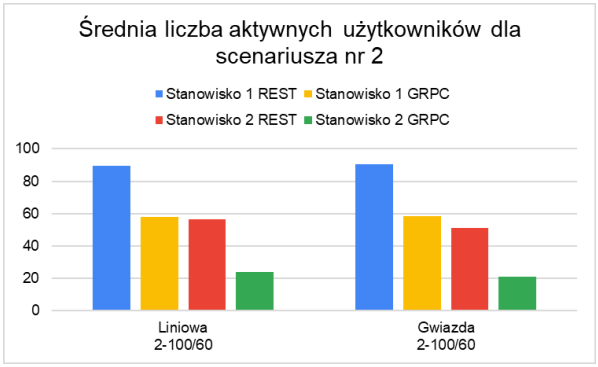
5.5. Analiza liczby aktywnych połączeń

Analizując wyniki średniej liczby aktywnych połączeń dla stanowiska nr 1 oraz scenariusza nr 1, przedstawione na Rysunku 10**,** widać znaczącą przewagę technologii gRPC nad technologią REST. Próg wydajności aplikacji korzystających technologii gRPC jest o 80% większy niż odpowiadających im aplikacji w technologii REST. Dla stanowiska nr 2 wyniki są odmienne, a mianowicie obie technologie komunikacji osiągają podobną wartość aktywnych połączeń, niezależnie od topologii, z nieznaczną przewagą REST nad gRPC.



Rysunek 10: Średnia liczba aktywnych użytkowników w momencie pierwszych problemów z przetwarzaniem kolejnych żądań dla scenariusza nr 1.

W przypadku wyników dla scenariusza nr 2, przedstawionych na Rysunku 11, różnice pomiędzy technologiami dla obu stanowisk oraz topologii wyglądają inaczej niż w przypadku scenariusza nr 1. W porównaniu z wynikami scenariusza nr 1 występuje tutaj odmienna sytuacja dla symulacji 2-100/60 wykonywanej w technologii REST, która osiąga o około 55% większą średnią liczbę aktywnych użytkowników dla stanowiska nr 1 oraz od 130% do 140% dla stanowiska nr 2, w stosunku do gRPC.



Rysunek 11: Średnia liczba aktywnych użytkowników w momencie pierwszych problemów z przetwarzaniem kolejnych żądań dla scenariusza nr 2.

5.6. Analiza skalowalności

W celu określenia skalowalności poszczególnych technologii komunikacji pod uwagę brane są metryki: użycie procesora i pamięci RAM, liczba żądań na sekundę oraz czas odpowiedzi dla poszczególnych stanowisk badawczych. Zestawienie ich na stanowiskach o różnych zasobach i mocy obliczeniowej pozwala na określenie, która z technologii skaluje się lepiej.

Analizując zużycie pamięci i procesora, zawartych na Rysunku 2 i Rysunku 3 oraz Rysunku 8 i Rysunku 9, można zauważyć istotne różnice między dwoma stanowiskami badawczymi. Na stanowisku o większej mocy obliczeniowej (stanowisko nr 1) widoczne jest zbliżone bądź mniejsze zużycie pamięci RAM oraz procesora w porównaniu do stanowiska nr 2, w każdym przypadku za wyjątkiem zużycia pamięci dla scenariusza nr 2. Na stanowisku nr 2 gRPC wykorzystuje znacznie więcej zasobów systemowych, mimo że nie przekłada się to na lepszą wydajność w porównaniu do REST. Z kolei na stanowisku nr 1 gRPC pomimo mniejszego zużycia zasobów w większości przypadków, wykazuje lepszą efektywność co można zaobserwować analizując średnią liczbę żądań zawartą na Rysunku 6 oraz Rysunku 7 oraz zestawiając je z procentowym stosunkiem poprawnych odpowiedzi zawartym w Tabeli 7. Jak można zauważyć, w przypadku obu scenariuszy, dostępność do większej ilości zasobów systemowych znacznie lepiej wpływała na wzrost efektywności gRPC niż REST. Dodatkowo biorąc pod uwagę średnie czasy odpowiedzi zawarte na Rysunku 4 oraz Rysunku 5, widoczna jest podobna tendencja, o czym świadczą znacznie mniejsze stosunki czasów otrzymywane dla stanowiska nr 1 względem stanowiska nr 2, praktycznie w każdej z przeprowadzonych symulacji, za wyjątkiem 5/150/60 w topologii gwiazdy.

Podsumowując te obserwacje można stwierdzić, że gRPC, dzięki swojej efektywności w zarządzaniu zasobami i wykorzystywaniu większej mocy obliczeniowej, może lepiej skalować się w pionie niż REST, zwłaszcza w środowiskach o dużych zasobach sprzętowych. REST, mimo że może być skuteczny w wielu scenariuszach, wydaje się mniej efektywny przy większym obciążeniu i bardziej ograniczony w skalowalności niż gRPC.

6. Wnioski

Otrzymane wyniki badań wskazują jednoznacznie na przewagę gRPC w większości z przeprowadzanych symulacji zarówno w skalowalności jak i wydajności, co częściowo potwierdza hipotezę nr 1, stwierdzającą, że gRPC osiągnie większą przepustowość niż REST z powodu korzystania z formatu binarnego. Podsumowując, gRPC ma znacznie większe możliwości odnośnie przepustowości, natomiast zależą one mocno od zasobów systemowych, na których uruchamiany jest dany system, a także rozmiarze danych przesyłanych na serwer, co jednocześnie przeczy hipotezie nr 2, gdyż zauważalne są znaczące różnice w skalowalności usług opartych na gRPC oraz REST, o czym świadczą lepsze wyniki na stanowisku badawczym o większej mocy obliczeniowej. Podsumowując wyniki zużycia zasobów dla aplikacji opartych na obu technologiach wynika, że hipoteza nr 3 również zostaje odrzucona, ze względu na zauważalne różnice w użyciu zasobów systemowych, a mianowicie: większe zużycie procesora w przypadku danych o większym rozmiarze dla gRPC oraz znacznie większe użycie pamięci RAM przez aplikacje oparte na REST w większości z przeprowadzanych symulacji. Badania przeprowadzane na dwóch stanowiskach badawczych w różnych topologiach pozwalają na stwierdzenie, że pomimo znacznych przewag gRPC w wydajności i skalowalności, nie powinien on być zawsze jedynym wyborem. Jak można zauważyć na podstawie wyników zużycia procesora dla scenariusza nr 2, w przypadku przesyłania plików o większym rozmiarze na stanowisku nr 2, REST radził sobie znacznie lepiej od gRPC. Oznacza to, że wybierając technologię komunikacji pomiędzy serwisami należy dokładnie wziąć pod uwagę ich zastosowanie, rozmiar przesyłanych danych, dostępne zasoby systemowe, a także schemat systemu, na którym oparta będzie komunikacja między serwisami. Oczywiście otrzymane wyniki tylko w pewnym procencie odpowiadają na pytania, które pojawiają się na etapie projektowania systemów, dlatego warto powtórzyć takie badania w różnych warunkach oraz specyficznych przypadkach występujących w obecnych systemach mikrousługowych.

Literatura

[1] M. Grambow, L. Meusel, E. Wittern, D. Bermbach, Benchmarking microservice performance: a pattern-based approach, SAC '20: Proceedings of the 35th Annual ACM Symposium on Applied Computing (2020) 232–241, https://doi.org/10.1145/3341105.3373875.

[2] A. Akbulut, H. G. Perros, Performance Analysis of Microservice Design Patterns, IEEE Internet Computing 23 (6) (2019) 19–27, https://doi.org/10.1109/MIC.2019.2951094.

[3] P. K. Kumar, R. Agarwal, R. Shivaprasad, D. Sitaram, S. Kalambur, Performance Characterization of Communication Protocols in Microservice Applications, In 2021 International Conference on Smart Applications, Communications and Networking (SmartNets) (2021) 1-5, https://doi.org/10.1109/SmartNets50376.2021.9555425.

[4] M. Samardžić, R. Šajina, N. Tanković, T. G. Grbac, Microservice Performance Degradation Correlation, In 43rd International Convention on Information, Communication and Electronic Technology (MIPRO) (2020) 1623-1626, https://doi.org/10.23919/MIPRO48935.2020.9245234.

[5] A. Avritzer, V. Ferme, A. Janes, B. Russo, A. van Hoorn, H. Schulz, D. Menasché, V. Rufino, Scalability Assessment of Microservice Architecture Deployment Configurations: A Domain-based Approach Leveraging Operational Profiles and Load Tests, Journal of Systems and Software 165 (2020) 110564-110579, https://doi.org/10.1016/j.jss.2020.110564.

[6] L. D. S. B Weerasinghe, I. Perera, Evaluating the Inter-Service Communication on Microservice Architecture, In 2022 7th International Conference on Information Technology Research (ICITR) (2022) 1-6, https://doi.org/10.1109/ICITR57877.2022.9992918.

[7] Introduction to gRPC, https://www.baeldung.com/grpc-introduction, [04.08.2024].

[8] Spring Framework Integration Documentation, https://docs.spring.io/spring-framework/reference/integration.html, [04.08.2024].

[9] Prometheus Documentation, https://prometheus.io/docs/introduction/overview/, [04.08.2024].

[10] Grafana Documentation, https://grafana.com/docs/grafana/latest/datasources/prometheus/, [04.08.2024].

[11] REST with Spring Tutorial, https://www.baeldung.com/rest-with-spring-series, [04.08.2024].

[12] Gatling Documentation, https://gatling.io/docs/gatling/ [04.08.2024]