The Journal of Open Source Software

Binary-parser: A declarative and efficient parser generator for binary data

Keichi Takahashi[1]

1 Nara Institute of Science and Technology

DOI: 10.21105/joss.03940

Software

- Review
- Repository
- Archive

Editor: Mark A. Jensen

Reviewers:

- @GaurangTandon
- @Fil

Submitted: 29 September 2021
Published: 21 November 2021

Summary

This paper presents `binary-parser`, a JavaScript/TypeScript library that allows users to write high-performance binary parsers, and facilitates the rapid prototyping of research software that works with binary files and network protocols. `Binary-parser`'s declarative API is designed such that expressing complex binary structures is straightforward and easy. In addition to the high productivity, `binary-parser` utilizes meta-programming to dynamically generate parser codes to achieve parsing performance equivalent to a hand-written parser. `Binary-parser` is being used by over 700 GitHub repositories and 120 npm packages as of September 2021.

Statement of need

Parsing binary data is a ubiquitous task in developing research software. Many scientific instruments and software tools use proprietary file formats and network protocols, while open-source libraries to work with them are often unavailable or limited. In such situations, the programmer has no choice but to write a binary parser. However, writing a binary parser by hand is error-prone and tedious because the programmer faces challenges such as understanding the specification of the binary format, correctly managing the byte/bit offsets during parsing, and constructing complex data structures as outputs.

`Binary-parser` significantly reduces the programmer's effort by automatically generating efficient parser code from a declarative description of the binary format supplied by the user. The generated parser code is converted to a JavaScript function and executed for efficient parsing. To accommodate diverse needs by different users, `binary-parser` exposes various options to ensure flexibility and provide opportunities for customization.

A large number of software packages have been developed using `binary-parser` that demonstrates its usefulness and practicality. Some examples include libraries and applications to work with rainfall radars (Starbeamrainbowlabs, 2021), software-defined radio (Houser, 2021), GNSS receivers (Swift Navigation, 2021), smart meters (Zehir, 2021), drones (Velez, 2020), and thermostats (Beek, 2020).

Design

`Binary-parser`'s design is characterized by the following three key features:

1. **Fast**: `Binary-parser` takes advantage of meta-programming to generate a JavaScript source code during runtime from the user's description of the target binary format.

The generated source code is then passed to the `Function` constructor to dynamically create a function that performs parsing. This design enables `binary-parser` to achieve parsing performance comparable to a hand-written parser.

2. **Declarative**: As opposed to parser combinator libraries (Couprie, 2015; Hutton & Meijer, 1998), `binary-parser` allows the user to express the target binary format in a declarative manner, similar to a human-readable network protocol or file format specification. The user can combine *primitive* parsers (integers, floating point numbers, bit fields, strings and bytes) using *composite* parsers (arrays, choices, nests and pointers) to express a wide variety of binary formats.
3. **Flexible**: Unlike binary parser generators that use an external Domain Specific Language (DSL) (Bangert & Zeldovich, 2014; Kaitai team, 2021), `binary-parser` uses an internal DSL implemented on top of JavaScript. This design allows the user to specify most parsing options as return values of user-defined JavaScript functions that are invoked at runtime. For example, the offset and length of a field can be computed from another field that has been parsed already.

Performance evaluation

To evaluate the parsing performance of `binary-parser`, we implemented a small parser using `binary-parser` (v2.0.1) and three major JavaScript binary parser libraries: `binparse` (v1.2.1), `structron` (v0.4.3) and `destruct.js` (v0.2.9). We also implemented the same parser using Node.js's Buffer API as a baseline. The binary data to be parsed was an array of 1,000 coordinates (each expressed as three 16-bit integers) preceded by the number of coordinates (a 32-bit integer). The benchmarks were executed on a MacBook Air (Apple M1 CPU, 2020). The JavaScript runtime was Node.js (v16.9.1).

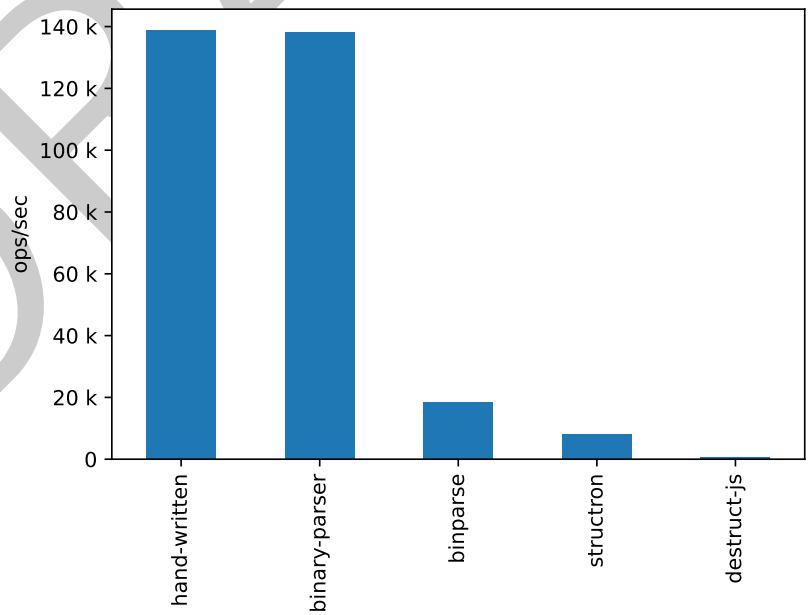

Figure 1: Performance comparison of binary-parser, binparse, structron, destruct.js and a hand-written parser.

Figure 1 shows the measurement results. Evidently, `binary-parser` significantly outperforms its alternatives by a factor of 7.5× to 180×. The plot also reveals that `binary-parser` achieves performance equal to a hand-written parser.

The Journal of Open Source Software

Acknowledgments

This work was partly supported by JSPS KAKENHI Grant Number JP20K19808.

References

Bangert, J., & Zeldovich, N. (2014). Nail: A practical interface generator for data formats. *2014 IEEE Security and Privacy Workshops*, 158–166. https://doi.org/10.1109/SPW.2014.31

Beek, F. (2020). A pimatic plugin to control MAX! Heating devices over a busware CUL stick. In *GitHub repository*. GitHub. https://github.com/fbeek/pimatic-maxcul

Couprie, G. (2015). Nom, a byte oriented, streaming, zero copy, parser combinators library in rust. *2015 IEEE Security and Privacy Workshops*, 142–148. https://doi.org/10.1109/SPW.2015.31

Houser, S. (2021). NodeRed nodes for working with FlexRadio 6xxx series software defined radios. In *GitHub repository*. GitHub. https://github.com/stephenhouser/node-red-contrib-flexradio

Hutton, G., & Meijer, E. (1998). Monadic parsing in haskell. *Journal of Functional Programming*, *8*(4), 437–444. https://doi.org/10.1017/S0956796898003050

Kaitai team. (2021). Kaitai struct: Declarative language to generate binary data parsers. In *GitHub repository*. GitHub. https://github.com/kaitai-io/kaitai_struct

Starbeamrainbowlabs. (2021). Data downloader for the 1km NIMROD rainfall radar data. In *GitHub repository*. GitHub. https://github.com/sbrl/nimrod-data-downloader

Swift Navigation. (2021). Swift binary protocol client libraries. In *GitHub repository*. GitHub. https://github.com/swift-nav/libsbp

Velez, C. (2020). Decrypts and parse DJI logs in node. In *GitHub repository*. GitHub. https://github.com/chrisvm/node-djiparsetxt

Zehir. (2021). In *GitHub repository*. GitHub. https://github.com/Zehir/eesmart-d2l