Porównanie sposobów serializacji Thrift

SYSTEMY ROZPROSZONE

DAWID BIAŁKA

nr Indeksu: 305276

1. Sposób przeprowadzania testów

W celu porównywania czasów działania danych protokołów serializacji dostępnych w Thrift użyte zostały trzy języki programowania – Java, Python i Ruby. Pomiar przesyłanych pakietów odbywał się z wykorzystaniem programu Wireshark. Proces serializacji był wykonywany za każdym razem tysiąc razy i dla tych wszystkich procesów był liczony łączny czas. Wynik następnie był dzielony przez tysiąc, aby otrzymać czas trwania pojedynczego procesu serializacji. Czas wysyłania struktur nie był brany pod uwagę. Wyniki eksperymentu są podzielone ze względu na języki programowania. Zostały przetestowane protokoły TBinaryProtocol, TJSONProtocol i TCompactProtocol.

2. Wykorzystane struktury

- SimpleStruct – wykonane trzy testy

```
struct SimpleStruct
{
    1:i32 integer,
    2:double decimal,
    3:string text
}
```

-CollectionStruct – wykonane trzy testy dla każdego z przypadków: 10 elementów w liście i zbiorze oraz dla 100 elementów

```
struct CollectionStruct
{
    1:list<i64> numbers,
    2:set<double> decimals
}
```

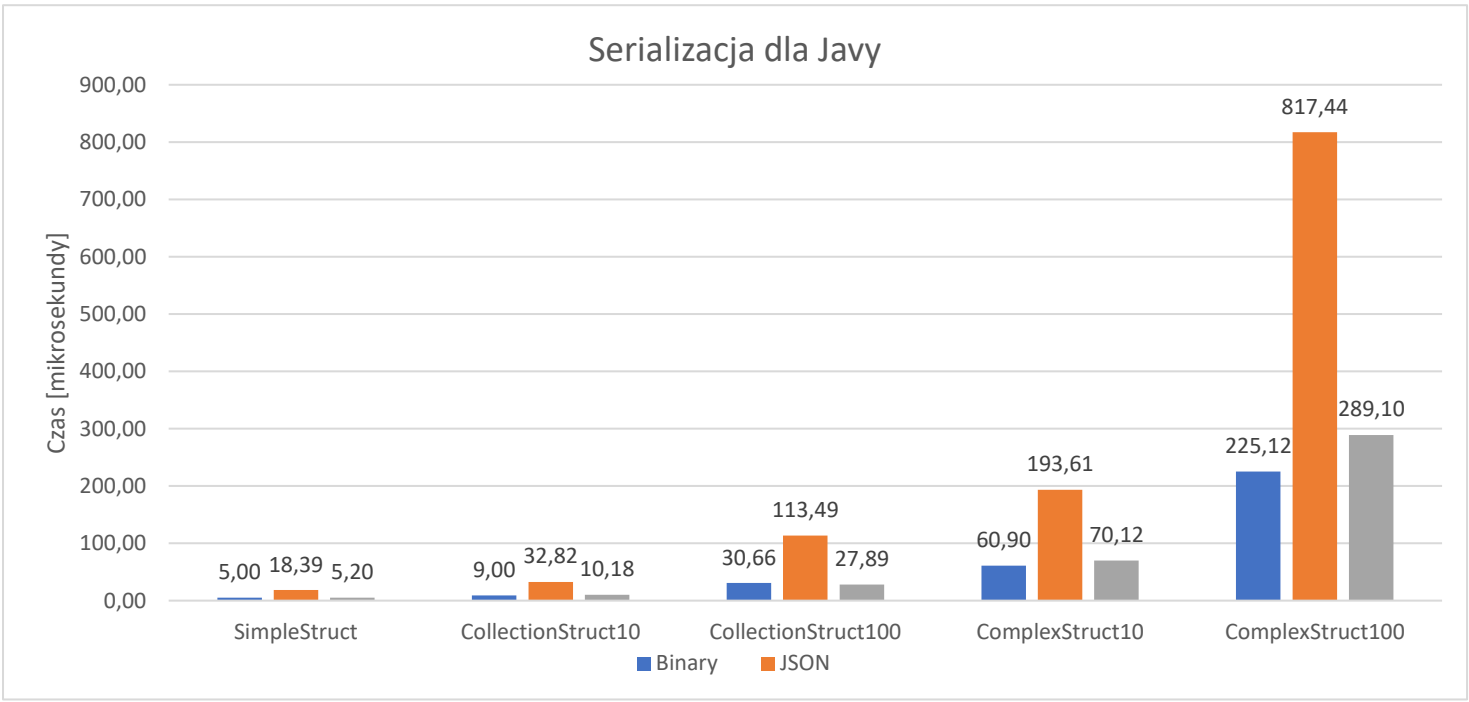
-NestedStruct (struktura pomocnicza)

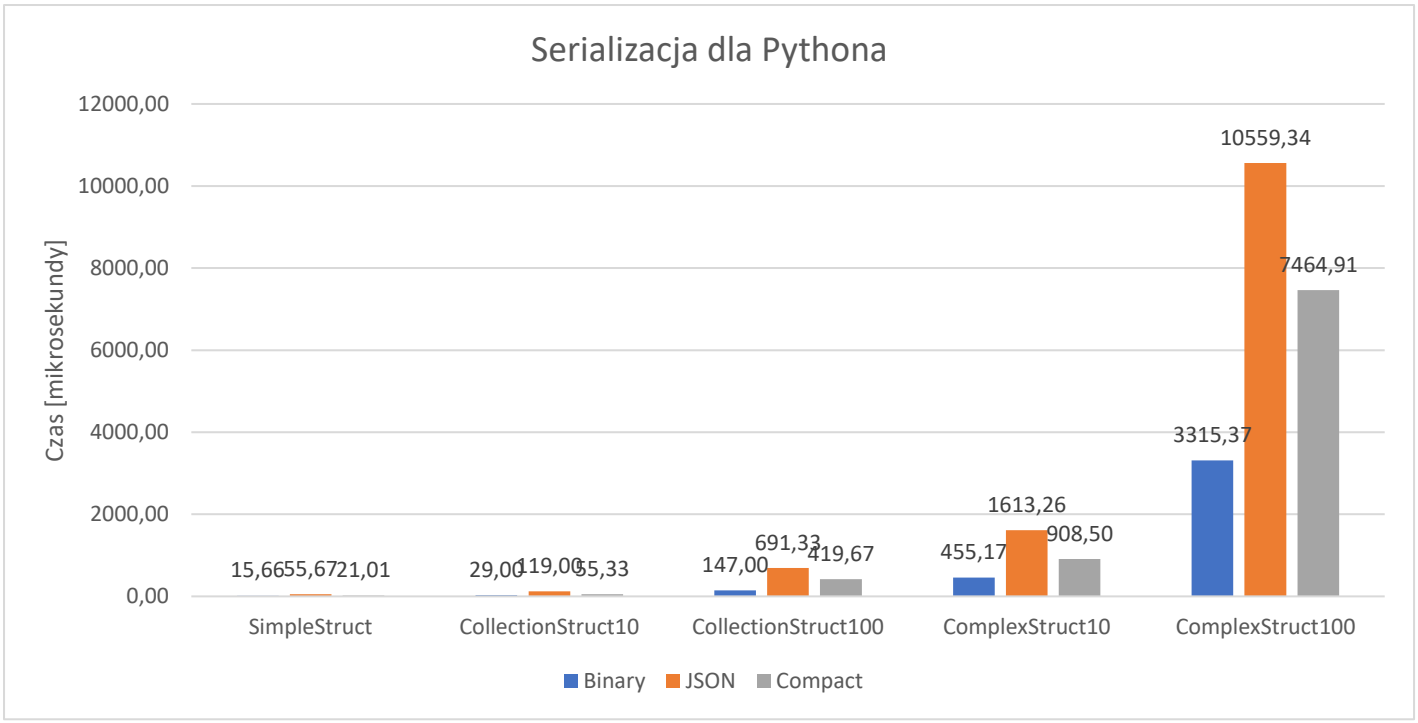
```
struct NestedStruct
{
    1:list<string> textList,
    2:map<i32, double> integerToDouble
}
```

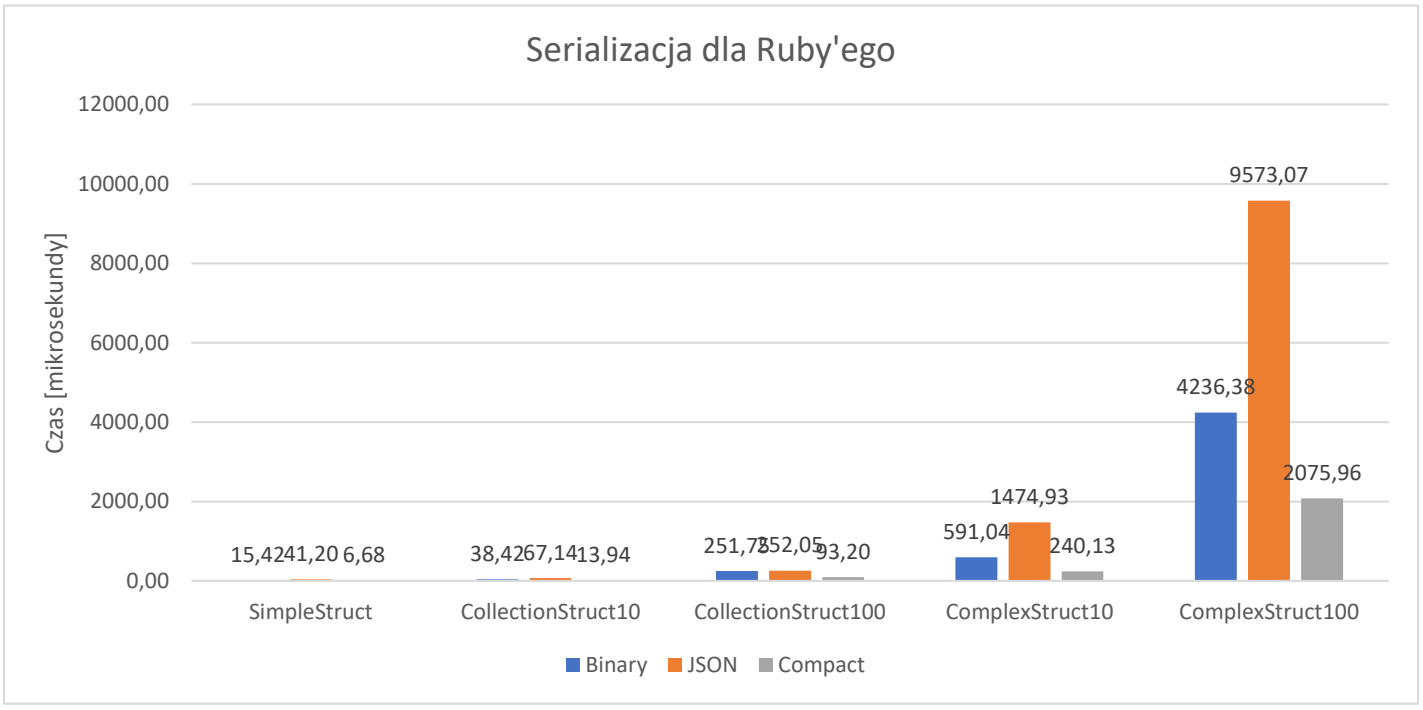
-ComplexStruct – wykonano trzy testy dla każdego z przypadków: 10 elementów w liście i zbiorze oraz 100 elementów w strukturze NestedStruct. Za każdym razem takich obiektów w liście nestedStructList i integerToNestedStruct było pięć.

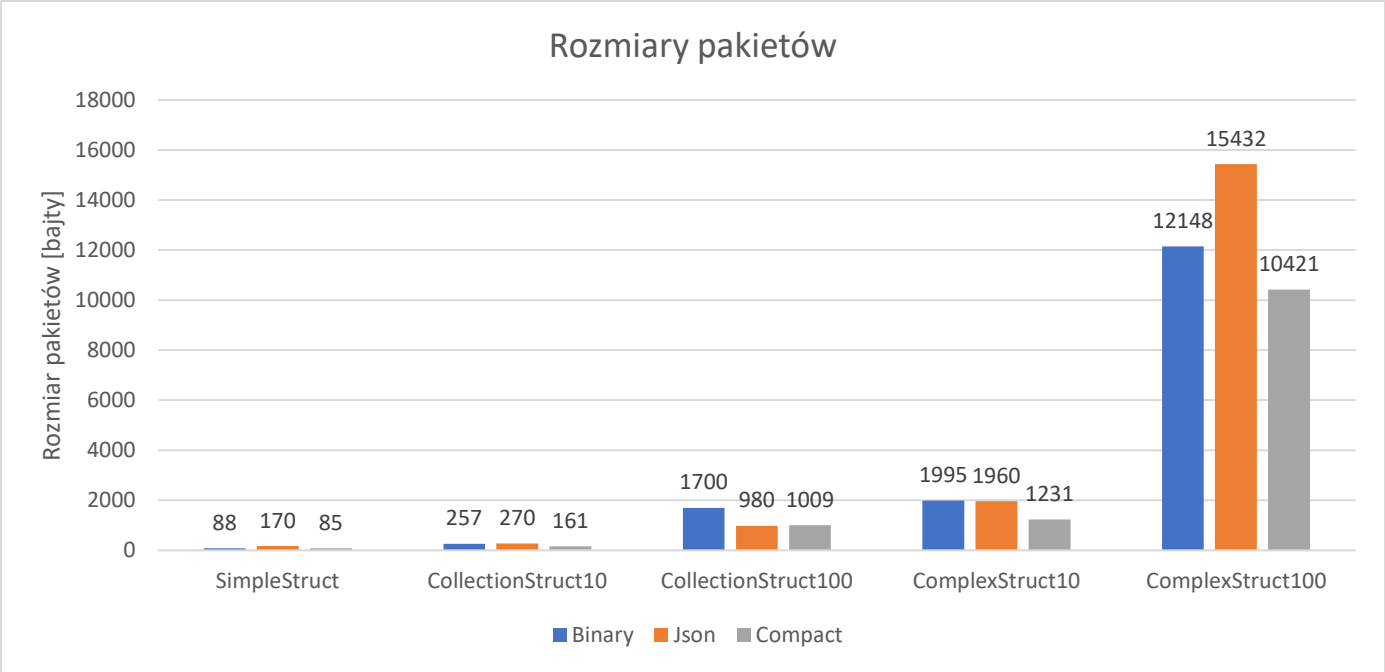
```
struct ComplexStruct
{
    1:list<NestedStruct> nestedStructList,
    2:map<i32, NestedStruct> integerToNestedStruct
}
```

3. Wyniki testów




Serializacja dla Pythona
Czas [mikrosekundy]
12000,00
10000,00
8000,00
6000,00
4000,00
2000,00
0,00
15,66
55,67
21,01
29,00
119,00
35,33
147,00
691,33
419,67
455,17
1613,26
908,50
3315,37
10559,34
7464,91
SimpleStruct
CollectionStruct10
CollectionStruct100
ComplexStruct10
ComplexStruct100
Binary
JSON
Compact


Serializacja dla Ruby'ego
Czas [mikrosekundy]
12000,00
10000,00
8000,00
6000,00
4000,00
2000,00
0,00
15,42
41,20
6,68
38,42
67,14
13,94
251,78
52,05
93,20
591,04
1474,93
240,13
4236,38
9573,07
2075,96
SimpleStruct
CollectionStruct10
CollectionStruct100
ComplexStruct10
ComplexStruct100
Binary
JSON
Compact



4. Wnioski

Dla każdego języka czas procesu serializacji był najdłuższy dla protokołu TJSONProtocol. W przypadku Python'a protokół TCompactProtocol był wolniejszy od TBinaryProtocol. Dla Ruby'ego było na odwrót. Protokół TBinaryProtocol dawał najlepsze wyniki zarówno dla Javy jak i dla Pythona, tylko w przypadku Ruby'ego nie był on najlepszy.

Protokół TCompactProtocol zawsze tworzył najmniejsze pakiety. Dla większych struktur wydajność TJSONProtocol była coraz gorsza w porównaniu do reszty.

Jakie negatywne strony ma użycie serializacji oferującej większą efektywność komunikacyjną (jeśli ma)

Serializacja, która ma większą efektywność komunikacyjną, np. TCompactProtocol wytwarza znacznie mniejsze pakiety niż pozostałe protokoły, zatem jest to pozytywna cecha. Jednak pakiety te mogą zostać odczytane tylko przez ten sam protokół po stronie odbiorcy. W przypadku TJSONProtocol pakiety mają znacznie większy rozmiar, jednak gdy ważne jest dla nas czytelność przesyłanych danych to ten protokół jest zalecanym wyborem.