



HPC – Projet final

Optimisation de DME (Rust)

Aubry Mangold et Samuel Roland

11 juin 2025

Sommaire

1. Contexte et architecture
2. Rust et Tree-Sitter
3. Tests et infrastructure de benchmark
4. Problème initial de colorisation
5. Optimisation de la colorisation
6. Optimisation de l'installation des grammaires
7. Optimisation de la recherche
8. Conclusion et perspectives

Contexte et architecture

- DME : « Delightful Markdown Experience » (projet scolaire)
- Conversion de Markdown vers HTML/CSS via Comrak
- Recherche de fichiers Markdown dans le système
- Architecture :
 - dme-core en Rust (ce que nous optimisons)
 - Frontend VueJS/Tauri

Rust et Tree-Sitter

- Rust : modèle mémoire strict, ownership & lifetimes
- Comrak pour parser Markdown
- Tree-Sitter pour syntax highlighting
 - CST (Concrete Syntax Tree)
 - Queries et HighlightConfiguration
 - Difficultés avec le modèle mémoire car librairie C
 - Documentation « minimale »

Tests et infrastructure de benchmark

- Tests unitaires et d'intégration
- `criterion.rs` testé mais pas retenu
- Benchmarks hyperfine avec le binaire Rust bench
 - Preview du code
 - Preview du markdown entier
- Exécution systématique en `--release`
- Compilation avec debug symbols pour Perf

Problème initial de colorisation

- HighlightConfig recréé pour chaque snippet
- 117 snippets mènent à 117 initialisations coûteuses
- Perf montre un pic dans HighlightConfig::new

Résultat du benchmark

- preview_code: 5.1575s
- preview_md: 0.0459s

ts_query__perform_analysis
ts_query_new
tree_sitter::Query::new::hd9538119c737e870
tree_sitter_highlight::HighlightConfiguration::new::h71d333ed702fdbe5
once_cell::unsync::OnceCell\$LT\$T\$GT\$::get_or_try_init::h0f0fc5ea937522a9
tree_sitter_loader::LanguageConfiguration::highlight_config::he421b9135ea03f5f
dme_core::preview::tree_sitter_highlight::TreeSitterHighlighter::new::h7b8c0566a5824b5a
_\$LT\$dme_core..preview..comrak..ComrakParser\$u20\$as\$u20\$comrak..adapters..SyntaxHighlighterAdapter\$GT\$::write_highlighted::h4877ee8a4c842d32
comrak::html::format_node_default::h9f596586e4d26695
_ZN6comrak4html30format_document_with_formatter17h03cf562bcdbbfbf5E.llvm.7657930845722867003
comrak::markdown_to_html_with_plugins::h32412c9806cce314
dme_core::markdown_to_highlighted_html::ha9b2e1d3dc29821e
bench::preview::run_preview::h8860045788b0178e
bench::util::run_fn::h4579a263e1e54594
bench::main::h62dbc840ed838d26
std::sys::backtrace::_rust_begin_short_backtrace::hf60e9c62d4cd8c86
_ZN3std2rt10lang_start28_\$u7b\$\$u7b\$closure\$u7d\$\$u7d\$17h59476358557a7bc3E.llvm.9789584609750470467
std::rt::lang_start_internal::h15895544e2012228
main
__libc_start_call_main
__libc_start_main_impl
_start

Optimisation de la colorisation

- Cache global TSH_CACHE: Lazy<RwLock<HashMap>>
- Lecture rapide des highlighters existants
- Écriture (creation) uniquement au premier usage par langue

Résultat du benchmark

- preview_code: 0.3144s → × 16.4
- preview_md: 0.0466s → pas d'amélioration

Optimisation de l'installation de grammaires

- Poids élevé dû à l'historique Git (.git 49 M)
- Passage à `git clone --depth 1 --single-branch`
 - Ajout de paramètres `only_latest_commits` et `single_branch`

Résultat du benchmark

- Avant : 11.4832s
- Après : 1.5225s $\rightarrow \times 7.54$

Optimisation de la recherche

- Indexation rapide du dépôt MDN
- Fuzzy matching sur titres et chemins
- Benchmark général_keyword : 159 ms (index + recherche)
- Streaming des résultats pour réactivité
- Pas eu le temps d'optimiser

Conclusion et perspectives

- Transposition techniques opti C \rightarrow Rust présente des défis
- Importance des tests et bench intégrés
- Gains majeurs avec mise en cache et clone léger
- Pistes futures :
 - Parallélisation du cache (lecture concurrente)
 - Optimisation de la recherche fuzzy