TEHNIČKI FAKULTET – RIJEKA

**Preddiplomski studij računarstva**

**Seminar na temu Green coding**

**Luka Biskupić,Luka Buršić, Patrik Paunović, Matteo Plišić**

Rijeka, svibanj 2023.

Sadržaj

[1. Što je green coding 1](#_Toc135077433)

[2.Green Cloud Computing i virtualne mašine 1](#_Toc135077434)

[2.1 Green Cloud Computing 1](#_Toc135077435)

[2.1 virtualne mašine 3](#_Toc135077436)

[3.   Energetska učinkovitost kroz programske jezike 5](#_Toc135077437)

[3.1 Utjecaj programskih jezika na energetsku učinkovitost 6](#_Toc135077438)

[3.2 Prakse za programiranje u energetski učinkovitim jezicima 7](#_Toc135077439)

[3.3 Fannkuch-redux 8](#_Toc135077440)

[3.4 Fasta 9](#_Toc135077441)

[3.5 Binarno stablo 10](#_Toc135077442)

[4 Optimizacija koda s ciljem smanjenja potrošnje energije 11](#_Toc135077443)

[4.1 Mjerenje potrošnje 11](#_Toc135077444)

[4.1.1 Joulemeter 12](#_Toc135077445)

[4.1.2 PowerAPI 12](#_Toc135077446)

[4.1.3 Intel Power Gadget 13](#_Toc135077447)

[4.2 Optimizacija koda 13](#_Toc135077448)

[4.2.1 Ciljevi optimizacije 14](#_Toc135077449)

[4.2.2 Principi optimizacije 14](#_Toc135077450)

[5. Plan Google-a i Amazona za smanjenje štetnih emisija 15](#_Toc135077451)

[5.1 Green Code - FAANG 15](#_Toc135077452)

[5.1.1 Učinkovitost podatkovnog centra 16](#_Toc135077453)

[5.1.2 Energetski učinkoviti algoritmi 17](#_Toc135077454)

[5.1.3 Virtualizacija i računalni oblak (eng. cloud computing) 18](#_Toc135077455)

[5.1.4 Ulaganje u obnovljive izvore energij**e** 19](#_Toc135077456)

[6. Zaključak 20](#_Toc135077457)

[7. Literatura 20](#_Toc135077458)

# Što je green coding

Bez sumnje, softver je „backbone“ gotovo svega u današnjem svijetu te je njegova potrošnja energije svakako velika i trebalo bi se tražiti riješenja kako bi taj softver bio što bolji sa ekološke perspektive, a tu dolazimo do Green codinga. „Green coding“ je praksa u programiranju kojom se nastoji minimizirati potrošnju energije u procesu obrade linija koda tj. njegovom izvršavanju ti na taj način pomoći organizacijama smanjiti ukupnu potrošnju energije i doprinosi ideologiji održivog razvoja. Ta se praksa ne bi trebala smatrati suprotnost trenutnoj prakse, već bi trebalo bit inkorporirano u principe programiranja koje bi trebalo razmotriti pri pisanju i dizajniranju koda kako bi se uravnotežila funkcionalost i potrošnja električne energije. U prošlosti pisanje kodova je imalo svoja ograničenja poput propusnost mreže i računalne snage tadašnjih računala. U današnje vrijeme može se zasigurno reći kako velike IT kompanije zanemarujući potrošnju energije svog softvera doprinose zagađenju okoliša slično kao što tvornice zagađuju ispušnim plinovima, otpadnim vodama i slično.

Količina energije potrošena u data centrima po svijetu je 1.8% - 3.9% ukupne globalne potražnje za strujom. Kako se svijet polako okreće održivom razvoju, u to se uključuju i IT kompanije i fakulteti kao što je naš te se radi na projektima o green codingu o kojima ćemo nešto više pričati u nastavku.

Neki od primjera tehnologija green codinga su: optimizacija kodova, korištenje, uporaba cloud-ova i slično. Uz sam softver, pod green coding se smatra i hardver, njegova bolja iskoristivost električne energije, manja potrošnja i recikliranja korištenih komponenti u nekom drugom sustavu. U ovim tehnologijama se moeže reći kako prednjači Europa, tj. Europska Unija kao i u ostalim aspektima održivog razvoja i očuvanja okoliša. Uz sve ove metode i tehnologije, spominjat ćemo akcije i planove pojedinih popularnih kompanija koje se okreću green codingu, kao što su Amazon i Google. Iako green coding neće riješiti sve probleme industrije (raspodjela, korištenje i upravljanje krajnjim životom mobilnih uređaja) može biti bitan iskorak pri smanjenju potrošnje energije. Dodatno, uključivanje načela green codinga može približiti održivi razvoj omugućavanjem veće uporabe obnovljivih izvora energije upravljanjem i raspodjelom radnih opterećenja i resursa u sustavima poput cloud-a.

# 2.Green Cloud Computing i virtualne mašine

## 2.1 Green Cloud Computing

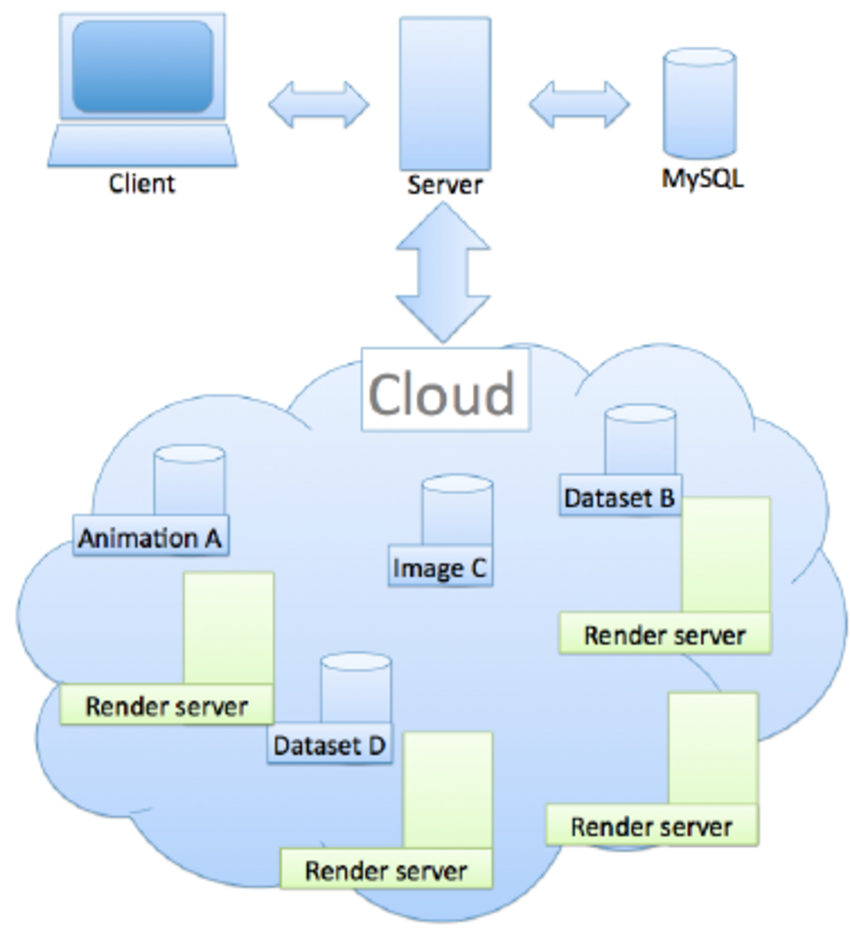
Green Cloud computing je takoreći buzzword u IT industriji koja se odnosi na potencijalne prednosti koje usluge temeljenim na oblaku doprinose društvu.

Green cloud computing ima 3 glavna cilja: maksimizirat iskoristivost energije tijekom životnog ciklusa proizvoda , promovirati korištenje reciklabilnih materijala i minimiziranje korištenja rizičnih it komponenti. Green cloud computing se može sagledati sa dvije perspektive: zeleni hardware i zeleni software.

Zeleni hardver uključuje energetski učinkovit i ekološki prihvatljivi alati informacijske i komunikacijske tehnologije kao što su serveri, mrežna oprema, uređaji za pohranu, napajanja i hladnjaci.

Green software uključuje sve aplikacije tj. programe koji upravljaju podatkovnim centrima i ostalim uslugama temeljenim na oblaku(Cloud-u). Najbitnija ideja iza metodologija zelenog softverskog inženjerstva je izgradnja kvalitetnih aplikacija koje pored zahtjeva organizacija, su također i energetski učinkovite. Logično je pitanje zašto je Cloud computing ekološki održiv, a odgovor na to pitanje se sastoji od tri dijela:

optimizira učinkovito osiguravanje resursa, nudi prednosti višestanarske uporabe, dematerealizira i smanjuje ukupne emisije ugljika.

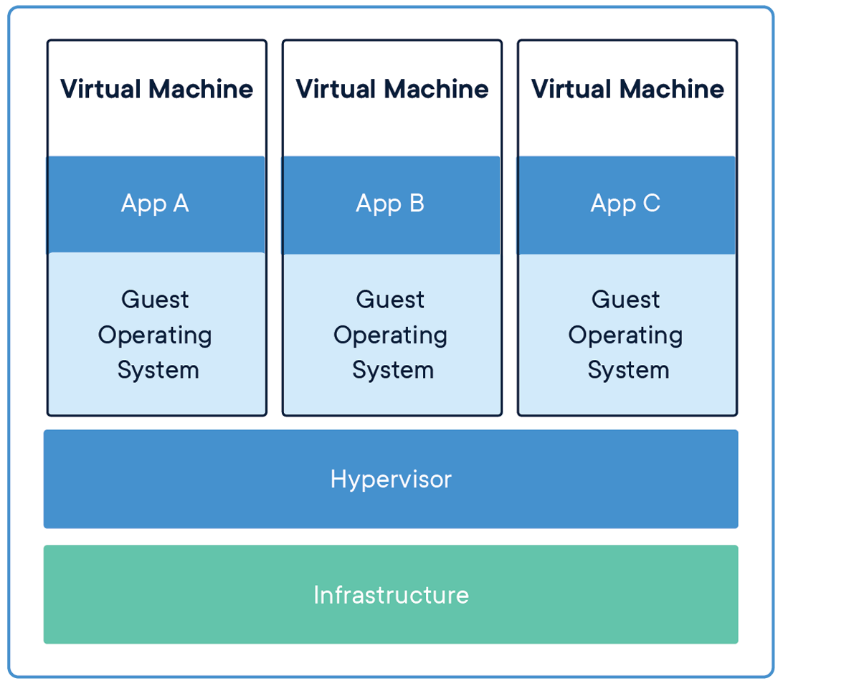


U prošlosti IT kompanije su implementirale više servera i ostalog hardwera nego li je realno bilo potrebno zato što su imali poteškoće shvačajući i predviđajući vrhunce opterećenja i rasta potražnje te su kupovale dovoljno hardvera kako bi bili sigurni. Sa cloud computing-om, organizacija može postići učinkovitiju iskoristivost servera, veću fleksibilnost radnog opterećenja i energetski efikasniju infrastrukturu nego u okruženja u vlastitim prostorima. U istraživanju koje je proveo Accenture, rezultati su pokazali kako organizacije koje su svoja radnja opterećenja prebacili na efikasna riješenja u oblaku su sačuvali 30%-40% cijene u odnosu na one koje su koristile IT infrastrukture u vlastitim prostorijama.

Dijeljene IT infrastrukture ili višestanarski okoliši su više učinkoviti u svojim podhvatima od autonomnih komponenti. To se može predstaviti analogijom kako više stanara u istom stanu troše manje energije nego li bi trošili odvojeno svatko u svom stanu. Isto tako više stanara na infrastrukturi baziranoj na oblaku minimizira potrošnju energije i povezani ugljični otisak. Također Cloud computing štedi energiju poboljšanom iskorištenosti servera u odnosu na IT arhitekturu u vlasitim prostorima. Cloud computing taođer smanjuje ukupne emisije ugljika. Jednostavan razlog tome je što za implementaciju vlastitih poslužitelja i ostalih mrežnih komponenti, potrebno je sve materijale dovesti do određene lokacije te ih sastaviti ili dovesti stručnjake koji će to učiniti te stoga zamjena fizičkih komponenti sa „virtualnima“ smanjuje i troškove i zagađenje. Prema procjenama Accentura, javni oblaci bi mogli pomoći pri smjanjenju emisija ugljika do čak 5.9%.

## 2.1 virtualne mašine

Virtualne mašine, iako dijele neke karakteristike sa Cloud computing-om, one nisu ista stvar. Virtualna mašina je softverska emulacija fizičkog računala, sa vlastitim operacijskim sustavom i hardverskim resursima, pokrenuto na fizičkom serveru. Također više virtualnih mašina može biti pokrenuto na istom fizičkom serveru, izoliranih međusobno kako problemi jedne ne bi utjecali na ostale virtualne mašine. One se koriste u svrhu manje, odnosto efektivnije potrošnje električne energije. U kontekstu podatkovnih centara, virtualizacija je instaliranje virtualne infrastrukture koja omogućava više operacijskih sustava i aplikacija na manji broji servera, smanjujući potrošnju energije i samim time cijenu. To sve naravno podrazumijeva kako svaka od tih virtualnih mašina ima manje resursa od cijelokupnog servera. Neki direktnih prendosti virtualizacije tj. virtualnih mašinasu: Vrijeme u kojem je sustav nedostupan je minimalan ili jednak nuli jer se virtualna mašina može migrirati sa jednog servera na drugi, dinamički uravnoteženo radno opterećenje kroz servere i automatski prelazak na rezervni sustav u slučaju kvara, Alokacija resursa je bolje kontrolirana i odražvana, povećana sposobnost servera da dijele korisnost, iskoristivost servera može biti povećana do 80% (iskoristivost servera ne povećanje od 80%). Kako bi se ovo stavilo u kontekst, ušteda energije po prosječnom serveru koristeći virtualizaciju godišnje bi iznosilo oko 700kwh.



Virtualizacija može biti podijeljena u 5 tipova ili podskupina:

Virtualizacija operacijskih sustava koja služi kako bi se riješili sigurnosni problemi, pri ćemu virtualizacija igra veliku ulogu.

Virtualizacija platformi, ovakav tip virtualizacije dopušta da pokrećemo bilo koji operacijski sustav i/ili aplikaciju u virtualnu okruženju. Postoje dvije metode platformske virtualizacije: Potpuna virtualizacija platforme i Parcijalna virtualizacija platforme. Razlika je u tome što u potpunoj virtualizaciji platforme, gost ne mora imati nikakve modifikacije, dok se kod parcijalne virtualizaciji platforme gostujući program pokreće u vlastitom izoliranom okruženju.

Virtualizacija pohrane, pod ovo bi spadali sustavi u oblaku i cloud computing o kojem je nešto više spomenuto u prethodnom podnaslovu.  
Mrežna virtualizacija koja je često korištena u lokalnim mrežama i virtualnim servisima, gdje su potrebne tehnike poput load balancinga zbog potrebe za dinamičkim skaliranjem ponuđenih usluga.

Virtualizacija aplikacija enkapsulira računalne programe sa operacijskih sustava na kojem se izvršavaju. Prednosti su kompaktnost, lokalna dostupnost i automatsko upravljanje za nadogradnje

Kao i velika većina stvari u IT industriji, virtualizacija ima svoje prednosti i nedostatke. Neke od prednosti su:

Korištenje resursa: Fizički poslužitelji često ne rade punim kapacitetom jer ovi operateri generalno dopuštaju dovoljno računalnih resursa da pokriju maksimalnu upotrebu i najzahtjevnije zahtjeve. Ako je više resursa potrebno pojedinom zadatku ili aplikaciji može ih dodatno uzeti, no ako je prevelika količina zahtjeva, može trebati dulje virjeme da se odgovori na zahtjev.

Menadžment: Moguće je kontrolirati resurse automatski pošto virtualna mašina može konfigurirana na takav način

Konsolidacija: Koristi se za pokretanje raznih aplikacija na manjem broju kompnenti

Potrošnja energije: Koristeći konsolidaciju, manji broj komponenti logički i troši manju količinu električne energije

Manje potrebnog prostora: Sa manjim potrebnim prostorom dolazi i manji trošak za iznjamljivanjem i kupnjom poslovnog prostora, što je pogotovo u današnje vrijeme rastućih cijena nekretnina veliki dobitak i bitna stavka poslovanja.

Planiranje u slučaju hitnoće: Moguće je maknuti virtualne mašine sa jednog skupa resursa na drugi što omogućava veću dostupnost prema postojećim uslugama

Nedostaci, iako manji i dalje su postojani, a ti su:

Visoka početna cijena implementacije zbog potrebnog posebnog softvera, hardvera i stručnjaka sposobnih za njegovu implementaciju.

Sigurnosni rizik zbog dijeljenih resursa, kao i kompleksne arhitekture što ostavlja veći prostor za grešku, što softvera, tako i ljudske greške

Mogući problemi sa dostupnosti u slučaju da resursi nisu dostupni za obavljanje zadataka.

Problem sa skalabilnosti, pošto puno entitea dijeli iste resurse, što potencijalno stvara zakašnjenja i eventualne zastoje unutar virtualizacijske mreže

Postoji više povezanost više dijelova te greška jednog dijela dovodi do prestanka rada cijelog „lanca“

# 3.   Energetska učinkovitost kroz programske jezike

U današnje vrijeme kada je potreba za smanjenjem potrošnje energije i emisije stakleničkih plinova sve veća, programiranje u energetski učinkovitim programskim jezicima postaje neizostavan element razvoja softvera. U ovom poglavlju ćemo istražiti kako programski jezici utječu na potrošnju energije i emisiju stakleničkih plinova, te koji su programski jezici najbolji za smanjenje potrošnje energije.

## 3.1 Utjecaj programskih jezika na energetsku učinkovitost

Programski jezici se razlikuju u svojim karakteristikama i performansama, što može utjecati na potrošnju energije. Na primjer, jezici koji koriste interpretaciju umjesto kompilacije (npr. Python, Ruby) imaju tendenciju da budu manje energetski učinkoviti, jer interpretacija zahtijeva dodatnu obradu i resurse. S druge strane, jezici koji se kompiliraju (npr. C, C++) obično su brži i energetski učinkovitiji, jer se programi prevode u strojni jezik prije izvršavanja.

Osim toga, karakteristike samog jezika također mogu utjecati na energetsku učinkovitost. Na primjer, jezici poput Java i C# imaju vlastitu virtualnu mašinu koja zahtijeva dodatne resurse, što može dovesti do povećane potrošnje energije u usporedbi s drugim jezicima.

Nadalje, ugrađene funkcionalnosti u jeziku, kao što su automatizirano upravljanje memorijom i garbage collection, mogu također utjecati na energetsku učinkovitost. Na primjer, programski jezik Python ima garbage collector koji se automatski izvršava u pozadini kako bi se oslobodili resursi koji nisu više u upotrebi. Iako ovo olakšava programiranje i smanjuje potrebu za ručnim upravljanjem memorijom, sam proces može zahtijevati više procesorske snage i povećati potrošnju energije.

Još jedan faktor koji može utjecati na energetsku učinkovitost jezika je njihova popularnost. Jezici koji se široko koriste i imaju veliku zajednicu programera često se razvijaju i optimiziraju za energetsku učinkovitost kako bi se poboljšala njihova performansa. S druge strane, manje popularni jezici mogu imati manje podrške i optimizacije za energetsku učinkovitost, što može dovesti do veće potrošnje energije.

Uz to, s obzirom na to da se softverska arhitektura i tehnologije brzo razvijaju, programski jezici se stalno nadograđuju kako bi poboljšali svoju energetsku učinkovitost. Na primjer, programski jezik Go razvijen je s naglaskom na brzinu i učinkovitost, a koristi se u mnogim aplikacijama koje zahtijevaju veliku procesorsku snagu, poput obrade podataka i mrežnih aplikacija.

U konačnici, odabir programskog jezika za određenu aplikaciju ovisi o mnogim faktorima, uključujući zahtjeve za performansom, razinu apstrakcije, korištenje interpretacije ili kompilacije te dostupnost optimizacija za energetsku učinkovitost. Stoga je važno da programeri budu svjesni ovih faktora i odaberu najbolju opciju koja će im omogućiti razvoj energetski učinkovitog softvera.

Slika na kojoj se prikazuje tekst, izbornik, snimka zaslona, Font

Opis je automatski generiran

## 3.2 Prakse za programiranje u energetski učinkovitim jezicima

Programeri mogu koristiti nekoliko praksi za programiranje u energetski učinkovitim jezicima. Jedna od najvažnijih praksi je optimizacija koda, koja uključuje smanjenje broja operacija koje program obavlja i smanjenje broja varijabli koje se koriste. Također je važno izbjegavati ponovno izračunavanje vrijednosti koje se mogu spremiti u memoriju i koristiti kasnije.

Osim optimizacije koda, programeri mogu koristiti i druge prakse za programiranje u energetski učinkovitim jezicima. Jedna od njih je upotreba specijaliziranih alata za mjerenje potrošnje energije, koji omogućuju programerima da procijene energetsku učinkovitost njihovog koda i pronađu potencijalne izvore potrošnje energije.

Također je važno koristiti energetski učinkovite biblioteke i okvire za razvoj softvera, koji su specijalizirani za nisku potrošnju energije. Ove biblioteke i okviri mogu uključivati optimizacije poput minimiziranja broja mrežnih zahtjeva, smanjenje broja operacija s diska i optimizacija memorije.

Još jedna važna praksa je korištenje algoritama i struktura podataka koji su energetski učinkoviti. Na primjer, korištenje algoritama s niskim troškovima u pogledu procesora i memorije može biti ključno za energetsku učinkovitost. Također, upotreba struktura podataka koje koriste manje memorije i brže se obrađuju može dovesti do značajne uštede energije.

Naposljetku, programeri bi trebali voditi računa o energetskoj učinkovitosti i u fazi dizajna i planiranja. Odabir energetski učinkovitih jezika, alata i arhitektura zahtijeva pomno razmatranje i procjenu potreba projekta. Integracija energetske učinkovitosti u razvojni proces od samog početka može dovesti do značajne uštede energije i poboljšanja performansi krajnjeg proizvoda.

Sve u svemu, programiranje u energetski učinkovitim jezicima zahtijeva kombinaciju pravilnog odabira jezika i alata, optimizacije koda, upotrebe energetski učinkovitih algoritama i struktura podataka, te pažljivo planiranje i dizajn. U konačnici, energetska učinkovitost bi trebala biti jednako važna kao i funkcionalnost i performanse u razvoju softvera.

## 3.3 Fannkuch-redux

"Fannkuch-redux" (Couto, 2017) je poznati problem optimizacije koji se često koristi za testiranje performansi različitih programskih jezika i optimizacijskih tehnika. Cilj problema je izračunati broj inverzija u permutaciji brojeva od 1 do N.

Konkretno, problem se sastoji od pronalaženja najveće moguće vrijednosti P, gdje P predstavlja permutaciju brojeva od 1 do N, tako da primjena operacije "flip" na permutaciju P, koja mijenja redoslijed prvih k elemenata u P, dovodi do permutacije koja ima manji broj inverzija od originalne permutacije.

"Fannkuch" u nazivu problema označava poznati recept za pripremu francuske torte "fancier cake", a "redux" znači "ponovo izrađen". Dakle, ime "fannkuch-redux" dolazi od ideje da se problem sastoji od ponovnog izračuna broja inverzija u permutaciji, što podsjeća na pripremu francuske torte koja zahtijeva ponovno izrađivanje kora.

Slika na kojoj se prikazuje tekst, snimka zaslona, izbornik, broj

Opis je automatski generiran

## 3.4 Fasta

FASTA je jedan od najstarijih i najčešće korištenih formata za pohranu i razmjenu sekvenci nukleinskih i proteinskih kiselina. FASTA format koristi jednostavan tekstualni format u kojem se prva linija zapisa obično sastoji od jedinstvenog identifikatora sekvence, dok se druga linija sastoji od same sekvence u obliku niza slova (A, C, G, T za DNA ili A, C, G, U za RNA kodiranje; ili različitih jednoznačnih kodona za proteinske sekvence).

Format FASTA se može koristiti za različite svrhe, kao što su pretraživanje baza podataka sekvenci, upoređivanje sekvenci, konstruiranje filogenetskih stabala i slično. Mnoge bioinformatičke alate i biblioteke podržavaju učitavanje i obradu sekvenci u FASTA formatu, što ga čini ključnim formatom u ovoj oblasti. Generiralo se i pisalo veća broja DNA sekvenci u svrhu istraživanja energetske učinkovitosti.

Slika na kojoj se prikazuje tekst, izbornik, snimka zaslona, broj

Opis je automatski generiran

## 3.5 Binarno stablo

Binarno stablo je struktura podataka koja se sastoji od čvorova, od kojih svaki sadrži jedan element podataka, i najviše dva potomka (lijevo i desno podstablo). Binarna stabla se često koriste za pretraživanje i sortiranje podataka.

Također, pri implementaciji binarnog stabla može se koristiti tehnika memoizacije, što omogućuje ponovnu upotrebu izračunatih vrijednosti i smanjenje broja operacija koje su potrebne za izvršavanje programa. Na primjer, ako se vrijednost lijevog podstabla već izračunala i spremljena je u memoriju, tada se ta vrijednost može koristiti za brži pristup podacima.

Kod raznih programskih jezika se u svrhu istraživanja njihove energetske učinkovitosti alociralo, obilazilo i dealociralo nekoliko binarnih stabala.

Slika na kojoj se prikazuje tekst, izbornik, snimka zaslona, broj

Opis je automatski generiran

# 4 Optimizacija koda s ciljem smanjenja potrošnje energije

Optimizacija koda s ciljem smanjenja potrošnje energije ima sve veću važnost u današnjem svijetu gdje su održivost i zaštita okoliša sve važniji izazovi. Ova vrsta optimizacije ima za cilj smanjiti potrošnju energije koju generira računalni program pri izvršavanju. Optimizacija koda predstavlja važan aspekt softverskog inženjerstva, no njena važnost ovisi o tipu projekta. Za slučaj da se kod pokreće jako rijetko, često nam nije bitno hoće li nekoliko sekundi brže ili sporije biti izvršen, ali ako se radi o procesu koji se primjerice izvršava tisuće puta na dan, njegova nam je optimizacija uvelike bitna.

## 4.1 Mjerenje potrošnje

Da bi se optimizacija koda mogla kvalitetno provesti, potrebno je koristiti alate za mjerenje potrošnje energije potrebne za izvršavanje programa. Neki od najpoznatijih alata su: Joulemeter, PowerAPI te Intel Power Gadget.

### 4.1.1 Joulemeter

Joulemeter je aplikacija koja korisniku daje uvid u potrošnju energije virtualnih mašina, uređaja te radnih površina, ali i zasebnih aplikacija. Potrošnja se određuje na temelju podataka koji su dostupni kroz Windows Performance Monitor API, a ti podaci opisuju vrijednosti kao što su iskorištenost CPU i diska te svjetlina ekrana. Osim što omogućuje mjerenje potrošnje energije, aplikacija nudi i mogućnost podešavanja željenih granica potrošnje te analize i izvještavanja o potrošnji energije. [6]

Na sljedećoj slici možemo vidjeti kako ovaj alat vraća podatke o energiji potrebnoj za rad procesora, ekrana, diska te energiji koju uređaj troši u stanju mirovanja.

Slika na kojoj se prikazuje tekst, softver, web-stranica, Ikona na računalu

Opis je automatski generiran

### 4.1.2 PowerAPI

PowerAPI je open source python okvir za izgradnju softverski definiranih mjerača snage. PowerAPI omogućuje korisnicima da izgrade mjerače potrošnje energije prema njihovim specifičnim potrebama, koristeći Python i biblioteke koje dolaze s okvirom. Mjerači se mogu konfigurirati da prikupljaju podatke o potrošnji energije na različitim razinama, uključujući pojedinačne aplikacije, procese i čak cijele sustave. Koristi dvije formule: RAPL formulu za mjerenje potrošnje kod uređaja te SmartWatts formulu za mjerenje potrošnje energije kod procesa.

### 4.1.3 Intel Power Gadget

Intel Power Gadget je alat za mjerenje potrošnje na Intel Core procesorima u stvarnom vremenu. Osim toga, ovim se alatom mogu mjeriti i temperatura i frekvencija procesora. Alat se može koristiti za izradu grafova kako bi se lako vizualizirale promjene u performansama i potrošnji energije tijekom vremena. Također, postoji i C/C++ API koji omogućuje dohvaćanje podataka unutar programa. [7]

Na sljedećoj slici možemo vidjeti neke od podataka koji se mogu izmjeriti koristeći ovaj alat, među kojima su iskorištenost procesora (CPU Utilization), prosječna temperatura procesorskog paketa (Package Temperature\_0) te potrošnja energije grafičkog podsustava (GT Power\_0).

Slika na kojoj se prikazuje tekst, račun, snimka zaslona, Font

Opis je automatski generiran

## 4.2 Optimizacija koda

### 4.2.1 Ciljevi optimizacije

Energetska učinkovitost postala je ključna tema u razvoju softvera zbog porasta potrebe za smanjenjem potrošnje energije i smanjenja emisije ugljika. Optimizacija koda jedan je od načina za postizanje energetski učinkovitog softvera.

Osnovni ciljevi optimizacije koda su:

* smanjenje procesorske aktivnosti - visoka aktivnost procesora ima direktan utjecaj na potrošnju energije
* optimizacija upita baze podataka - cilj je da se izvršavanje upita i pristupanje podacima izvršava u što kraćem roku
* efikasno korištenje memorije - za energetsku učinkovitost programa važno je smanjiti alokaciju nepotrebnih resursa te izbjeći curenja memorije
* smanjenje nepotrebnih operacija - uklanjanje operacija kao što su neke beskorisne provjere i dupliranje koda doprinosi smanjenju potrošnje energije

### 4.1.3 RAPL (Running Average Power Limit)

RAPL je alat za upravljanje potrošnjom energije koji se koristi na određenim procesorima, posebno Intel procesorima. On omogućuje nadzor, kontrolu i optimizaciju potrošnje energije u različitim komponentama računalnog sustava.

RAPL pruža mogućnost praćenja i mjerenja trenutne potrošnje energije CPU-a, GPU-a i ostalih komponenti u stvarnom vremenu. To omogućuje korisnicima da dobiju uvid u energetsku učinkovitost svojih sustava i identificiraju potencijalne izvore visoke potrošnje energije.

Jedna od popularnih knjžnica za korištenje RAPL-a je PyRapl.

Identifikacija kritičnih dijelova koda koji značajno utječu na potrošnju energije također je važan aspekt optimizacije. Ona je ključna za usmjerenje optimizacijskih napora na dijelove koji najviše utječu na potrošnju energije. To mogu biti dijelovi koda koji često izvršavaju petlje ili zahtijevaju intenzivnu obradu podataka. Identificiranje tih dijelova omogućuje programerima da ciljano optimiziraju kod i postignu značajna poboljšanja u energetskoj učinkovitosti.

### 4.2.2 Principi optimizacije

Postoje razni pristupi optimizaciji koda koji se razlikuju ovisno o programskom jeziku te o ciljevima samog programa. No, svi ti pristupi se zasnivaju na nekoliko univerzalnih principa, kao što su:

* korištenje ispravnih tipova podataka
* Korištenjem ispravnih tipova podataka smanjuje se broj pretvorbi tipova, čime se postiže veća efikasnost. U pravilu, potrebno je koristiti najmanji tip podataka koji dovoljno precizno predstavlja podatke koji se koriste.
* spremanje često korištenih podataka u cache memoriju
* Ova se tehnika koristi za podatke kojima se često pristupa tijekom izvršavanja programa. Umjesto da se podaci svaki put ponovno izračunavaju ili dohvaćaju iz sporijih izvora kao što su baze podataka ili vanjski API-jevi, podaci se spremaju u cache memoriju, što omogućava puno brži pristup. Prilikom implementacije ove tehnike, potrebno je odrediti veličinu cachea, politiku zamjene podataka u cacheu te mehanizam osvježavanja cachea.
* izbjegavanje nepotrebnih ulazno-izlaznih operacija

Ovom tehnikom želi se postići da se čitanje i zapisivanje podataka u datoteke izvršava samo kada je to potrebno zbog toga što nepotrebne I/O operacije imaju negativan utjecaj na energetsku učinkovitost. To se može postići provođenjem ovih principa:

o   čitanje samo onih podataka koji su stvarno potrebni

o   grupiranje podataka prilikom zapisivanja

o   korištenje efikasnih protokola komunikacije

* korištenje efikasnih algoritama

Prilikom pisanja programa, potrebno je odabrati algoritam koji je optimalan po pitanju prostorne i vremenske složenosti. Na primjer, ako se u programu treba sortirati određeni skup podataka, korištenje algoritma brzog sortiranja (quick sort) bit će efikasnije od korištenja algoritma mjehurićastog sortiranja (bubble sort). Brzo sortiranje ima nižu vremensku složenost od mjehurićastog sortiranja, što znači da će manje vremena trebati za sortiranje podataka. Osim toga, brzo sortiranje zahtijeva manje resursa od mjehurićastog sortiranja, čime se smanjuje opterećenje sustava. [8]

# Plan Google-a i Amazona za smanjenje štetnih emisija

## 5.1 Green Code - FAANG

U ovom dijelu seminara ćemo razmotrit kako su velike FAANG korporacije (Facebook - Meta, Amazon, Apple, Netflix, Google) implementirale *Green code* principe tj. na koji način su te tvrtke optimizirale svoje softewara-e i svoju infrastrukturu tako da smanje negativan učinak koji ostavljaju na okoliš.

Usredotočit ćemo se na pet glavnih načina na koji su te tvrtke smanjile loš utjecaj na okoliš:

Učinkovitost podatkovnog centra (*Data center efficiency*)

Energetski učinkoviti algoritmi

Virtualizacija i Cloud computing

Održivi razvoj software-a

Ulaganje u obnovljive izvore energije

### 5.1.1 Učinkovitost podatkovnog centra

FAANG tvrtke su se usredotocile na optimiziranju svojih podatkovnih centara kako bi smanjili potrošnju energije. To su ostvarili tako da su implementirali napredne tehnike hlađenja, učinkovit dizajn za svoje servere i obnovljive izvore energije za napajanje svojih podatkovnih centara.

#### 5.1.1.1 Tehnike hlađenja

Neke od naprednih tehnika hlađenja koje su implementiranje su sljedeće: ekonomizator zraka (air-side economizer), likvidno hlađenje, zadržavanje toplo/hladnog prolaza, ponovna uporaba topline.

**Ekonomizator zraka** (eng. *Airside Economization*) je tehnika hlađenja u kojoj se koristi vanjski zrak za hlađenje podatkovnog centra umjesto da se koristi mehanički sustav hlađenja kao što je klima uređaj ili nešto slično tome. Podatkovni centri dizajnirani su sa sustavima za upravljanje protokom zraka i senzorima koji detektiraju razinu temperature i vlažnosti. Kada su uvjeti povoljni, vanjski zrak se dovodi i koristi za hlađenje, čime se smanjuje potreba za mehaničkim hlađenjem i štedi energija. [9]

**Likvidno hlađenje** (eng. *Liquid Cooling*) je efikasnije rješenje hlađenja sustava u odnosu na hlađenje hladnim zrakom. Ova tehnika uključuje korištenje rashladnih tekućina za raspršivanje topline koju stvaraju serveri. [10]  
  
**Zadržavanje toplo/hladnog prolaza** (eng. *Hot/Cold Aisle Containment*). Serveri u podatkovnim centrima su poslagani na način da stvaraju tople i hladne prolaze. Zadržavanje toplog zraka u toplim prolazima uključuje izoliranje toplog zraka koji izlazi iz servera i preventiranje da se taj zrak miješa sa hladnim. Isto vrijedi i za hladne prolaze gdje se hladni zrak izolira i preventira se da se hladni zrak miješa sa toplim i sve u svrhu toga da se topli zrak uspješno i efikasno izbaci van sistema te kako bi se hladni dove do servera i iste ohladio.

**Ponovna upotreba topline.** FAANG tvrtke istražuju mogućnosti ponovne upotrebe otpadne topline koju generiraju podatkovni centri. Na primjer, višak topline može se uhvatiti i koristiti za grijanje obližnjih zgrada ili za druge industrijske procese, smanjujući oslanjanje na odvojene sustave grijanja i minimalizirajući gubitak energije.

#### 5.1.1..2 Učinkoviti dizajn servera

Učinkovit dizajn poslužitelja odnosi se na razvoj i implementaciju hardverske-a koji daje prioritet energetskoj učinkovitosti i optimizaciji resursa. FAANG tvrtke uložile su značajne napore u dizajn i implementaciju poslužitelja koji troše manje energije, a istovremeno održavaju visoke performanse.

Jedan od načina na koji se postiže takva optimizacija servera tj. poslužitelja je kroz suradnju sa proizvođačima komponenti i proizvođačima poslužitelja kako bi tvrtke izabrale energetski učinkovite procesore, memorijske module, uređaji za pohranu i mrežnu opremu. Ove komponente dizajnirane su za pružanje visokih performansi uz minimalnu potrošnju energije, omogućujući bolju energetsku učinkovitost u podatkovnim centrima.

Učinkoviti dizajni poslužitelja usmjereni su na osiguranje da se potrošnja energije poslužitelja skalira s njihovim radnim opterećenjem. Energetski proporcionalno računalstvo uključuje projektiranje poslužitelja koji mogu dinamički prilagoditi svoju potrošnju energije na temelju trenutne potražnje.

To rezultira učinkovitim radom poslužitelja čak i u razdobljima niske iskorištenosti tako da smanjuje izgubljenu energiju.

#### 5.1.1.3 Obnovljivi izvori energije za napajanje podatkovnih centara

FAANG tvrtke su instalirale sustave za proizvodnju obnovljive energije izravno u objektima svojih podatkovnih centara. To uključuje postavljanje solarnih ploča, vjetroturbina ili druge infrastrukture za obnovljivu energiju na licu mjesta. Time smanjuju ovisnost o tradicionalnoj električnoj mreži i smanjuju emisije ugljika.

Ugovor o kupnji električne energije (PPA eng. Power Purchase Agreements). FAANG tvrtke sklapaju dugoročne ugovore s razvijačem obnovljivih izvora energije. Ovim sporazumi uključuju kupnju električne energije izravno iz projekata obnovljivih izvora energije kao što su vjetroelektrane ili solarne instalacije.

Velike tvrtke doprinose rastu obnovljive energije time da podržavaju i surađuju s vjetroelektranama i sličnim projektima.

### 5.1.2 Energetski učinkoviti algoritmi

Velike korporacije iz Silicijske Doline razvijaju algoritme na način da minimiziraju potročnju energije.

Jedan od primjera takve implementacije je primjena tehnika za optimizaciju *video streaming-a* od strane Netflixa. Primijenjene tehnike omogućuju prijenos videa na način da se visoka kvaliteta videa ne smanjuje dok se minimizira podatkovna veličina.

Kako bi postigli veliku kvalitetu videa uz što manju podatkovnu veličinu videa Netflix koristi napredne algoritme kompresije (eng. *compression algorithms*) kao što su VP9 I AV1. Kompresijom videa, Netflix smanjuje potrebni *bandwidth* što rezultira štednjom energije tijekom prijenosa videa.

Još jedna od primijenjenih tehnika je prilagodljivo strujanje brzine prijenosa (eng. *Adaptive Bitrate Streaming*). To znači da se kvaliteta videa prilagođava kvaliteti internetske veze. Prilagođavanjem kvalitete videa Netflix optimizira korištenje *bandwidth-a* i smanjuje gubitak podataka, a to sve rezultira štednjom energije potrebne za prijenos videa.

Tijekom šifriranja videa Netflix ne koristi jednu tehniku za sve videozapise, već za različite videozapise različitih karakteristika koristi različite tehnike šifriranja (eng. *encoding optimization technique*). [11]

### 5.1.3 Virtualizacija i računalni oblak (eng. cloud computing)

Tvrtke FAANG koriste tehnologije virtualizacije i računarstva u oblaku kako bi optimizirale korištenje resursa i smanjile potrošnju energije. Korištenjem oblaka i virtualizacije za servere, mrežnu opremu i baze podataka povećava se efikasnost korištenja resursa i štedi se na energiji.

Zašto korištenje računalnog oblaka rezultira time da se resursi efikasno koriste i kako korištenje oblaka rezultira time da se štedi energija?

Prvo, smanjuje se broj ukupno potrebnih fizičkih servera jer se kapacitet jednog server maksimalno iskorištava time da se na jednom serveru istodobno pokreće više virtualnih mašina i aplikacija. Nadalje, jedna od prednosti korištenja oblaka i virtualizacije je dinamička alokacija resursa (eng. *dynamic resource allocation*). Drugim riječima, resursi su alocirani odnosno dislocirani ovisno o njihovoj potražnji. Ovakva fleksibilnost omogućuje da se resursi koriste na efikasan način, da su aktivni samo onda kada su potrebni i time smanjuju energiju koja se troši. Važno je i spomenuti lokaciju velikih centara gdje se nalaze pružatelji usluga u oblaku (eng. *Could Providers*). Pružatelji usluga u oblaku strateški lociraju svoje podatkovne centre kako bi iskoristili obnovljive izvore energije i povoljne klimatske uvjete. Cilj im je minimizirati utjecaj svojih operacija na okoliš odabirom lokacija na kojima je obnovljiva energija u izobilju ili gdje se mogu koristiti prirodni mehanizmi hlađenja, kao što su hladna klima ili blizina vodenih tijela. To osigurava da je potrošnja energije optimizirana i da veći udio energije dolazi iz obnovljivih izvora.

Na sljedećoj slici vidimo kako je Google svoje podatkovne centre smjestio u blizini vjetrenjača, na pogodnoj lokaciji u blizini vodenih područja koja mogu poslužiti za hlađenje.

Slika na kojoj se prikazuje nebo, vanjski, trava, oblak

Opis je automatski generiran

*Google Data Center* [*url*](https://www.google.com/about/datacenters/)

U oblaku, korisnici imaju pristup alatima koji služe za praćenje i optimizaciju potrošnje energije. Administratori mogu pratiti potrošnju resursa, identificirati neučinkovite procese ili aplikacije i sukladno tome implementirati strategije upravljanja energijom.

Najbolji primjeri takvih tehnologija su AWS (Amazon Web Service) i Microsoft Azure. AWS i Azure su platforme za računarstvo u oblaku koje nudi Amazon i Microsoft. I AWS i Azure široko su korištene platforme u oblaku koje nude niz usluga i rješenja za tvrtke i programere. Korisnicima omogućuju pristup i upravljanje virtualiziranim računalnim resursima, kao što su virtualne mašine, pohrana, baze podataka i umrežavanje, na fleksibilan i skalabilan način.

### 5.1.4 Ulaganje u obnovljive izvore energij**e**

Tvrtke FAANG ulažu u projekte obnovljive energije kako bi tom energijom potaknule svoje poslovanje. Neki od primjera takvih investicija:

* Amazon se obvezao napajati svoje poslovanje 100% obnovljivom energijom do 2025 [12]
* Google je uspio napajati svoje podatkovne centre koristeći izvore električne energije koji ne proizvode emisiju ugljika oko 66% vremena [13]
* Apple je aktivno ulagao u projekte obnovljive energije, uključujući solarne farme i postrojenja za energiju vjetra. Tvrtka je uspostavila partnerstvo s programerima obnovljive energije i ima za cilj napajati svoje objekte širom svijeta 100% obnovljivom energijom [14]
* Microsoftova suradnja sa proizvođačem solarnih panela Qcells [15]

# Zaključak

Prakse green codinga nude značajnu priliku tehnološkoj industriji da doprinese održivijoj budućnosti. Integriranjem pitanja zaštite okoliša u proces razvoja softvera, programeri i organizacije mogu pozitivno utjecati na okoliš.

Green code uključuje usvajanje tehnika i strategija koje smanjuju potrošnju energije, smanjuju emisije ugljika i optimiziraju korištenje resursa. To može uključivati ​​prakse kao što su učinkoviti algoritmi, optimizacija koda, upravljanje napajanjem i korištenje tehnologija računalstva u oblaku i virtualizacije.

Integriranjem praksi green code-a u procese razvoja softvera, možemo stvoriti ekološki svjesniji i održiviji digitalni ekosustav za dobrobit sadašnjih i budućih generacija.

# 7. Literatura

[1] <https://www.parallels.com/blogs/ras/what-is-green-cloud-computing/>

[2] <https://www.ibm.com/cloud/blog/green-coding>

[3] <https://geekflare.com/green-coding/>

[4] <https://www.geeksforgeeks.org/virtual-machines-in-operating-system/>

[5] <https://www.geeksforgeeks.org/virtualization-cloud-computing-types/>

[6] <https://www.softpedia.com/get/System/System-Miscellaneous/Joulemeter.shtml>

[7] <https://www.pcguide.com/tips/intel-power-gadget/>

[8] <https://dev.to/roy8/how-to-optimize-your-code-for-performance-1km5>

[9] <https://www.energystar.gov/products/low_carbon_it_campaign/12_ways_save_energy_data_center/air_side_economizer>

[10] <https://www.techtarget.com/whatis/definition/liquid-cooling>

[11] <https://netflixtechblog.com/>

[12] <https://sustainability.aboutamazon.com/>

[13] <https://sustainability.google/reports/>

[14] <https://www.apple.com/hr/newsroom/2023/04/apple-and-global-suppliers-expand-renewable-energy-to-13-point-7-gigawatts/>

[15] <https://news.microsoft.com/2023/01/25/microsoft-and-qcells-announce-strategic-alliance-to-curb-carbon-emissions-and-power-the-clean-energy-economy/>