HexaURL Performance: Rejected Changes

Yota Inomoto

“This paper records performance-change proposals that were evaluated but not adopted. The scope includes optimizations around encode, validate, and utils. Adoption decisions are based on measured results from nightly benchmarks.”

February 22, 2026

Contents

1. Purpose

This document records **rejected performance changes** so they can be revisited later with clear context: what was changed, how it was measured, and why it was not adopted.

2. Measurement Policy

- Use `cargo +nightly bench --offline`.
- Run the same benchmark multiple times and decide by average or median.
- Isolate impact by changing one optimization theme at a time.

3. Rejected Changes

3.1. 1. Branch Simplification in `encode_core_validated_inner`

Proposed change

In `crates/hexaurl/src/encode.rs`, simplify per-character branching and remove state-update paths after delimiter errors are detected.

Outcome

Rejected.

Reason (measured)

- `encode_long`: 20.14ns -> 27.45ns (+36.26% slower)
- `encode_medium`: 11.04ns -> 15.09ns (+36.70% slower)
- `encode_short`: 4.61ns -> 5.79ns (+25.62% slower)

The code became simpler, but runtime performance regressed.

3.2. 2. Single-pass `validate_with_config`

Proposed change

In `crates/hexaurl-validate/src/lib.rs`, merge delimiter checks into the first validation pass to remove the delimiter re-scan.

Outcome

Rejected (reverted).

Reason (measured)

- `validate_short`: 5.94ns -> 13.63ns (129% slower)
- `validate_medium`: 18.05ns -> 19.62ns (8.7% slower)
- `validate_long`: 28.92ns -> 29.11ns (0.7% slower)

Extra state-management overhead outweighed potential gains.

3.3. 3. Replace `utils::len` with `memchr`

Proposed change

Replace the custom search in `crates/hexaurl/src/utils.rs` with `memchr::memchr(0, bytes)` and add a `memchr` dependency.

Outcome

Rejected (`utils.rs` and `Cargo.toml` restored).

Reason (measured)

- `len_hexaurl`: 0.98ns -> 1.42ns (43.65% slower)

For current array sizes and access patterns, the existing implementation performs better.

4. Revisit Conditions

Revisit only when all conditions below are met.

1. Gains are confirmed with production-like input distributions (length and delimiter frequency).
2. Improvement is stable across repeated runs.
3. Error precedence and API compatibility remain unchanged.

5. Note

Adopted improvements (outside this document) include pointer-centric `decode_core` optimization and the `decode_into` API family.