JSON on Mobile: is there an Efficient Parser?

Ricardo Queirós  
ricardo.queiros@eu.ipp.pt

June 19, 2014

SLATE’14  
Escola Superior de Tecnologia e Gestão  
Instituto Politécnico de Bragança
Outline

1 Motivation
2 Serialization Formats
   1 Textual serialization
   2 Binary serialization
   3 Selection of a serialization format
3 Comparison and Benchmark of JSON Libraries
   1 Setup and Methodology
   2 Performance Benchmark
4 Conclusions
Mobile devices & Internet

- Use of mobile devices more than PCs to access the Internet;
- US time spent accessing the Internet by device (Jan’14);

The flip side of this global trend is battery consumption;
- Main causes: the display and network traffic;
Best practices to perform for network operations

1. Consider who initiate the network call (pull vs. push);
2. Consider how much data you need to retrieve (cache);
3. Use transparent compressions;
4. Choose a better **data serialization format**;

- In data storage and transmission, **serialization** is the process of writing an object to a stream of bytes;
- **XML** was used for a long time as the standard language for data representation;
- **XML**’s well-known **verbosity** was tackled by **JSON**;
Best practices to perform for network operations

1. Consider who initiate the network call (pull vs. push);
2. Consider how much data you need to retrieve (cache);
3. Use transparent compressions;
4. Choose a better data serialization format;

- In data storage and transmission, serialization is the process of writing an object to a stream of bytes;
- XML was used for a long time as the standard language for data representation;
- XML’s well-known verbosity was tackled by JSON;
Motivation

Goals

- To find the most efficient JSON parser implementation for mobile communications;
- Based on a performance benchmark we compare the required time to read and write data with several popular JSON parser implementations.
Definition

- Serialization consists in the conversion of an object into a representation that can be transmitted;
- An application that is aware of the serialization format used can then recreate a serialized object by deserialization;
- The object is then restored to its original state. In this process the serialization format plays a central role.
- Two types of serialization: textual and binary.
## Textual serialization

<table>
<thead>
<tr>
<th>Name</th>
<th>Date</th>
<th>Creator</th>
<th>Based on</th>
<th>Schema/IDL</th>
<th>Human-Readable</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSV</td>
<td>1967</td>
<td>Yakov Shafranovich</td>
<td>n/a</td>
<td>partial</td>
<td>yes</td>
</tr>
<tr>
<td>XML</td>
<td>1998</td>
<td>W3C</td>
<td>SGML</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>XML-RPC</td>
<td>1998</td>
<td>Dave Winer</td>
<td>XML/SOAP</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>JSON</td>
<td>2001</td>
<td>Douglas Crockford</td>
<td>JavaScript</td>
<td>partial</td>
<td>yes</td>
</tr>
<tr>
<td>YAML</td>
<td>2001</td>
<td>Clark Evans</td>
<td>C/Perl/XML</td>
<td>partial</td>
<td>yes</td>
</tr>
<tr>
<td>Candle</td>
<td>2005</td>
<td>Henry Luo</td>
<td>XML/JSON</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>OpenDDL</td>
<td>2013</td>
<td>Eric Lengyel</td>
<td>C/PHP</td>
<td>no</td>
<td>yes</td>
</tr>
</tbody>
</table>
# Serialization Formats

## Binary serialization

<table>
<thead>
<tr>
<th>Name</th>
<th>Date</th>
<th>Creator</th>
<th>Based on</th>
<th>Schema/IDL</th>
<th>Human-Readable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avro</td>
<td>2009</td>
<td>ASF</td>
<td>n/a</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>BSON</td>
<td>2003</td>
<td>MongoDB</td>
<td>JSON</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Cap’n Proto</td>
<td>2013</td>
<td>Kenton Varda</td>
<td>protobuf</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Protocol Buffers</td>
<td>2008</td>
<td>Google</td>
<td>n/a</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Thrift</td>
<td>2007</td>
<td>Facebook/Apache</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>
Criteria

- Selection based on the amount of research papers found in the Google Scholar engine;
- Comparison of the three most cited serialization formats:

![Graph showing research articles on serialization]

- Despite the XML format being the most cited in research articles, is the JSON format that has the highest growth in recent years.
Setup and Methodology

- Purpose of this benchmark is only to ensure a reasonable reading and writing performance compared to other parsers;
- To examine the performance of serializing and deserializing structured data, an experiment was designed using the following hardware and software:
  - **Hardware:** ASUS Padfone with 1.5 GHz dual-core Qualcomm Krait and 1 GB memory;
  - **Operating System:** Android version 4.1.1;
  - **Java:** version 1.6.0.
Setup and Methodology

- The test object is based on service called OpenWeatherMap;
- This REST service delivers the weather status of a given city;
- A typical URL request:
  
  http://api.openweathermap.org/data/2.5/weather?q=porto

- The service returns the output in JSON format:

```
{
  "id": 88319, "dt": 1345284000, "name": "Porto",
  "coord": {"lat": 41.15, "lon": -8.61},
  "main": {
    "temp": 306.15, "humidity": 44,
    "temp_min": 306, "temp_max": 306
  },
  "wind": {
    "speed": 1, "deg": -7
  },
  "weather": [
    {
      "id": 500, "main": "Rain",
      "description": "light rain", "icon": "10d"
    },
    {
      "id": 701, "main": "Mist",
      "description": "mist", "icon": "50d"
    }
  ],
  "clouds": {"all": 90},
  "rain": {"3h": 3}
}
```
Setup and Methodology

- The JSON libraries are selected based on their popularity;
- Tested libraries and versions as follows:
  - Gson (2.2.4)
  - Jackson (2.2.1)
  - Minimal-json (0.9.1)
  - org.json (n/a)
- The experiment comprises:
  - 100 iterations for warming, and then 100 iterations for time measure;
  - The execution time was measured using `System.currentTimeMillis()`
  - The average execution time is taken for each operation and library.
When you need to serialize/deserialize without sacrificing performance you should choose Jackson;

Although minimal-json cannot outperform Jackson’s writing performance, it offers a very good reading and writing performance
Conclusions

The main contribution of this paper is two-fold:

- a survey on serialization formats organized by types: textual and binary;
- a performance benchmark that can be useful for others that need to select an efficient parser for mobile communication;

Based on the benchmark results one can conclude that Jackson showed the best combined results.

However, if your mobile app will only deserialize data, minimal-json offers the best performance;
JSON on Mobile: is there an Efficient Parser?

Ricardo Queirós
ricardo.queiros@eu.ipp.pt

June 19, 2014

SLATE’14
Escola Superior de Tecnologia e Gestão
Instituto Politécnico de Bragança