Parsing Millions of URLs per Second

Yagiz Nizipli 🇺🇸

GitHub: https://github.com/anonrig

Daniel Lemire 🇨🇦

GitHub: https://github.com/lemire/

Software performance

- Reduces cost (AWS, Azure)
- Improves latency
- Reduce complexity (parallelism, caching)

State of Node.js Performance 2023

> Since Node.js 18, a new URL parser dependency was added to Node.js — Ada. This addition bumped the Node.js performance when parsing URLs to a new level. Some results could reach up to an improvement of 400%. (State of Node.js Performance 2023)

Structure of an URL

Example: https://user:pass@example.com:1234/foo/bar?baz#quu

- protocol
- user name, password
- hostname
- port
- pathname
- search
- hash

Examples

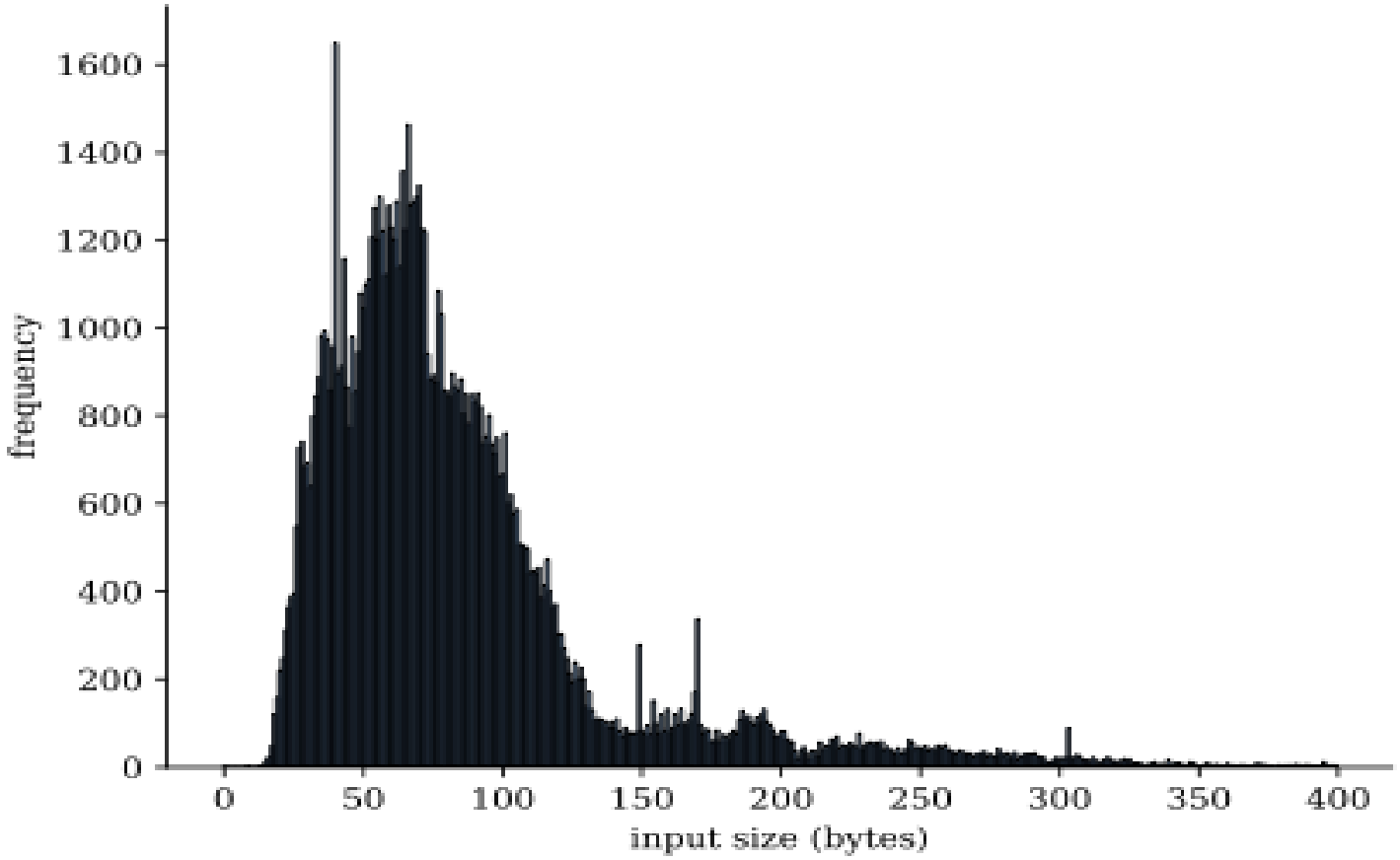
- Long URLs: `http://nodejs.org:89/docs/latest/api/foo/bar/qua/13949281/0f28b//5d49/b3020/url.html#test?payload1=true&payload2=false&test=1&benchmark=3&foo=38.38.011.293&bar=1234834910480&test=19299&3992&key=f5c65e1e98fe07e648249ad41e1cfdb0`
- non-ASCII: `http://你好你好.在线`
- File: `file:///foo/bar/test/node.js`
- JavaScript: `javascript:alert("node is awesome");`
- Percent Encoding: `https://\%E4\%BD\%A0/foo`
- Pathname with dots: `https://example.org/./a/../b/./c`

WHATWG URL

most browsers, JavaScript runtimes; curl, runtime libraries → RFC 3986

string source	value
input string	`https://www.7-Eleven.com/Home/../Privacy/Montréal`
WHATWG URL	`https://www.xn--7eleven-506c.com/Home/Privacy/Montr%C3%A9al`
curl 7.87	`https://www.7-Eleven.com/Privacy/Montr%C3%A9al`
Go runtime (`net/url`)	`https://www.7-Eleven.com/Home/../Privacy/Montr%C3%A9al`

How long are URLs?



https://github.com/ada-url/url-various-datasets/tree/main/top100

How long does it take to parse a URL on average?

curl 7.81.0 (RFC 3986), written in C

- 18 000 instructions/URL
- 7 100 cycles/URL

http benchmark

```
const f = require('fastify')()

f.post('/simple', async (request) => {
    const { url } = request.body
    return { parsed: url }
})

f.post('/href', async (request) => {
    const { url } = request.body
    return { parsed: new URL(url).href }
})
```

Input:

```
{ "url": "https://www.google.com/hello-world?query=search\#value" }
```

URL parsing was a bottleneck in Node 18.15

node version	request/second (simple)	request/second (href)	gap
18.15	60k	54k	10%

Wrote a C++ library (called ada)

- Implemented WHATWG URL support from scratch
- No dependency, full portability
- over 20,000 lines of code
- Six months of work, 25 contributors
- Apache-2.0, MIT
- https://github.com/ada-url/ada

To go faster, use fewer instructions

name	instr./URL	cycles/URL
ada	24,00	550
rust-url (deno)	10,000	2,000
curl	19,000	4,200

https://github.com/ada-url/url-various-datasets/tree/main/top100

Trick 1: perfect hashing

```
std::string_view is_special_list[] = {"http", " ",      "https", "ws", "ftp",  "wss", "file",  " "};


if (scheme.empty()) { return NOT_SPECIAL; }
int hash_value = 2 * scheme.size() + (scheme[0] % 8);
std::string_view target = is_special_list[hash_value];
```

Trick 2: use memoization (tables)

https://en.wikipedia.org/wiki/Memoization

```
uint8_t accumulator = 0;
for (auto c : input) {
    accumulator |=
        is_forbidden_domain_code_point_table_or_upper[c];
}
```

Trick 3: use vectorization

All reasonable desktop/laptop/server general-purpose processors support SIMD (SSE2 or NEON, at least):

Do no process byte-by-byte when you can process 16-byte by 16-byte.

```
size_t i = 0;
const __m128i mask1 = _mm_set1_epi8('\r');
const __m128i mask2 = _mm_set1_epi8('\n');
const __m128i mask3 = _mm_set1_epi8('\t');
__m128i running{0};
for (; i + 15 < user_input.size(); i += 16) {
  __m128i word = _mm_loadu_si128((const __m128i*)(user_input.data() + i));
  running = _mm_or_si128(
      _mm_or_si128(running, _mm_or_si128(_mm_cmpeq_epi8(word, mask1),
        _mm_cmpeq_epi8(word, mask2))),
        _mm_cmpeq_epi8(word, mask3));
}
return _mm_movemask_epi8(running) != 0;
```

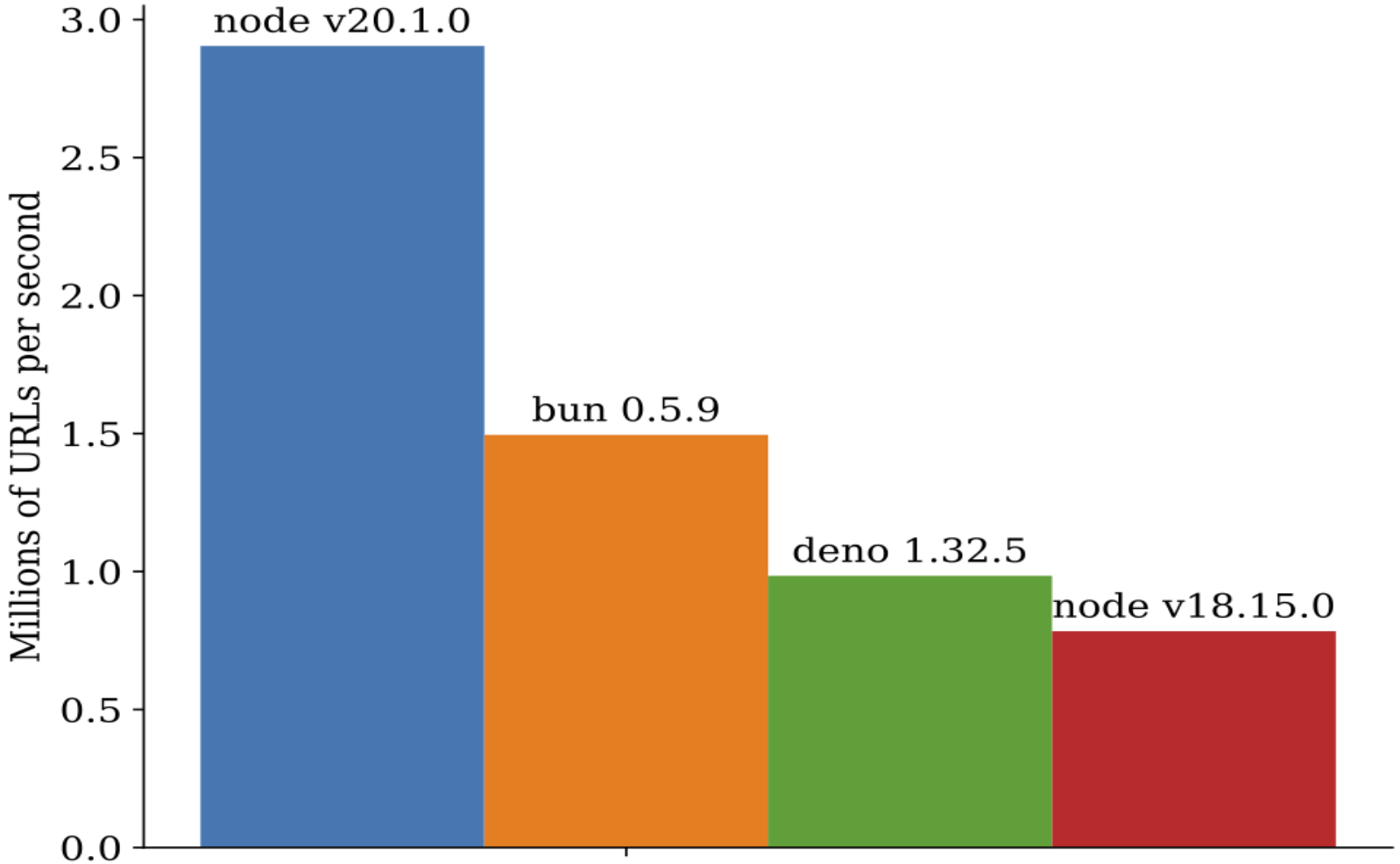
Compilers may do it for you, but not always.

JavaScript benchmark

```
bench(filename, () => {
    for (let i = 0; i < lines.length; i++) {
      try {
        length += new URL(lines[i]).href.length;
        good_url++;
      } catch (e) {
        bad_url++;
      }
    }
    return length;
  });
```

https://github.com/ada-url/js_url_benchmark/

JavaScript results



URL parsing no longer a bottleneck in Node 20

node version	request/second (simple)	request/second (href)	gap
20.1	61k	59k	3%

The ada C++ library is safe and efficient

- Modern C++
- Sanitizers
- Fuzzing
- Unit tests

→ A few minor bugs were reported, related to the standard. Quickly fixed.

Ada is available in the language of your choice

- JavaScript with Node.js (better than 18.15)
- C bindings at https://github.com/ada-url/ada
- Rust bindings at https://github.com/ada-url/rust
- Go bindings at https://github.com/ada-url/goada
- Python bindings at https://github.com/ada-url/ada-python
- R bindings at https://github.com/schochastics/adaR

Often the only way to get WHATWG URL support!

Links

- https://www.ada-url.com (includes a playground)
- @yagiznizipli's blog: https://www.yagiz.co
- @lemire's blog: https://lemire.me