



**FCTUC** FACULDADE DE CIÊNCIAS  
E TECNOLOGIA  
UNIVERSIDADE DE COIMBRA

# Data Representation and Serialization Formats

## Integração de Sistemas

# 1- Introdução

Neste trabalho vai ser comparado o uso de XML vs Message Pack.

Vão ser comparados os tempos de serialização e de deserialização assim como o tamanho final do resultado após a serialização.

No final pretende-se que se possa entender quando usar XML e quando usar o Message Pack.

## 2- Metodologia

Para este projeto foi nos definida uma estrutura de dados pet-owner many to one para as quais foram criadas classes em Java. Cada owner tem um certo número de animais que foram colocados numa Array List e associados ao owner respetivo. Cada owner é por fim adicionado a outra Array List que será serializada nos dois formatos escolhidos.

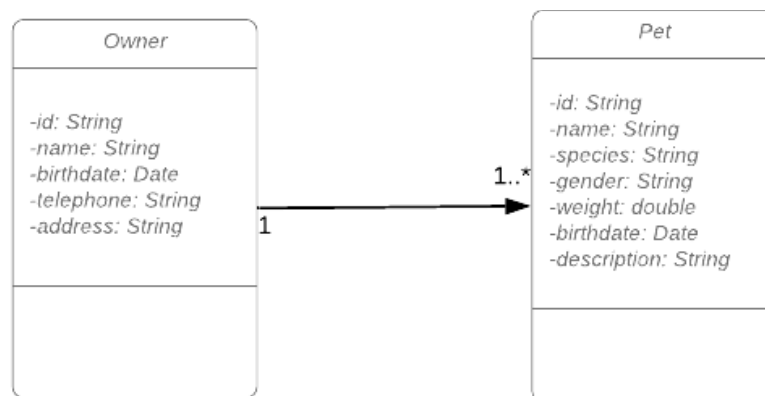


Figura 1: UML das classes Owner e Pet

Os formatos seleccionados foram Message Pack(binary format) e XML(text format) que serão comparados ao longo da fase de testes.

```
// Serialize
long startTimeSerialize = System.currentTimeMillis();
JAXBContext contextObj = JAXBContext.newInstance(Owners.class);
Marshaller marshallerObj = contextObj.createMarshaller();
marshallerObj.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, true);
marshallerObj.marshal(owners1, fso);
long endTimeSerialize = System.currentTimeMillis();

// Deserialize
long startTimeDeserialize = System.currentTimeMillis();
JAXBContext contextObj1 = JAXBContext.newInstance(Owners.class);
Unmarshaller unmarshaller = contextObj1.createUnmarshaller();
Owners owners2 = (Owners) unmarshaller.unmarshal(fr);
long endTimeDeserialize = System.currentTimeMillis();
```

```
// Serialize
long startTimeSerialize = System.currentTimeMillis();
byte[] bytes = msgpack.write(owners);
outputStream.write(bytes);
long endTimeSerialize = System.currentTimeMillis();

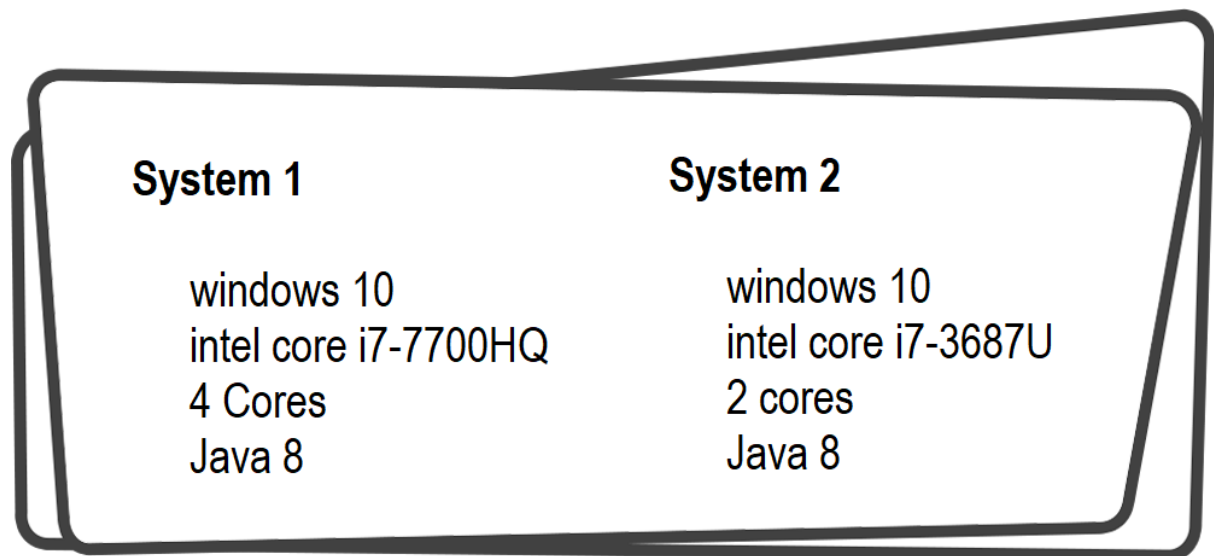
// Deserialize
long startTimeDeserialize = System.currentTimeMillis();
byte[] bytes1 = Files.readAllBytes(Paths.get("first: outputs/message.bin"));
Owner[] dst = msgpack.read(bytes1, Owner[].class);
long endTimeDeserialize = System.currentTimeMillis();
```

Figura 2: Código XML(esquerda) e Message Pack(direita)

Para cada formato foram realizados testes com doze ficheiros e para cada um destes ficheiros foram medidos dez tempos para serialização e dez tempos para a deserialização. Entre os ficheiros utilizados varia o número de owners e/ou pets de forma avaliar como cresce o tempo de execução.

Para além dos tempos foi retirado também os tamanhos dos ficheiros criados.

Estes valores foram retirados de dois sistemas diferentes com as correspondentes versões de Java:



<b>System 1</b>	<b>System 2</b>
windows 10	windows 10
intel core i7-7700HQ	intel core i7-3687U
4 Cores	2 cores
Java 8	Java 8

Figura 3: Specs dos sistemas usados + Java versions

### 3- Resultados

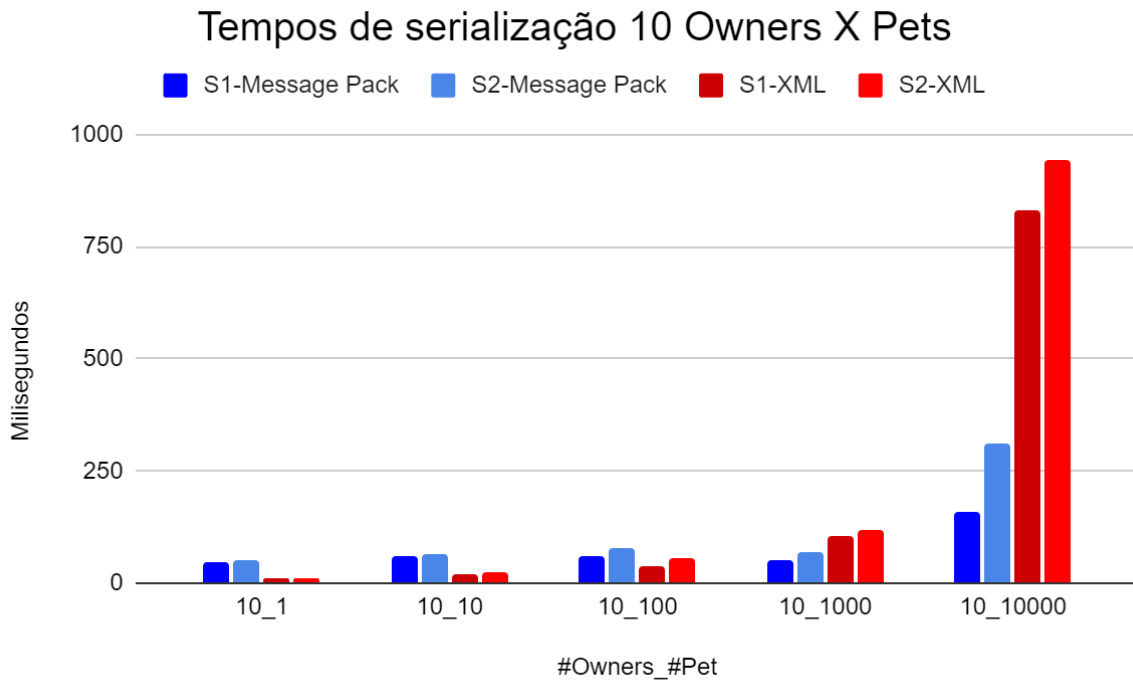


Gráfico 1: Tempos médios de serialização para 10 owners e um certo número de pets para os dois sistemas(S1 e S2).

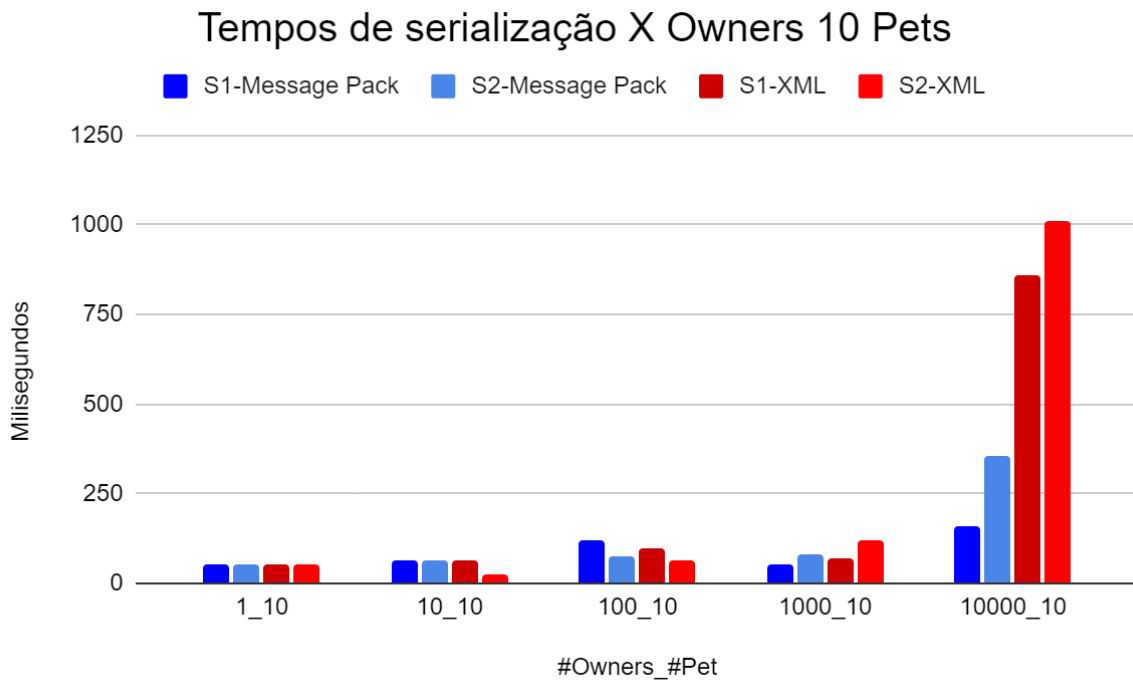


Gráfico 2: Tempos médios de serialização para 10 pets e um certo número de owners para os dois sistemas(S1 e S2).

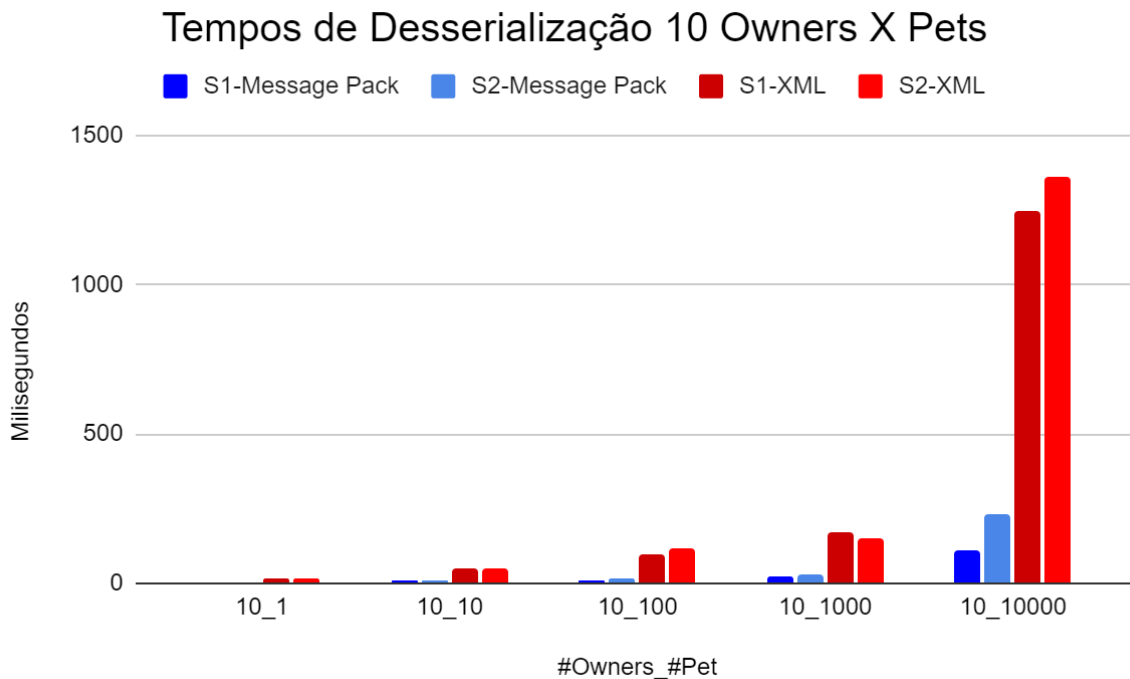


Gráfico 3: Tempos médios de desserialização para 10 owners e um certo número de pets para os dois sistemas(S1 e S2).

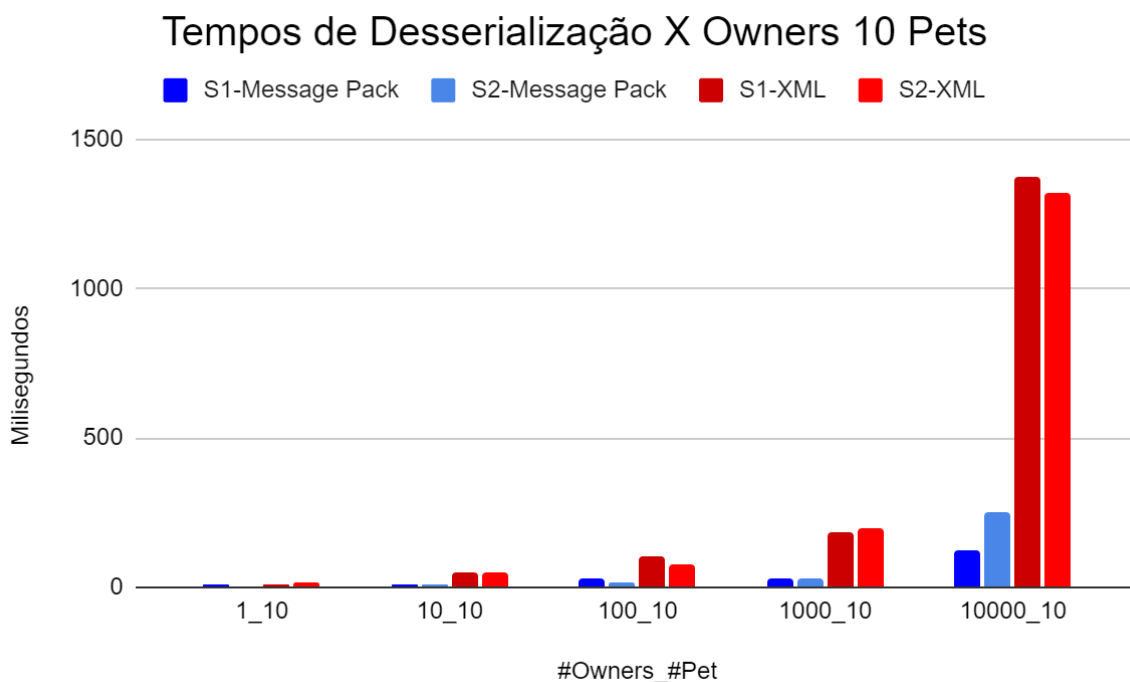


Gráfico 4: Tempos médios de desserialização para 10 pets e um certo número de owners para os dois sistemas(S1 e S2).

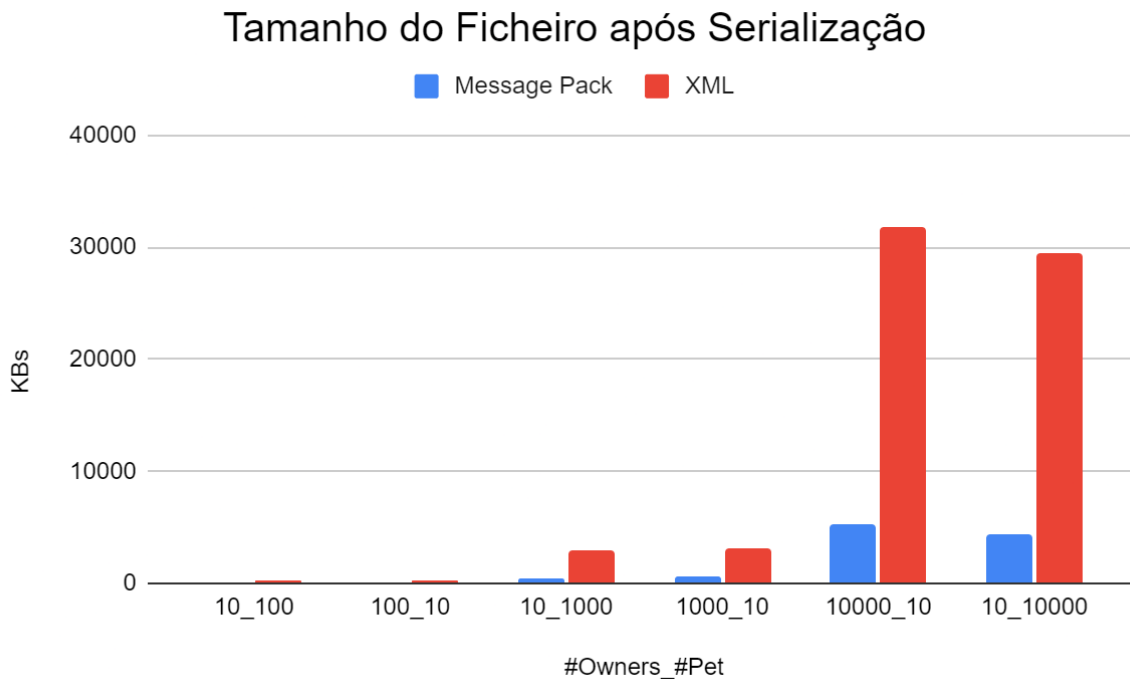


Gráfico 5: Tamanho final do ficheiro serializado.

## 4- Discussão

Como é possível verificar pelos gráficos 1 e 2, os tempos de serialização para ficheiros com número elevado de *Owners* ou *Pets* são bastante mais elevados no XML quando comparado ao Message Pack. Já para ficheiros pequenos essa diferença é mínima, tendo o XML até uma certa vantagem, como se repara no gráfico 1 para o ficheiro 10\_1.

Observando os gráficos 3 e 4, verifica-se que na desserialização o Message Pack é sempre mais rápido que o XML, sendo irrelevante o tamanho dos dados. Observa-se de igual forma que os tempos de desserialização do XML são maiores que na serialização, e que tal não acontece no Message Pack.

Em ambos os formatos os tempos de serialização/desserialização aumentam mais quando se aumenta o número de *Owners* em vez do número de *pets*.

Por fim, e de forma evidente pelo gráfico 5, os tamanhos dos ficheiros binários gerados pelo Message Pack são incrivelmente inferiores ao XML.

Estes resultados concordam com o que era esperado, uma vez que o Message Pack se trata de um *Binary Format* enquanto o XML de um *Text Format*. O XML precisa de *tags*, *elements* e *attributes* que não só aumentam o tamanho do ficheiro como aumenta a carga do CPU, uma

vez que precisa de *parsing*, demorando mais a escrever e a ler. O Message Pack escreve dados diretamente nos ficheiros sem separações uma vez que muitos dos valores têm tamanho constante, usando elementos binários especiais para sinalizar, fazendo não só a escrita como a leitura mais rápida e ocupando menos espaço em disco.

Relativamente à complexidade programática, o XML é mais fácil de programar e de fazer debug, uma vez que é possível visualizar os dados de maneira organizada.

## 5- Conclusão

Ao longo deste relatório, foi avaliado as diferenças de tempo, complexidade e tamanho do ficheiro entre um formato binário e um formato de texto, mais especificamente, XML e Message Pack.

Com os resultados obtidos, verifica-se que para dados muito elevados o Message Pack é o vencedor, tanto em velocidade como em espaço, mas vem com um lado negativo de não ser *user friendly* para quem programa, uma vez que não é possível ir ao ficheiro verificar se ficou tudo correto. Caso o número de dados seja baixo, tanto o XML como o Message Pack têm um comportamento semelhante.

Concluindo, em caso de ser preciso realizar uma escolha deve-se optar pelos formatos binários já que este tem uma performance (tempo e espaço) melhor que o formato de texto. No entanto, em volumes de dados pequenos deve-se utilizar o formato de texto porque para além de ser mais fácil de utilizar/programar também oferece uma melhor representação visual de dados no ficheiro.

## 6- Referências

Al Williams, 2020, accessed 8 of October 2021,  
<https://hackaday.com/2020/03/12/MessagePack-is-a-more-efficient-json/>

Marshal Cline, 2012, accessed 8 of October 2021,  
<https://www.cs.technion.ac.il/users/yechiel/c++-faq/serialize-decide-text-vs-binary.html>

Daniel Mošmondor, 2010, accessed 8 of October 2021,  
<https://stackoverflow.com/a/3904593/13072186>

Szhnet, 2021, accessed 8 of October 2021,  
<https://github.com/msgpack/msgpack-java>