Ultrafast[1] Lua JSON Parsing

Writing a Lua/JSON encoder + decoder as a LuaJIT module
!!ULTRA DRAFT VERSION MOST OF SLIDES ARE NOT CLEANED UP!!

Adam Ivora

February 1st, 2026

1

The Problem

JavaScript Object Notation ↔ Lua table

```
"{\"foo\":3.5,\"arr\":" +
"[\"x\",true,{}]}"
```

```
{
    ["foo"] = 3.5,
    ["arr"] = {"x", true, {}}
}
```

- not only parsing, but also Lua table construction

Who put comments in my JSON?! aka JBeam

brakepads.jbeam:

```
{
//BASIC BRAKE PADS
"brakepad_F": {
    "information":{
        "authors"="BeamNG"
        "name":"Basic Front Brake Pads",
        "value":[150 3 16],
    },
...
```

solution: pure Lua libraries

- lunajson
- json.lua
- the problem is they are slooooooow

solution: bindings to a fast C++ library

todo slide: simplified json format makes rigid validating parsers (simdjson) infeasible for the task

json-lua-ours

- show bits of code
- choose better name

json-lua-ours: why we need to move forward

- nice that it’s in pure Lua, but
- it’s slow without jit
- performance unpredictable

writing json-c-ours: LuaJIT internals

- key takeaway should be it's a very small codebase for a complete language, and (surprisingly?) extensible

string.buffer serialization

- describe luajit buffer and `lj_serialize.c`

json-c-ours: We did it! What next?

- put perf numbers from initial implementation, compare to pure Lua version
- before any performance number describe basic HW info - CPU model
- decide whether to show final numbers from ryzen or m1 cpu (or both but that'd be too much clutter)

Optimizations: Branch Predictor Hints

`__builtin_expect`

impact: low, put precise number

Optimizations: Intrinsics

whitespace skipping, don’t go too deep
impact: medium

Optimizations: Scratch buffer

avoid realloc overhead, include image
impact: high, mention profiling (or put it into its own slide)

Benchmark: JSON files decoding

todo: describe json corpus and measurement process

	JIT off	JIT on
lunajson	40.3 MB/s	49.6 MB/s
simdjson	TBA	TBA
json-lua-ours	26.7 MB/s	428.6 MB/s
json-c-ours	744.5 MB/s	850.9 MB/s

Benchmark: JBeam files decoding

todo: describe jbeam dataset

	JIT off	JIT on
lunajson	TBA	TBA
simdjson	not applicable	not applicable
json-lua-ours	TBA	TBA
json-c-ours	TBA	TBA

JSON encoding

- decide how much time to devote to encoding, most optimizations were done on decoding so it’s not so interesting

Benchmark: JSON encoding

this is more

	JIT off	JIT on
lunajson	TBA	TBA
simdjson	not applicable	not applicable
json-lua-ours	TBA	TBA
json-c-ours	TBA	TBA

The End

- Benchmark code + implementations available at `https://github.com/BeamNG/TODO`