C++ Benchmarking

**Platformski nezavisno programiranje**

**Sveučilište u Rijeci, Tehnički fakultet**

**smjer računarstvo**

Nino Križanec, 0069055799 Rijeka, 30.08.2016

Sadržaj

[1 Uvod 3](#_Toc460375214)

[1.1 Zašto želimo raditi benchmark koda? 3](#_Toc460375215)

[1.2 Postojeće benchmark knjižnice za C++ 3](#_Toc460375216)

[2 Proces provođenja Benchmark-a 4](#_Toc460375217)

# Uvod

Što je to Benchmarking? Benchmarking je proces prikupljanja, analize, obrade i zaključivanja na temelju dobivenih podataka. Benchmarking kompjuterskog koda nešto je drugačija disciplina pošto ju nije lagano kvantificirati kao neke druge stvari.

U ovome ćemo se izvješću baviti benchmarkom C++ koda, preciznije neovisno o platformi ćemo izraditi alat koji će biti u mogućnosti mjeriti performanse dijelova C++ koda.

Benchmarking se u kodu najčešće izvodi kako bi poboljšali performanse, to naravno uključuje potrošnju CPU vremena, potrošnju memorije, bandwitha i naravno s uvođenjem .

## Zašto želimo raditi benchmark koda?

Kako bismo poboljšali naš kod, tj povećali njegovu efikasnost, moramo znati koliko je on zapravo efikasan, te mijenjaju li naše promjene koda njegovu efikasnost na bolje ili lošije. Efikasnost programa možemo kvantificirati jednostavno, vremenom i resursima utrošenim da bi program izvršio zadatak.

U modernom svijetu važnost efikasnog koda prenosi se ne samo na potrošnju električne energije u data centrima, super računalima pa i običnim stolnim računalima, danas je svijet okružen mobilnom tehnologijom koja najčešće crpi električnu energiju iz baterija. Sve se svodi na to da benchmarkom i profiliranjem našeg koda možemo napraviti kod efikasnijim i smanjiti vrijeme koje procesor mora raditi, te tako produljiti i trajanje baterija kod mobilnih uređaja.

Sada, posebno, potrebno je raditi benchmark koda svakog jezika, ali posebno je važno u C++ jeziku iz razloga što nam C++ dopušta da kontroliramo performanse i direktno utječemo na kompajliranje i izvršavanje koda.

Efikasan program je onaj program koji učini najmanje posla da izvrši zadatak.

## Postojeće benchmark knjižnice za C++

Postoje mnoge gotove i takoreći usavršene knjižnice za benchmark, neke od njih su:

* Hayai library[[1]](#endnote-1)
* Celero library[[2]](#endnote-2)
* Nonius library[[3]](#endnote-3)
* Google Benchmark library[[4]](#endnote-4)

# Proces provođenja Benchmark-a

Kako bismo mogli uspoređivati potrebno je napraviti osnovnu točku (bazno očitanje) benchmark koda kako bismo mogli znati što se događa kada mjenjamo i optimiziramo kod.

## Usporedba dvaju benchmarka

Za primjer ćemo uzeti proteklo procesorsko vrijeme. Postoji više načina usporedbe baznog očitanja i novog (poboljšanog).

Tradicionalni:

* Pokrenuti benchmark početnog koda *n* puta, izmjeriti vrijeme
* Pokrenuti benchmark novog koda *n* puta, izmjeriti vrijeme
* Relativno poboljšanje

Diferencijalni:

* Pokrenuti benchmark početnog koda 2*n* puta, izmjeriti vrijeme
* Pokrenuti benchmark novog koda *n* puta, početnog koda *n* puta i izmjeriti vrijeme
* Relativno poboljšanje
* Poništava jedan dio greške

1. https://github.com/nickbruun/hayai [↑](#endnote-ref-1)
2. https://github.com/DigitalInBlue/Celero [↑](#endnote-ref-2)
3. https://github.com/rmartinho/nonius [↑](#endnote-ref-3)
4. https://github.com/google/benchmark [↑](#endnote-ref-4)