Systemy rozproszone

Porównanie wydajności i efektywności komunikacyjnej technologii middleware

Albert Gierlach

03.06.2021

1. Zadanie

Celem zadania jest porównanie wydajności (czas wykonania) i efektywności komunikacji (liczba bajtów na poziomie L3) omawianych technologii middleware dla porównywalnych danych (podobieństwo interfejsów IDL) i sposobu implementacji strony serwerowej. Testy należy wykonać dla kilku (trzech) jakościowo różnych struktur danych i różnych wielkości samych danych (tj. np. sekwencja o długości 1 i 100 elementów) oraz dla różnych języków programowania. Dla uproszczenia realizacji wszystkie operacje/procedury powinny zwracać wartość pustą (void). Eksperymenty należy wykonać w taki sposób, by wyeliminować/uwzględnić wartości odstające (outlier).

2. Środowisko testowe

Do testów wykorzystane zostały następujące technologie:

- Java (część serwerowa)
- Python (część kliencka)

Do testów zostały przygotowane następujące struktury:

- Mała
  - 2x int
  - 1x string
  - 1x bool
  - 1x sekwencja wartości typu int
- Średnia
  - 4x int
  - 2x string
  - 2x bool
  - 2x double
  - 1x sekwencja wartości typu int
  - 1x sekwencja wartości typu string
- Duża
  - 10x int
  - 8x string

- 5x bool
- 5x double
- 2x sekwencja wartości typu int
- 2x sekwencja wartości typu string
- 2x sekwencja wartości typu double

Dodatkowo wykonano testy dla różnych długości sekwencji w strukturach:

- 5 elementów
- 100 elementów
- 300 elementów

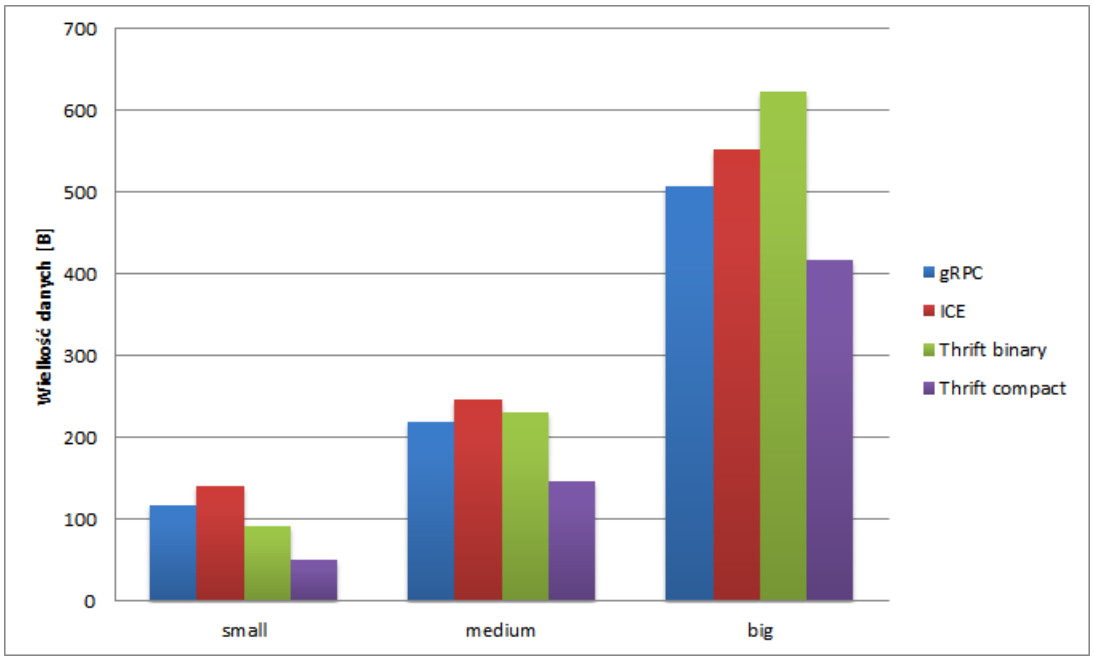
Testy zostały wykonane dla dwóch środowisk:

- lokalnie (na tym samym komputerze)
- w sieci LAN

Dla każdego przypadku testowego wykonano 1000 prób, następnie odrzucono wartości odstające, a z pozostałych wartości obliczono wartość średnią. Dane w kolejnych próbach generowane są losowo. Dla zapewnienia tych samych danych dla każdej z technologii użyto tego samego *seed* dla funkcji *random*.

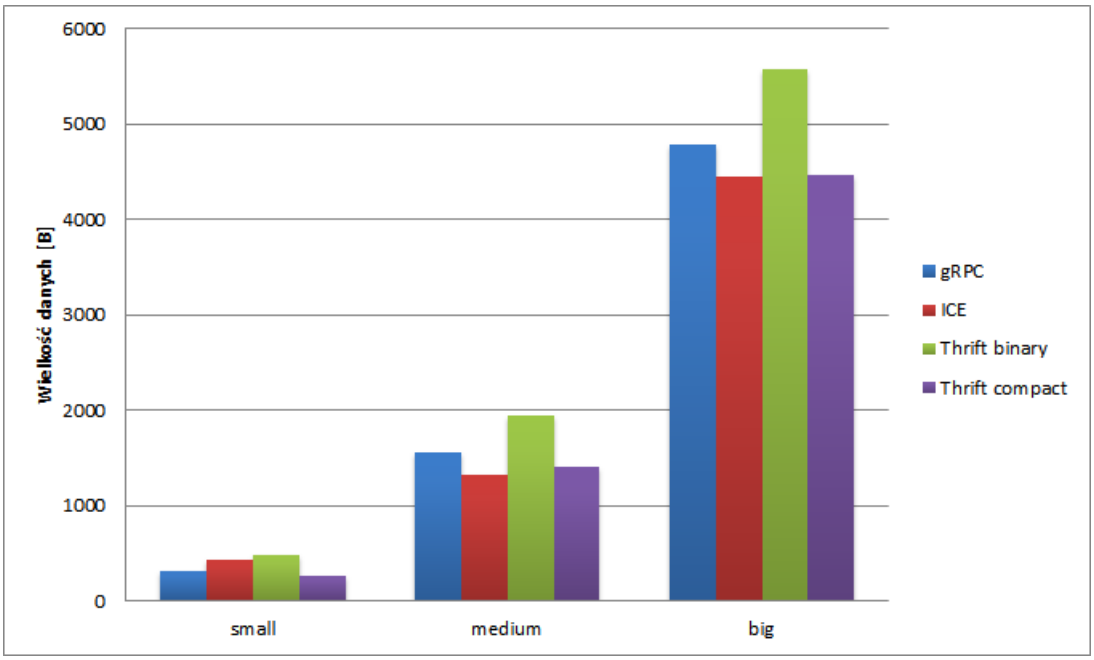
3. Efektywność komunikacji na poziomie L3

Dla każdego przypadku testowego została sprawdzona liczba bajtów przesyłana na poziomie L3. Z zebranych danych utworzono następujące wykresy:

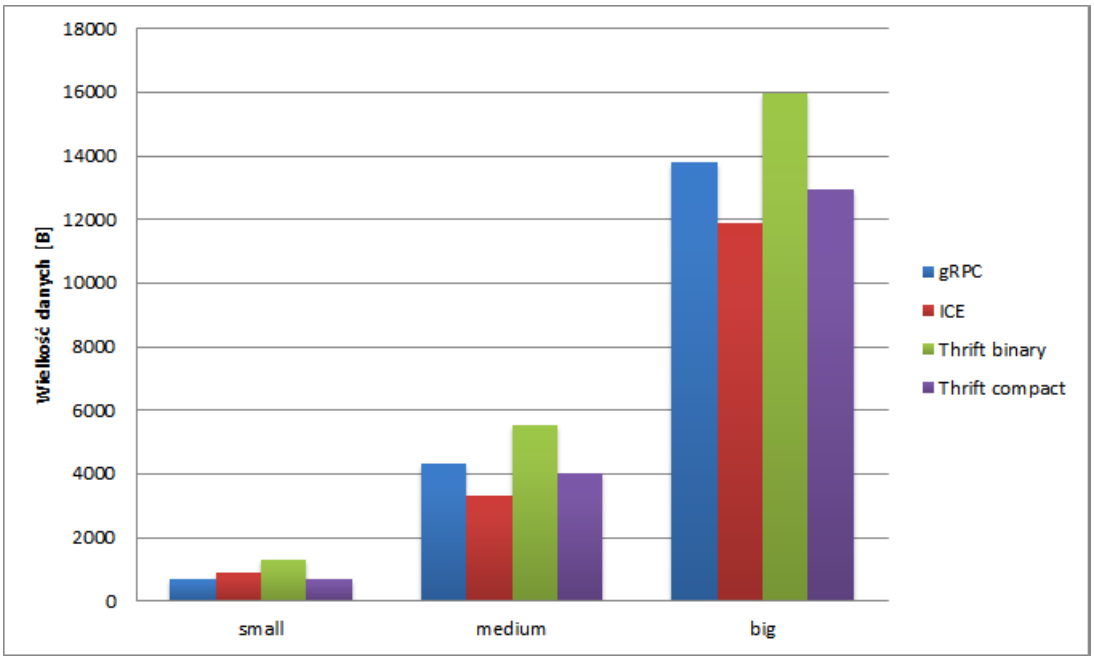


Rysunek 1: Sekwencje o długości 5

Dla małych struktur serializacja Thrift Compact daje najlepsze rezultaty (najmniej bajtów). Thrift binary przekracza granicę 600B dla dużej struktury, jednak wszystkie metody serializacji mają porównywalną skuteczność serializacji.



Rysunek 2: Sekwencje o długości 100

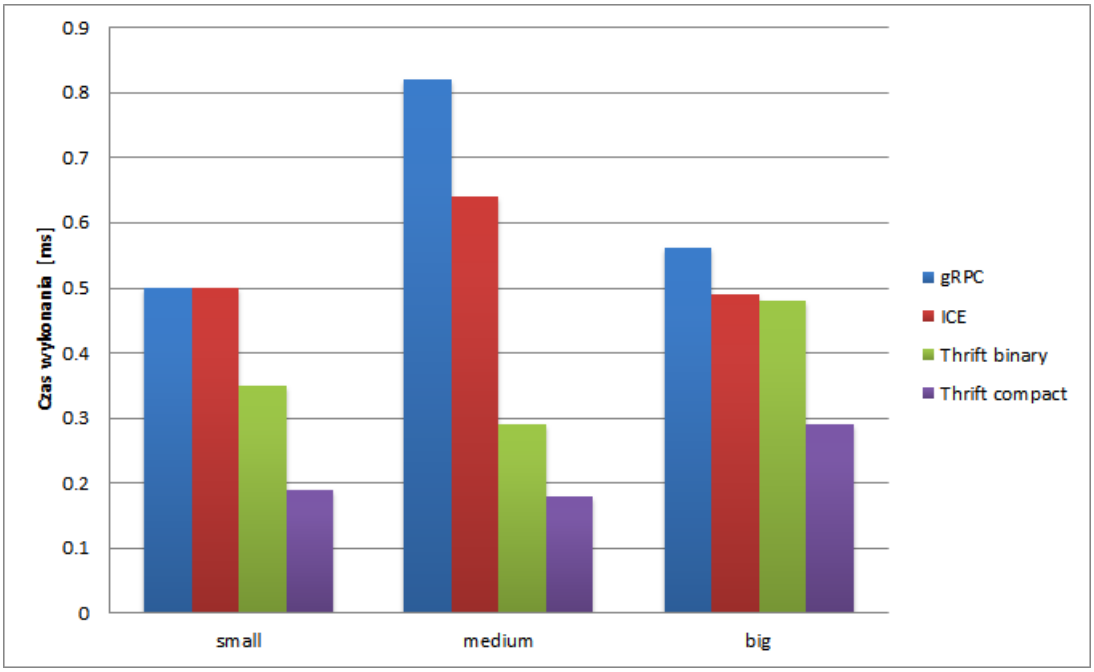


Rysunek 3: Sekwencje o długości 300

Dla sekwencji o większej długości widzimy pewien trend - Thrift binary jest najmniej oszczędny w bajtach. Natomiast serializacja ICE daje najlepsze rezultaty. Serializacja gRPC oraz Thrift Compact są bardzo zbliżone do siebie pod względem objętości danych.

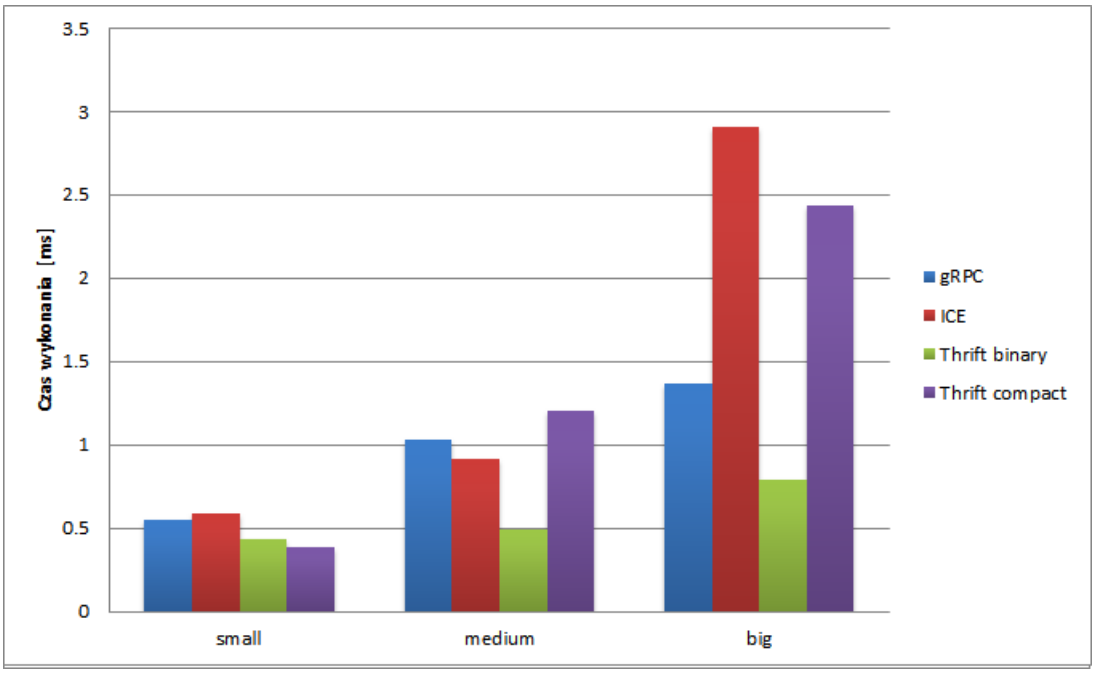
4. Czasy wykonania

4.1. localhost

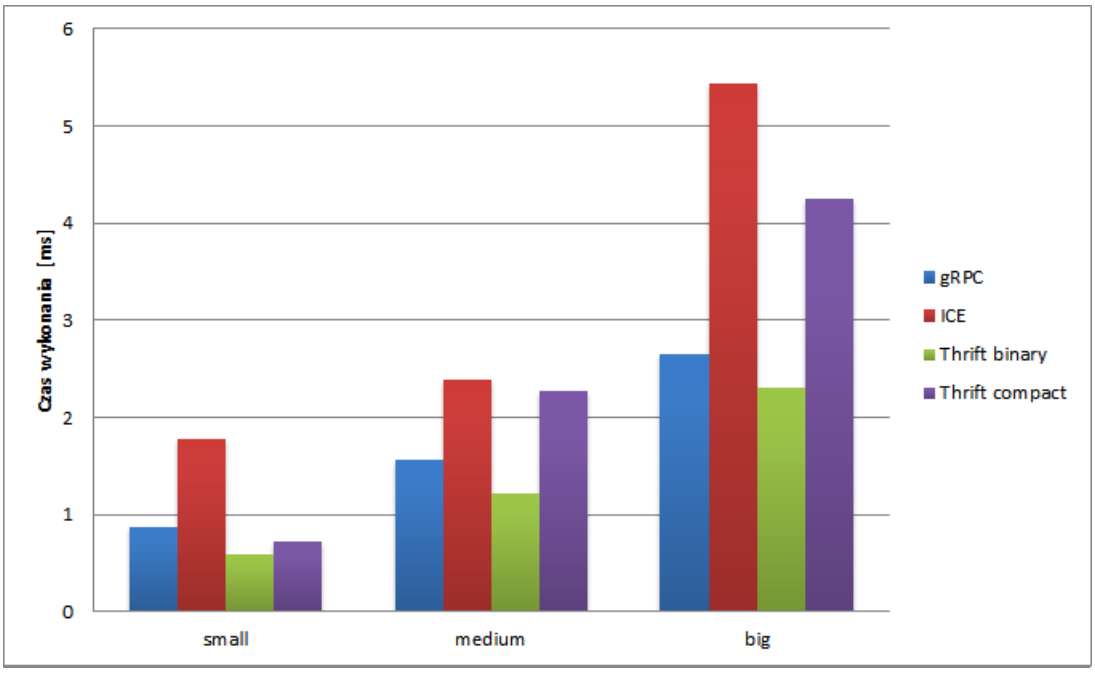


Rysunek 4: Sekwencje o długości 5

Dla krótkich sekwencji można zauważyć, że gRPC oraz ICE mają najdłuższy czas wykonania. Najszybszy okazuje się być Thrift compact, a tuż za nim Thrift binary.



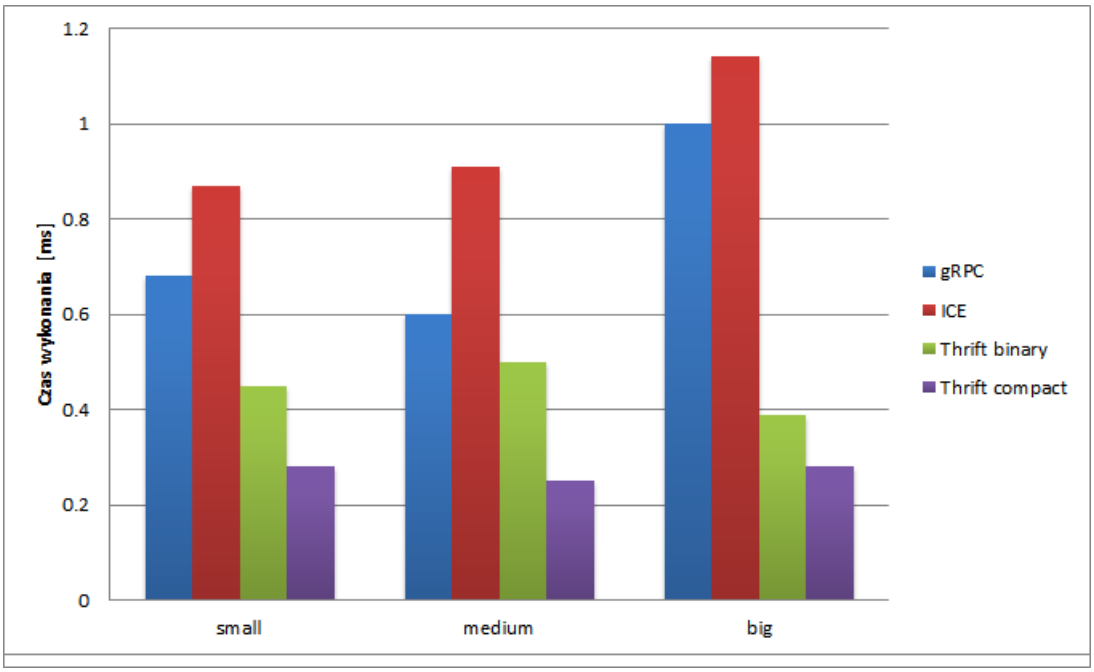
Rysunek 5: Sekwencje o długości 100



Rysunek 6: Sekwencje o długości 300

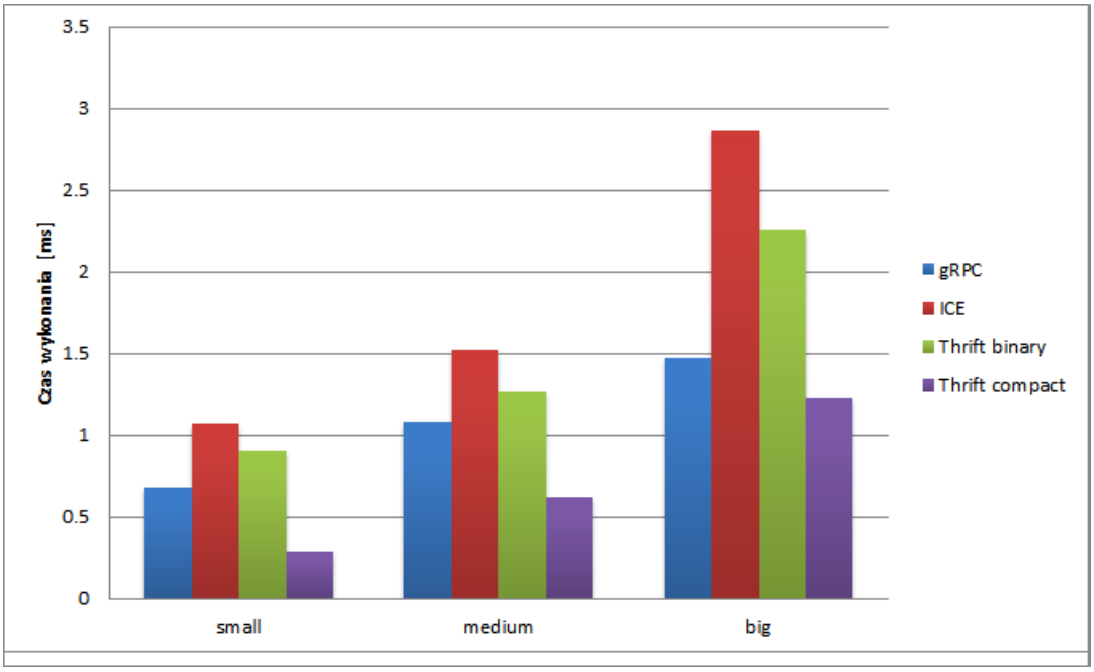
Gdy sekwencje się wydłużają, czyli ilość danych do przesłania oraz serializacji rośnie, obserwujemy sporą różnicę w wydajności technologii ICE oraz Thrift compact w porównaniu do pozostałych - narzut czasowy jest największy. Najlepszym pod względem opóźnień okazuje się być Thrift binary, który zdecydowanie jest najszybszy pod względem czasu wykonania. gRPC w kilku przypadkach jest bardziej zbliżony do Thrift binary niż do np. ICE.

4.2. LAN

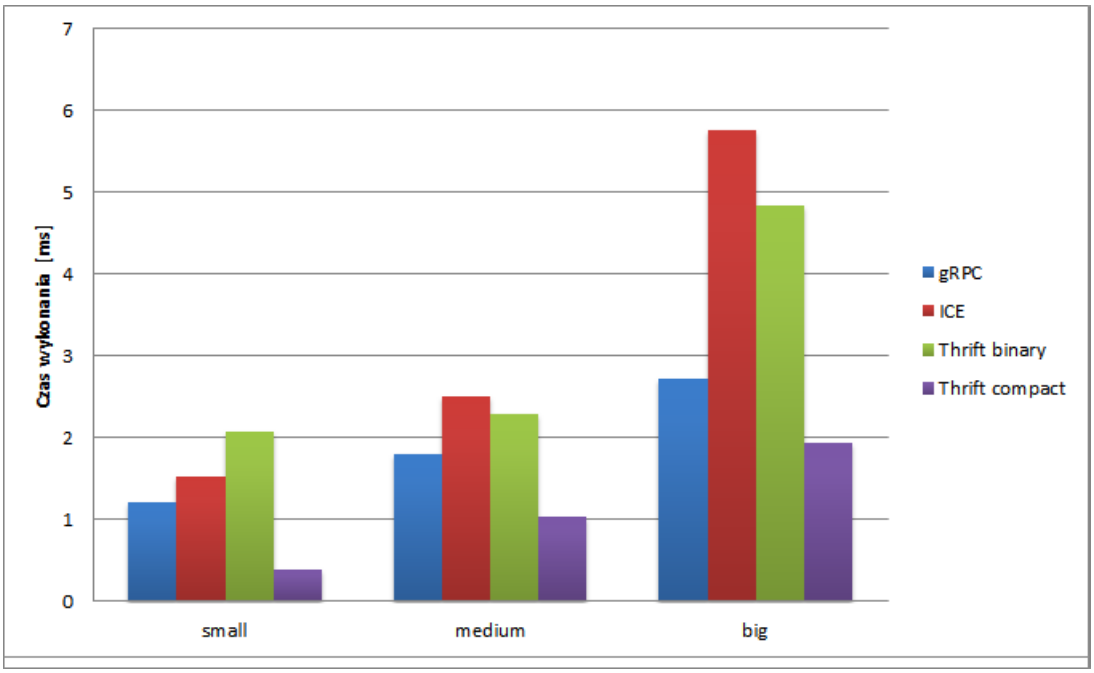


Rysunek 7: Sekwencje o długości 5

Dla struktur o mniejszych rozmiarach, po raz kolejny obie serializacje Thrift dają najlepsze rezultaty. Technologia ICE wypada w tym zestawieniu najgorzej, jednak gRPC osiąga tylko niewiele lepsze wyniki.



Rysunek 8: Sekwencje o długości 100



Rysunek 9: Sekwencje o długości 300

Po raz kolejny Thrift compact wygrywa z pozostałymi technologiami osiągając najkrótsze czasy. Najwolniejszy okazuje się ICE, a zaraz po nim plasuje się gRPC. W przypadku sekwencji o długości 300 elementów Thrift compact jest ponad dwa razy szybszy nić ICE oraz Thrift Binary.

5. Wnioski

- Najlepszą (pod względem ilości bajtów) serializację pozwala osiągnąć technologia ICE. Niezależnie od danych zawsze była w czołówce
- Najmniej oszczędnym w bajty Thrift binary, który w każdym zestawieniu wypadał najsłabiej
- Thrift compact osiąga znakomite rezultaty, gdy dane są przesyłane przez sieć. Zysk z dobrej serializacji i kompresji jest tu kluczowy, aby zmniejszyć liczbę danych przesyłanych w sieci
- W przypadku localhost'a najważniejszym czynnikiem jest szybkość serializacji i deserializacji, ponieważ narzut sieciowy jest dość niski
- gRCP niemalże w każdym zestawieniu plasuje się na drugim lub trzecim miejscu, co czyni go dość uniwersalnym rozwiązaniem
- ICE nie wypadł dobrze w porównaniach - słaba serializacja oraz długie czasy wykonania
- Duża entropia danych może znacząco wpływać na wyniki