Wat is de snelste manier van het berichten serialisatie & de-serialisatie

Voor ons project willen wij iets beters gebruiken dan Json.
Om dit te onderzoeken en uit elkaar te trekken voor benchmarks moeten we eerst weten wat de meest voorkomende manieren zijn om dit voor elkaar de krijgen hiervoor hebben we drie mogelijkheden.

1. Json
2. Messagepack
3. Protobuf

Deze drie worden het meest gebruikt op het moment van dit onderzoek.

Hiervoor houden we drie tests met verschillende berichten groottes.
gelukkig hebben we al mooie grafieken voor door al gemaakte tests.

Test 1: Klein bericht

serializer	size (bytes)	serialization time (µs)	deserialization time (µs)
json	74	4.02	4.45
protobuf	24	21	12.1
messagepack	48	4.48	0.912

Figuur 1: Resultaten 1ste test (Silva, 2018)

	number of requests	median response (ms)	average response (ms)
small json	173,592	11	28
small protobuf	170,625	21	43
small messagepack	170,936	17	41

Figuur 2: Resultaten 1ste test response time (Silva, 2018)

Uit deze resultaten is er niet echt duidelijk te zien of er een nauw beter is dan de andere
daarom gaan we meerder test houden met verschillende bericht groottes.

Test 2: Medium bericht

message type	size (bytes)	serialization time (µs)	deserialization time (µs)
json	36,747	303	892
protobuf	22,160	8540	3940
messagepack	30,797	866	202

Figuur 3: Resultaten 2de test (Silva, 2018)

	number of requests	median response (ms)	average response (ms)
big json	146,927	170	213
big protobuf	161,907	71	96
big messagepack	152,764	130	156

Figuur 4: Resultaten 2de test response time (Silva, 2018)

We kunnen nu zien dat er verschil is in de serialisatie tijden zijn en dat de gemiddelde tijden wat verder uit elkaar liggen.
voor alle zekerheid houden we nog een test met een groot bericht.

Test 3: Groot Bericht

message type	size (bytes)	serialization time (ms)	deserialization time (ms)
json	105,770	0.971	2.63
protobuf	63,804	31.7	13
messagepack	88,684	2.53	0.614

Figuur 5: Resultaten 3de test (Silva, 2018)

	# of requests	# failed requests	median response (ms)	average response (ms)
3.1.a JSON	122,749	35,998	81	118
3.1.b Protobuf	131,556	44,024	10	16
3.1.c MessagePack	128,954	46,026	11	19

Figuur 6: Resultaten 3de test Response time (Silva, 2018)

	# of requests	# failed requests	median response (ms)	average response (ms)
3.2.a JSON	57,776	60	1600	2003
3.2.b Protobuf	122,733	0	510	458
3.2.c MessagePack	143,030	0	280	250

Figuur 7: Resultaten 3de test met aantal failed requests (Silva, 2018)

Bij grootte berichten gaat Json al snel “stuk” aangezien er 60 verzoeken van de 57,776 verzoeken gefaald waarbij de andere twee geen enkele keer gefaald hebben en nog aardig snel waren.

Maar omdat wij geen validatie nodig hebben wij gekozen voor Messagepack omdat het ook kleiner is en daarom ook minder ruimte in neemt.

Bronnen:
https://medium.com/unbabel/the-need-for-speed-experimenting-with-message-serialization-93d7562b16e4 (Silva, 2018)