Lõputöö

Tõhusate Andmestruktuuride ja Algoritmide Arendamine Veebirakendustele: Pythoni Jõudluse Optimeerimine WebAssembly (Wasm) abil

Sergei Ivanov

IT Süsteemide arendus, Tartu Ülikooli

Juhendaja: Andre Sääsk

November 5, 2025

Kokkuvõte

Käesolev lõputöö uurib tõhusate andmestruktuuride ja algoritmide arendamist veebirakenduste jaoks, keskendudes Pythoni jõudluse optimeerimisele WebAssembly (Wasm) abil. Uurimistöö hindab Wasm-moodulite integreerimist Pythoni-põhistesse süsteemidesse ning mõõdab jõudluse paranemist reaalse kasutuse kontekstis.

Võtmesõnad: WebAssembly, Cythoni integreerimisega, andmestruktuur, algoritmid, rakendus, jõudlusnäitajaid

Sisukord

Aktuaalsus

Tänapäeva veebirakenduste ja mikroteenuste arenduses on oluline tagada süsteemide skaleeruvus ja jõudlus. Pythoni piiratud jõudlus interpreteeritava keelete tõttu on muutnud süsteemikeeltes arendatud DSA moodulite integreerimise oluliseks uurimisvaldkonnaks.

Probleem ja eesmärk

Probleem

CPythoni standardrakenduses puuduvad tõhusad, mäluturvalised ja suure jõudlusega andmestruktuurid, mis põhjustab viivitusi suurte andmemahtude töötlemisel.

Eesmärk

Arendada ja hinnata süsteemikeeltes (Rust, Zig, Cython) ja WebAssembly abil loodud DSA-mooduli efektiivsust võrreldes standardse Pythoni implementatsiooniga.

Uurimisülesanded

Uurimisülesanded on järgnevad:

1. Arendada samaväärne andmestruktuur (nt Red-Black Tree) kolmes erinevas tehnoloogias: Rust/Wasm, Zig/Wasm ja Cython.
2. Integreerida loodud moodulid Pythoni veebirakendusse ja tagada nende funktsionaalne võrreldavus.
3. Mõõta ja võrrelda jõudlusnäitajaid (täitmise aeg, mälukasutus, integratsiooni overhead) erinevate lähenemiste vahel.
4. Analüüsida, kas Wasm-i kaudu integreerimine vähendab jõudluse kadu võrreldes otsese Cythoni integreerimisega.
5. Hinnata arenduse keerukust ja hooldatavust, et selgitada, milline lahendus on arendajale kõige optimaalsem.

Andmete kogumise ja analüüsimise meetodid

Uuring põhineb kvantitatiivsel lähenemisel. Andmete kogumise meetod: Eksperimentaalne mõõtmine — iga implementatsiooni täitmise aja ja mälukasutuse registreerimine erineva suurusega andmekogumitega. Selleks kasutatakse tekke, näiteks, pytest Pythoni keeles.

Andmete analüüsimise meetod: Statistiline võrdlus (keskmine täitmisaeg, standardhälve, protsentuaalne jõudlusvõit). Tulemusi visualiseeritakse graafikutena (nt matplotlib või pandas DataFrame analüüs).

Akadeemilised allikad

1. Beazley, D. (2010). Understanding the Python GIL. PyCon 2010 Proceedings.
2. Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to Algorithms (3rd ed.). MIT Press.
3. Matsakis, N., & Klock, F. (2014). The Rust Language Design: Safe, Concurrent, Practical Systems Programming. Communications of the ACM, 58(6).
4. Haas, A., Rossberg, A., Schuff, D., et al. (2017). Bringing the Web Up to Speed with WebAssembly. Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI), 185–200.