

**Veriga blokov kot storitev za beleženje dogodkov**

**UM FERI**

Maribor, April 2024

**KAZALO**

**KAZALO SLIK**

[Slika 5.1: Oblikovanje tipične strani. 8](#_3as4poj)

**KAZALO TABEL**

[Tabela 5.1: Primeri uporabe velikosti pisav. 7](#_1pxezwc)

**SEZNAM UPORABLJENIH SIMBOLOV IN KRATIC**

POW - Proof of Work (dokaz za delo)

POS - Proof of Stake (dokaz o deležu)

POA - Proof of Authority (dokaz avtoritete)

BaaS - Blockchain as a Service (veriga blokov kot storitev)

MPI - Message Passing Interface

Veriga blokov kot storitev za beleženje dogodkov

**Ključne besede:** veriga blokov, dokazilo o delu, MPI, prepleteni pristop

**UDK:** xxxxxx

**Povzetek**

*Analiziral sem različne tipe verig blokov, optimizacije in uporabo verige blokov kot storitev. Implementiral sem storitev verige blokov in uporabniško aplikacijo. Veriga blokov pridobiva informacije o dogodku preko protokola HTTP in nato tekmuje proti drugimi instancami storitve z algoritmom dokazilo o delu.*

Blockchain as a service for recording events

**Keywords:** Blockchain, proof of work, MPI, intertwined approach

**UDC:** xxxxxx

**Abstract**

*I analyzed different types of blockchains, optimizations, and the use of blockchain as a service. I implemented a blockchain service and a user application. The blockchain receives event information via the HTTP protocol and then competes with other service instances using the proof-of-work algorithm.*

# 1 UVOD

Projekt vsebuje veriženje blokov z namenom arhiviranja naravnih nesreč.

Problem ki ga poskušamo odpraviti je hitro in varno arhiviranje raznih vrst nesreč (cestne nesreče, naravne nesreče in drugi podobni dogodki). Potrebujemo decentralizirani sistem kjer lahko uporabniki beležijo, ko se zgodi neka nesreča, nato pa se te nesreče shranjujejo v eni verigi blokov. V kontrastu z drugimi podatkovnimi tipi, je veriga blokov tu lahko rešitev ki arhivira podatke iz različnih virov pri tem pa ohranjuje hitrost in integriteto, ker je verigo blokov nemogoče spremeniti ko je enkrat narejena, kar pomeni da se nobena nesreča ne more iz nje izbrisati, prav tako se ne morejo spreminjati podatki v njej.

Cilj je narediti spletno storitev, ki jo lahko kdorkoli na svojem računalniku zažene in deluje v ozadju. Nato lahko uporabnik z ustreznimi certifikati iz uporabniške aplikacije zabeleži dogodek nesreče na lokaciji kjer se je zgodila. Spletne storitve nato tekmujejo med sabo za zapisovanje dogodka v verigo blokov (ki je skupna vsem). Tekmujejo z uporabo algoritma POW.

V seminarski nalogi bom implementiral storitev za verigo blokov z algoritmom POW, nato bom naredil primer uporabniške aplikacije za beleženje nesreč. V naslednjih poglavjih bom opisal metodologijo in rezultate, ter podal mnenje o naslednjih korakih za razvoj tehnologije.

# 2 TEORETSKI PREGLED PODROČJA

Veriga blokov spada med eno izmed novejših tehnologij prvič predstavljena leta 1982 (David L. Chaum <https://cdn.sanity.io/files/r000fwn3/production/e630610a9c7c3c8f950cfb5632b44ecf1b8926d2.pdf>). Problem ki ga rešuje je integriteta podatkov ko si med seboj več računalnikov izmenjava podatke. Za dosego tega cilja uporablja kriptografske tehnike, ki omogočajo varno in zanesljivo shranjevanje ter prenos podatkov med različnimi točkami v omrežju. Tehnologija pa je postala bolj znana z idejo decentralizirane verige blokov za uporabo v finančnem sektorju, katere najbolj znan primer je Bitcoin.

## 2.1 Struktura bloka

Vsak blok je podatkovna struktura sestavljena iz raznih elementov, vsi pa imajo skupno vsaj:

* Podatek, v katerega ponavadi shranimo sporočilo, lahko pa tudi bajte.
* Časovna značka, ta je pomembna za določitev zmagovalne instance, saj ni smiselno da bi zmagala instanca, ki je dlje časa porabila verifikacijo transakacije.
* Zgoščena vrednost bloka in zgoščena vrednost prejšnjega bloka. To v verigi dva bloka enolično povezuje, zgoščenim vrednostim lahko sledimo v preteklost in tako vidimo zgodovino transakcij.

Veriga blokov je nespremenljiva, podatkov, ko so dodani v verigo, ne moremo več spreminjati, je torej tudi neke vrste arhiv podatkov.

## 2.1 Decentralizirana veriga blokov

Prva decentralizirana veriga blokov, Blockchain, je bila zasnovana 2008 (<https://bitcoin.org/bitcoin.pdf>). Opisuje finančni sistem kjer so transakcije definirane kot digitalni podpisi v verigi. Vsak nov digitalni podpis v verigi se mora uskladiti s prejšnjim podpisom, tj. vsak blok v verigi hrani podpis prejšnjega elementa verige. To samo po sebi še ne pomeni da je veriga blokov decentralizirana, lahko bi bila tudi centralizirana v eni entiteti (organizacija, računalnik, …). Ideja sistema Bitcoin je, da omogoča decentralizirano verigo blokov s pomočjo algoritma POW. S tem, ko vsak nov blok v verigi zahteva kompleksno računsko nalogo za dokazovanje svoje veljavnosti, se ustvari mehanizem, ki preprečuje manipulacijo ali nadzor nad celotno verigo. Tako se prepreči monopol nad procesom dodajanja novih blokov, kar omogoča, da je sistem decentraliziran. Vsakdo, ki želi sodelovati pri dodajanju blokov, lahko to počne, ne glede na svoj položaj ali moč. To pomeni, da ni ene same entitete, ki bi imela nadzor nad celotnim omrežjem, kar zagotavlja varnost in tudi zaupanje v sistem, saj majhna skupina uporabnikom ni zmožna porušiti sistem verige blokov. Decentralizirane verige blokov so pogosto uporabljene v finančnih sistemih, kjer potekajo finančne transakcije v kriptovalutnih mrežah. Kriptovalute so digitalna vrednost, ki same po sebi ne potrebujejo intrinzične vrednosti, imajo pa konvertibilnost v relaciji s fiat-vrednostjo.

## 2.2 Tipi verifikacije blokov

Za verifikacijo naslednjega bloka se zgodovinsko največ uporablja POW. Za rudarjenje (tj. zaporedno izvajanje algoritma POW) je ponavadi zmagovalec tudi nagrajen, saj porablja procesorsko moč ki troši energijo. POW deluje na osnovi zgoščevalne funkcije SHA (običajno SHA-256), kjer algoritem išče ustrezno število ki zadošča temu, da ima rezultat SHA podniz ki ustreza iskanemu. Ponavadi je ta podniz zaporedje ničel; število teh pa tudi definira stopnjo težavnosti rudarjenja. Stopnja težavnosti se dinamično spreminja glede na čas v katerem želimo da se nov blok dokaže. Za izračun SHA uporabimo podatke v bloku, primer bi bil indeks + časovna značka + podatek + prejšnja zgoščena vrednost, kjer + predstavlja konkatenacijo nizov. POW je najlažji za implementacijo, saj je koda le ena zanka v kateri v vsaki iteraciji računamo SHA z novim številom in novo časovno značko dokler ne dobimo rezultata, ki ustreza stopnji težavnosti, nato pošljemo podatek o novemu bloku na omrežje, kjer se nahajajo druge instance programa verige blokov, in če je blok ustrezen glede na nižjo časovno porabo, se vse druge verige blokov v omrežju posodobijo in začnejo rudariti nov blok, v katerem je prejšnja zgoščena vrednost enaka zgoščeni vrednosti ravnokar dodanega bloka.

Novejše verige blokov pa že uporabljajo razne druge koncepte kot so POS, kjer se za dokazilo uporabljajo ekonomski viri ki jih verifikator že ima. POS je okolju bolj prijazna metoda, ker ne računa funkcij SHA. Slaba stran te metode je, da potrebuješ neko vrsto minimalnega začetnega kapitala, v večini primerov je to kriptovaluta. Nekatere verige blokov, ki so v preteklosti uporabljale POW, zdaj uporabljajo POS, najbolj znan primer je Ethereum (<https://ethereum.org/en/developers/docs/consensus-mechanisms/pos/> ).

Nekatere privatne verige blokov uporabljajo tudi POA. V tem procesu so validatorji določeni vnaprej in ti so edini ki lahko ustvarjajo nove bloke transakcij, ki so dodani v omrežje. Mehanizem sprejemanja in rudarjenja novih blokov v omrežju je odvisen od konfiguracije. Zlonamerno vedenje v verigi je tu manjše, ker so validatorji vnaprej določeni. (https://arxiv.org/pdf/2109.02480.pdf).

## 2.3 Optimizacije in pohitritve verige blokov

Veriga blokov mora biti dovolj hitra, da se lahko bloki rešujejo v zmernem času in da lahko uporabimo vso procesorsko moč računalnika. Algoritem POW lahko preuredimo tako, da uporablja vse procesorske niti računalnika. Možna modifikacija algoritma je s prepletenim pristopom, kjer dodelimo vsaki niti del računanja s tem da jim damo vnaprej izračunane številne vhodne vrednosti zanke s katero računa kriptografski algoritem.

(<https://www.hindawi.com/journals/sp/2021/9996132/>). Pri algoritmu POS teh pohitritev ni in je v tem smislu lažji za dobro implementacijo, saj ni računanja algoritmov za dodeljevanje nagrad, ti so takoj določeni glede na ekonomske vire.

## 2.4 Veriga blokov kot storitev

Sama veriga blokov ni še storitev, postane pa storitev, ko omogoča da uporabnik pošlje na strežnik verigi podatke, ki jih ta potem shranjuje. Nudi poenostavitev uporabljanja verige blokov, saj je implementacija te ponavadi kompleksna, uporabnik pa se s tem ne želi ukvarjati . V industriji ta koncept uporabljajo storitve IBM Blockchain, Microsoft Azure Blockchain, Ethereum BaaS na Azure, AWS Blockchain. (<https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8840920> ). Veriga je nato lahko na oblaku, uporabniki odjemalne aplikacije pa lahko do nje dostopajo brez karkšnegakoli znanja o notranjem delovanju verige blokov, saj se vse izvaja v ozadju ki uporabniku ni direktno dostopen, vidi pa le rezultate.

## 2.5 Uporabe verige blokov

Poleg kriptovalut obstajajo še drugi primeri uporabe verige blokov.

V primerjavi s tradicionalnimi spletnimi bankami, je decentralizirana veriga blokov varnejša za uporabnike. Lažje je vdreti v centralizirane banke, kot pa v decentralizirane sisteme, saj lahko vdiralci vdrejo v centraliziran sistem preko ene same napake v sistemu, medtem ko v decentraliziranem sistemu (<https://theblockchaintest.com/uploads/resources/file-265154939225.pdf> )sodeluje več računalnikov, ki se morajo med seboj ves čas sinhronizirati in primerjati verige, kar pomeni da če en računalnik poskuša v verigo dodati nepravilen blok, to vidijo vsi drugi računalniki, ki preverijo pravilnost bloka in vidijo da ni pravilen, in torej ga ne dodajo v svojo verigo blokov. To pomeni, da se zlonamerna veriga blokov ne bo na noben način uspela sinhronizirati s pravilno verigo blokov.

Uporabljati se začenja tudi v KYC, kar je sistem, ki odobri istosvetnost oseb. V preteklosti so se za to uporabljali zelo dolgi postopki, doba digitalizacije pa je ta problem delno že rešila, saj so postopki identifikacije preko spleta krajši kot preko fizičnega obiska v ustanovah. Veriga blokov bi lahko še dodatno pospešila ta postopek, saj bi se lahko informacije o identiteti shranjevale v javni verigi blokov, razne organizacije pa bi lahko isto verigo blokov uporabljali za namene zelo hitre verifikacije, saj bi takoj vedele da je oseba že dokazala svojo identiteto (kako je odvisno od implementacije). Platforma ki to že uporablja je SWIFT, študije pa so pokazale da lahko ta način KYC bankam prišpara 160 milijonov ameriških dolarjev letno.

V zadnjih letih je veliko podjetij začelo uporabljati verigo blokov v oskrbovalnih verigah. Veriga blokov lahko poenostavi veliko problemov koordinacije in organizacije produktov in osebja. Vsaka transakcija in premik produktov je beležena sprotno, vsak pa lahko javno vidi pomanjkljivosti in napake ki so se pri tem zgodile. Pametne pogodbe pa lahko ta proces tudi avtomatizirajo; izraz označuje programersko kodo, ki se avtomatsko izvrši glede na neko stanje v verigi blokov. To pomeni, da lahko sprotno odpravljamo in obveščamo o napakah v oskrbovalnih verigah.

Veriga blokov postaja tudi zanimiva za državne institucije, v zadnjih letih se razvijajo sistemi digitalnega glasovanja, ker se glasovi v primerjavi s fizičnim glasovanjem ne morejo narobe prešteti, glasove izgubiti, rezultati pa so lahko takojšnji. Veriga blokov s primerno enkripcijo in s sistemom privilegijev, kjer je ne more do vsakega podatka dostopati vsak je lahko uporaben tudi za zdravstvene arhive, osebno identiteto, digitalne potne liste ipd. Za standardizacijo in lahek dostop do podatkov bi lahko imeli tudi mednarodne organizacije, ki skrbijo za te podatke (<https://www.researchgate.net/publication/346805616_Blockchain_Technology_Emerging_Applications_and_Use_Cases_for_Secure_and_Trustworthy_Smart_Systems>).

# 3 METODOLOGIJA

## 3.1 Cilji in teze

Cilj je narediti čimbolj procesorsko učinkovito in uporabnikom prijazno verigo blokov ki je sposobna pridobivati informacije iz uporabniških aplikacij. Ko program dobi informacijo, mora tekmovati z drugimi instancami programa za verifikacijo v verigo blokov.

Poleg cilja izdelave programov, želim v seminarski nalogi odgovoriti tudi na vprašanja:

* Lahko dosežemo asinhronost procesiranja informacij z uporabo več instanc storitve in ali se lahko med sabo po procesiranju dogodka storitve spet konsistentno sinhronizirajo?
* Katera metoda optimizacije verige blokov je bolj učinkovita, MPI ali prepleteni pristop?
* Kako blizu linearni učinkovitosti pohitritve lahko pridemo z MPI in s prepletenim pristopom?

## 3.2 Orodja

Za izbor programskega jezika in drugih orodij moram upoštevati več faktorjev.

Prvi je hitrost jezika, ta mora biti dovolj hiter, saj nočemo da je preveč časa porabljeno na procesiranju posameznega dogodka, ker želimo da se ob prijavi dogodka nesreče karseda hitro ta tudi zabeleži kot blok. Druga plat hitrosti jezika je ponavadi tudi manjša poraba energije. Manj časa in inštrukcij kot potrebuje procesor za procesiranje posameznega bloka, manj energije porablja. Odločil sem se za C++ ker je eden izmed najbolj energijsko učinkovitih jezikov, prevede pa se v strojno kodo, kar pomeni večjo hitrost izvajanja kot skriptni jeziki (<https://www.sciencedirect.com/science/article/pii/S0167642321000022>).

Slaba plat izbire je, da koda ne more biti zagnana na vseh možnih naprav brez ponovnega prevajanja kode v strojno kodo. Ker uporabljam koncept verige blokov kot storitev je predvideno da imajo uporabniki ki zaganjajo kodo potrebno tehnično znanje da lahko sami izvedejo prevajanje kode. Da bi karseda poenostavil ta proces sem uporabil CMake, ki zagotavlja da lahko kodo prevedemo le z zagonom skripte CMakeLists.txt, sam proces grajenja kode v strojno kodo pa je nato popolnoma avtomatiziran.

Uporabnik ki želi zagnati novo instanco verigo blokov mora to kodo iz nekje pridobiti. Odločil sem se da bo koda javna na strani Github za deljenje kode. Omogoča tudi komentiranje hroščev v kodi s strani uporabnikov, kar je za projekt kjer je pomembna varnost dobro, saj ne želimo imeti nikakršnih napak, ki bi omogočale porušitev verige blokov.

Za uporabniško aplikacijo bom dodal možnost beleženja dogodkov v že obstoječo mobilno aplikacijo.

## 3.3 Arhitektura verige blokov

Potrebna je odločitev med POW in POS. Odločil sem se za POW, ker uporaba ekonomskih virov (pogosto kriptovalute) tu ni potrebna. Za optimizacijo implementacije POW sem uporabil metodo prepletenega pristopa v kombinaciji z MPI. Prepleteni pristop uporablja niti procesorja, medtem ko pristop MPI uporablja različne instance programa, ki komunicirajo med seboj, zato lahko kombiniram oba pristopa. V program je potrebno vključiti tudi HTTP strežnik za pridobivanje informacij iz uporabniške aplikacije. Na koncu je potrebno razviti za testiranje tudi uporabniško aplikacijo, ki podatke pošilja na verigo blokov v namene beleženja dogodkov.

# 4 REZULTATI

Naredil sem storitev verige blokov in uporabniško aplikacijo. Iz uporabniške aplikacije se pošiljajo informacije o dogodku na verigo blokov.

## 4.1 Veriga blokov

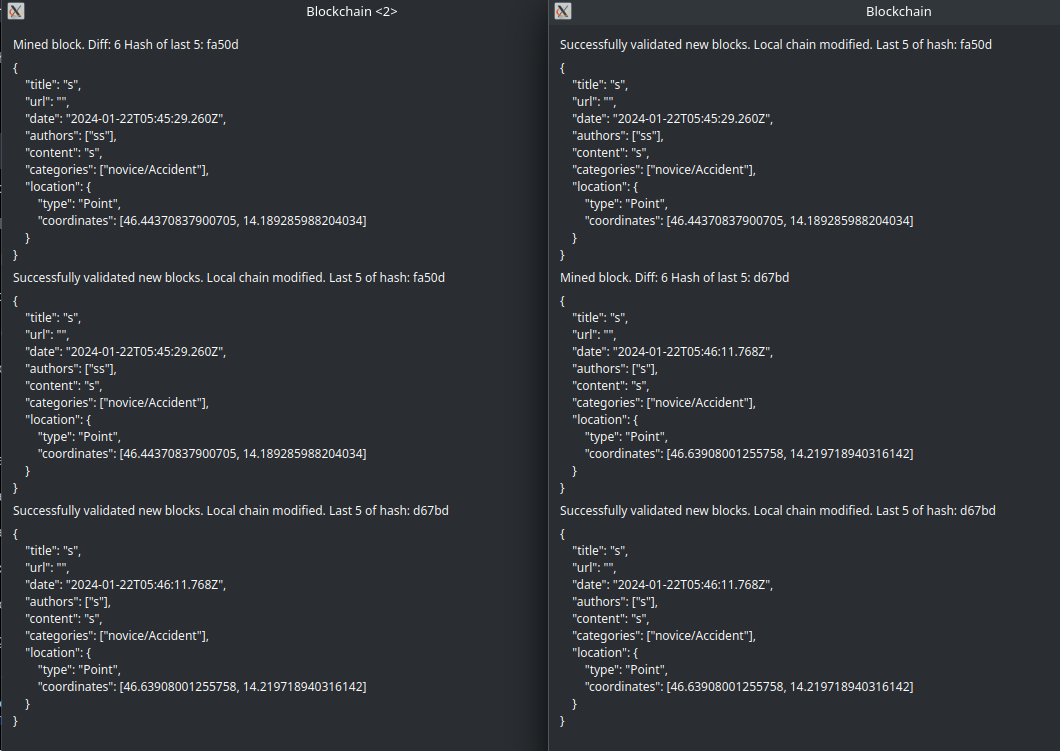
Sestava bloka je zelo podobna projektu Bitcoin, saj je tudi tam uporabljen algoritem POW. Vsebuje podatke:

* ID, ki hrani zaporedno število bloka v verigi.
* Datum nastanka bloka.
* Podatek, ki je razdeljen na več delov.
* Zgoščena vrednost prejšnjega bloka.
* Zgoščena vrednost bloka.
* Stopnja težavnosti s katero je bil blok narejen.
* Zmagovalno število, s katero je bila izračunana zgoščena vrednost.

V verigi blokov se kot podatek hrani posebno formatiran JSON, popularna notacija za označevanje podatkov in omogoča standardizacijo med tehnologijami in programskimi jeziki. Vsebuje podatke:

* Naslov, ki hrani informacijo o naslovu dogodka.
* URL, ki hrani naslov, če je dogodek iz spletnega vira.
* Datum, ko se je dogodek zgodil.
* Avtorji, ki so dogodek objavili.
* Vsebina, razlaga o dogodku.
* Kategorije, ki označujejo tip dogodka, in omogočajo grupiranje dogodkov in iskanje istih tipov dogodkov.
* Lokacija, ki je izražena v koordinatih in označuje, kje se je dogodek zgodil.

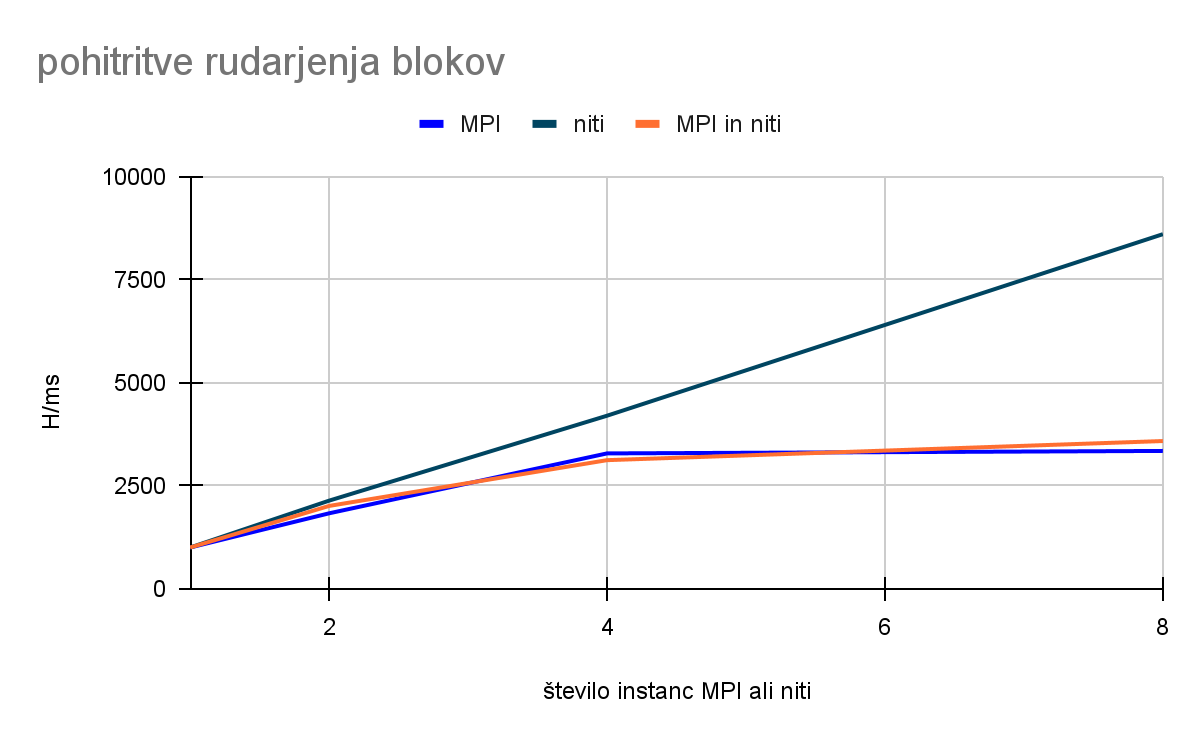
Instance verige blokov se med seboj v omrežju sinhronizirajo. V sliki je videno, da leva veriga blokov prva najde nov blok, pošlje ta blok vsem v omrežju, kjer ga desna instanca sprejme in opravi verifikacijo pravilnosti. Ugotovi da je blok pravilen, leva stran je o tem tudi obveščena. Skupaj nato iščeta naslednji blok, ki ga tokrat prva najde desna instanca. Ta jo spet posreduje levi, ta preveri pravilnost bloka in spet se začne iskanje novega bloka.



Naredil sem tudi avtomatsko prilagajanje stopnje težavnosti, ki ublaži problem premalega ali prevelikega števila instanc verige blokov, ki skupaj rudarijo bloke. V prvem primeru rešuje problem tega, da je lahko čas za nastanek novega bloka veliko prevelik, saj je manjša možnost da najdemo rezultat hitro z manjšim številom rudarjenja. Če je preveč instanc in zelo majhna težavnost, pa to pomeni da bodo bloki najdeni prehitro, kar lahko povzroči zmedo v sistemu, saj bi bilo rudarjenje hitrejše kot sama izmenjava blokov med sabo, kar bi privedlo do tega, da je zmagovalna instanca vedno v prihodnosti tudi zmagovalna, ker izračuna zgoščeno vrednost novega bloka še preden pride prejšnji blok do drugih instanc v omrežju. Avtomatsko se posodobi težavnost rudarjenja na vsakih 10 blokov.

## 4.2 Optimizacije verige blokov

Pohitritve s prepletenim pristopom so zelo učinkovite, metoda MPI pa je občutno manj učinkovita. Pri 2 nitih je pohitritev 2.1-kratna, pri štirih pa 4.1-kratna, ta linearnost se ohrani vse do 8 niti (maksimalno testiranih). Pri 2 instancah MPI je pohitritev 1.8-kratna, vendar pa se učinkovitost zelo hitro preneha dvigovati, in ostane na okoli 3.2-kratnosti pri 4 ali več instancah MPI.



## 

## 4.3 Uporabniška aplikacija

Izdelal sem uporabniško aplikacijo, ki omogoča uporabniku prijazen vnos dogodka v verigo blokov. Veriga blokov je torej abstrakcija do katere uporabnik sam ne dostopa, vnese le podatke, ki se nato avtomatsko pretvorijo v JSON, ta pa se nato pošlje v verigo blokov preko protokola HTTP na verigo blokov, ki ta podatek posreduje vsem članom omrežja, nato pa ti tekmujejo za validacijo in shranjevanje.

Aplikacija je mobilna in deluje na sistemu Android, program pa je napisan v programskem jeziku Kotlin z uporabo Android Studio. Omogoča vnos naslova, avtorjev in sporočila, ima pa tudi vnaprej nastavljene kategorije, kot so nesreča, vreme, temperatura. Lokacijo vneseš s pomočjo zemljevida, kjer označiš koordinato. Zemljevid omogoča približevanje in oddaljevanje in je centriran na Slovenijo, da omogoča čim hitrejšo izbiro lokacije.

## 

# 5 DISKUSIJA

Odgovor na hipotezo ali je bolj učinkovit MPI ali prepleteni pristop je, da je prepleteni pristop bolj učinkovit. Potrdilno sem odgovoril na hipotezo da lahko pridemo blizu linearne učinkovitosti s prepletenim pristopom, ovrgel pa sem da je to mogoče tudi z MPI. Menim da je to zelo težko z MPI, saj je potrebno pošiljati podatke med različnimi instancami MPI, prepleteni pristop pa tega ne zahteva, ampak nam omogoča da nastavimo vrednosti le pred začetkom algoritma POW.

Uspešno je tudi sinhroniziranje instanc med sabo, to drži za instance narejene z MPI in za instance kot različni računalniki. Verige blokov se lokalno sinhronizirajo, ko zaznajo da obstaja v omrežju veriga boljša od lokalne verige.

# 6 ZAKLJUČEK

Uspešno sem izdelal aplikacijo verige blokov in mobilno uporabniško aplikacijo za beleženje dogodkov v verigo blokov glede na cilje naloge.

Več pozornosti bi bilo potrebno v prihodnosti nameniti uporabniški aplikaciji, saj ni določenih privilegijev, kdo lahko vpisuje podatke v verigo blokov, kar bi lahko prišlo do zlonamernih podatkov.

Za nadaljne izboljšave v delovanju verige blokov bi lahko odstranili MPI in s tem pridobili možnost testiranja drugih algoritmov, ki bi mogoče bili boljši od MPI, kot so OpenCL, kjer računajo POW jedra grafične procesne enote.

# 7 LITERATURA IN VIRI