UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Anže Bergant

Simulacija molekulske dinamike na visokozmogljivih infrastrukturah

DIPLOMSKO DELO NA UNIVERZITETNEM ŠTUDIJU

Mentor: doc. dr. Mojca Ciglarič

Somentor: dr. Janez Mavri

Ljubljana, 2016

Tema diplomskega dela

Opišite pojem visoke zmogljivosti in navedite arhitekturne značilnosti sistemov, ki jih označujemo kot visoko zmogljive. Pojasnite pojem skalabilnosti, ga definirajte in opišite metriko, s katero lahko vrednotimo skalabilnost. Pojasnite problematiko simulacije molekulske dinamike. S pomočjo paketa Amber izvajajte simulacijo molekulske dinamike hidratiranega encima monoamin oksidaza (MAO) na različnih računalniških platformah in različnih procesorjih z uporabo enega do šestnajstih jeder in z uporabo ene in dveh GPE. Ocenite skalabilnost in podajte vodila za izbiro ustrezne infrastrukture za nadaljnje izvajanje simulacij.

Iskreno se zahvaljujem svoji mentorici doc. dr. Mojci Ciglarič za pomoč in vse koristne nasvete pri izdelavi diplomske naloge.

Posebna zahvala gre somentorju dr. Janezu Mavriju za usmerjanje tekom izdelave diplomske naloge in za pomoč pri interpretaciji kemijskih rezultatov. Hvala tudi ostalim članom Odseka za računsko biokemijo in načrtovanje učinkovin na Kemijskem inštitutu – Kaji, Urški in Deniju. Hvala tudi Aleksandri za pomoč pri izvedbi molekularnih simulacij.

Zahvalil bi se še partnerki Anji za vso podporo, razumevanje, spodbujanje in potrpežljivost tekom pisanja diplomske naloge ter hčerki Zoji in še nerojenemu članu naše družine za vse vesele trenutke.

Nazadnje bi se zahvalil še staršem za vso podporo tekom študija.

Kazalo

Seznam uporabljenih kratic

Amber	Assisted Model Building with Energy Refinement
CPE	Centralna procesna enota
FAD	Flavin-adenin dinukleotid
FLOPS	Floating-point operations per second
GFLOPS	Giga FLOPS
GPE	Grafična procesna enota
GPGPU	General-purpose computing on graphics processing units
HPC	Visoko zmogljivo računalništvo (High-Performance Computing)
MAO	Monoamin-oksidaza
MD	Molekulska dinamika
MPI	Message Passing Interface
PEA	Feniletilamin (Phenethylamine)
PCIe	Peripheral Component Interconnect Express
PDB	Baza struktur proteinov (Protein Data Bank)
PM	Pretočni multiprocesor
SIMT	Single-instruction, multiple-thread
VMD	Visual Molecular Dynamics

Povzetek

Diplomsko delo zajema primerjavo različnih računalniških platform visoko zmogljivega računalništva pri simulacijah molekulske dinamike, ki spada med računsko zelo zahtevne probleme in potrebuje veliko procesorsko moč. Naš namen je bil kritično ovrednotiti različne platforme pri reševanju molekulske dinamike, zato smo simulacije izvajali od eno jedrnega do 16 jedrnega procesorja na računalniški gruči in z uporabo ene in dveh grafično procesnih enot (GPE). Rezultati bodo služili kot pomagalo pri načrtovanju nabave nove strojne opreme za izvajanje biomolekularnih simulacij.

Kot merilo smo uporabili čas, ki je potreben za izvedbo simulacije, ter kako dobro so platforme skalabilne. Pri primerjavi hitrosti se v biomolekularnih simulacijah uporablja mera ns/dan, ki nam pove, koliko nanosekund simulacije je sistem sposoben izračunati v enem dnevu.

Za potrebe diplomskega dela smo simulirali velik hidratiran protein MAO B z endogenim substratom feniletilaminom. Simulirani sistem je izjemno relevanten za področje nevroznanosti, saj ta encim regulira nivoje biogenih aminov, ki imajo ključno vlogo pri prenosu živčnega signala.

Ugotovili smo, da se simulacija molekulske dinamike občutno hitreje izvaja na GPE kot na klasičnih procesorjih. Poleg tega je tudi cenovno najbolj ugodna platforma za izvajanje klasičnih molekulskih simulacij. Z vidika skalabilnosti pa je smiselna uporaba le ene GPE hkrati, saj je pohitritev ob uporabi dveh GPE v primerjavi z eno slaba.

Ključne besede: visoko zmogljivo računalništvo, skalabilnost, CPE, GPE, encim MAO B, molekulska dinamika

Abstract

This thesis covers comparison between different computer platforms of high performance computing while performing molecular dynamics simulations, which falls under very complex problems and needs lots of processing power. Our goal was to critically evaluate different platforms while solving molecular dynamics, so we used 1 to 16 processor cores on a computer cluster and one and two graphics processing units (GPU) for simulations. The results will be used while planning on buying new computer hardware for biomolecular simulations.

We used time needed for simulations and platform scalability as our benchmarks. For comparing speed in biomolecular simulations we used ns/day for comparison. Ns/day tells us how many nanoseconds is a system capable of simulating in one day.

For this thesis we simulated a large hydrated MAO B protein with endogen phenylethylamine substrate. The simulated system is extremely important for neuroscience, since it regulates levels of biogenic amines, which have an essential part in neuro signal transmitting.

The results have shown us that the use of GPUs is significantly faster than regular processors when it comes to molecular dynamics. Moreover it is also the most cost effective platform for classical molecular dynamics. From the perspective of scalability it makes sense to only use one GPU at the time, since the speed-up when using two GPUs is lower than expected.

Key words: high performance computing, scalability, CPU, GPU, MAO B enzyme, molecular dynamics

1 Uvod

Visoko zmogljivo računalništvo (HPC – High Performance Computing) imenujemo skupke računalnikov, ki so sposobni hitrega in učinkovitega reševanja kompleksnih računskih problemov. Take probleme srečamo na vseh področjih znanosti, med katere sodijo kvantna mehanika, molekularno modeliranje, napoved vremena, fizikalne simulacije,… Vsem tem problemom je skupna velika računska zahtevnost in bi bili brez velike procesorske moči praktično nerešljivi. V preteklih letih se je pri reševanju teh problemov kot alternativa pojavilo računanje s pomočjo splošno namenskih grafično procesnih enot (GPGPU – General-purpose computing on graphics processing unit). Grafično procesne enote (GPE) so posebna tiskana vezja, ki so bila razvita za pospeševanje izrisa slik na računalniškem monitorju. GPE ponujajo ogromno računsko moč, saj vsebujejo veliko manjših procesorskih jeder (danes tipično več tisoč) in s tem omogočajo veliko istočasnih izračunov. Ravno zato so se izkazala kot primerno orodje pri reševanju znanstvenih problemov.

Pri razvoju in načrtovanju skupkov računalnikov igra pomembno vlogo skalabilnost. Skalabilnost nam pove, kakšna je pohitritev računalniškega sistema, če mu povečamo kapaciteto v smislu večjega števila procesorjev ali povečanja velikosti pomnilnika.

V diplomskem delu bomo v sodelovanju s Kemijskim inštitutom v Ljubljani izvedli kemijske simulacije na različnih računalniških platformah in sicer na različnih procesorjih z uporabo enega do šestnajstih jeder in z uporabo ene ali dveh GPE. Uporabili bomo tri različne procesorje: AMD Opteron 6182, Intel Xeon E5620 in Intel Xeon E5520. Uporabljen GPE bo Nvidia GeForce GTX 780, ki vsebuje 2304 računskih jeder. Odločili smo se, da bomo simulirali molekulsko dinamiko hidratiranega encima monoamin-oksidaza B (MAO B). MAO je protein, natančneje encim, ki sodeluje pri presnovi biogenih aminov (živčni prenašalci), kot so serotonin, noradrenalin in dopamin. Presnova zajema kemijske reakcije, ki jih encim kot katalizator pospešuje, saj znižuje energijsko bariero. Prekomerno delovanje encima vodi v povečano razgradnjo in posledično nizke nivoje živčnih prenašalcev, kar je osnova za številne bolezni, kot sta depresija in Parkinsonova bolezen. Molekulska dinamika nam omogoča vpogled v obnašanje bioloških makromolekul, kot je encim MAO. Simulacija molekulske dinamike proteinov nam daje odgovore na ključna vprašanja o strukturi in obnašanju proteina ter pomaga pri razvoju novih zdravil.

Izbran simulirani sistem predstavlja računsko zahteven problem, saj ga sestavlja 75545 atomov. Naša naloga je ustrezna priprava simulacijskega protokola, ki ga sestavljajo priprava vhodnih parametrov, ustrezen potek simulacije in pravilna interpretacija rezultatov.

Naš namen je kritično vrednotenje različnih platform za simulacije molekulske dinamike velikega hidratiranega proteina. Rezultate bomo ovrednotili iz vidika skalabilnosti in vzpostaviti vodila in napotke za razvoj infrastrukture v prihodnosti.

Naše ugotovitve bodo ključnega pomena pri načrtovanju nabave nove strojne opreme za izvajanje kemijskih simulacij.

2 Visoko zmogljivo računalništvo

Najhitrejšim in najzmogljivejšim računalnikom, ki so na voljo, pravimo superračunalniki. To so naprave z več sto tisoči procesorji, njihovo zmožnost računanja pa merimo v več sto milijardah operacij na sekundo. Sestavljeni so iz več manjših računalnikov, ki so med sabo povezani s hitrimi lokalnimi povezavami. Ti računalniki nato sodelujejo pri hitrem reševanju najzahtevnejših znanstvenih in inženirskih problemov današnjega časa. Uporabo superračunalnikov in tehnik paralelnega procesiranja za reševanje kompleksnih računskih problemov imenujemo lahko tudi visoko zmogljivo računalništvo [1].

2.1 Hitrost super računalnikov

Ena od metrik za vrednotenje zmogljivosti superračunalnikov je število izvršenih operacij s plavajočo vejico v sekundi (FLOPS – Floating-point operations per second). Zapis s plavajočo vejico je zapis realnih števil v formatu:

$$m \times r^e \tag{1}$$

V enačbi 1 je *m* mantisa, *r* baza in *e* eksponent. Primer zapisa vidimo v enačbi 2:

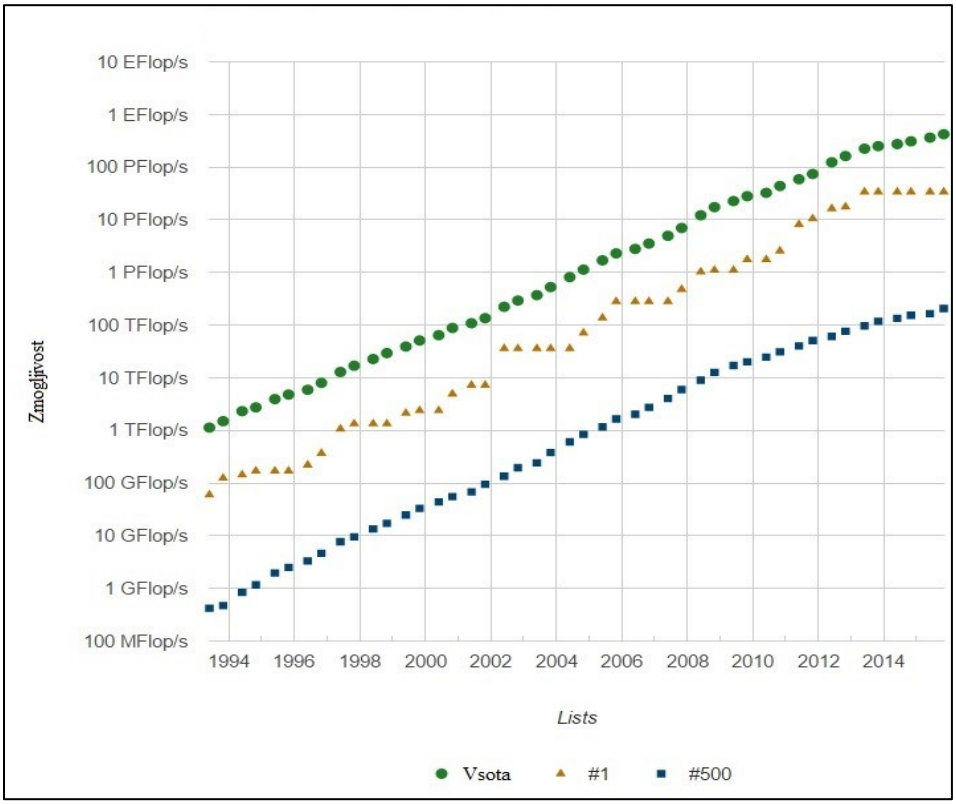
$$1{,}2345 = 12345 \times 10^{-4} \tag{2}$$

Eksponent nam pove, kje v mantisi leži decimalna vejica. S spreminjanjem eksponenta se vejica premika levo/desno po mantisi, zato tudi poimenovanje plavajoča vejica.

FLOPS izračunamo:

$$FLOPS = vozlišča \times \frac{jedra}{vozlišče} \times hitrost \times \frac{FLOPS}{cikel} \tag{3}$$

Hitrosti najhitrejših superračunalnikov se danes merijo v petaFLOPS-ih (peta = 10^{15}). Trenutno najhitrejši superračunalnik je kitajski Tianhe-2, ki uporablja 1600 računalniških vozlišč in 3.120.000 jeder za hitrost 33.86 petaFLOPS [2]. Slika 1 nam kaže rast hitrosti računanja najzmogljivejših superračunalnikov v zadnjih 20 letih. Vidimo linearno rast na logaritmični lestvici, kar dejansko pomeni eksponentno rast hitrosti računanja.

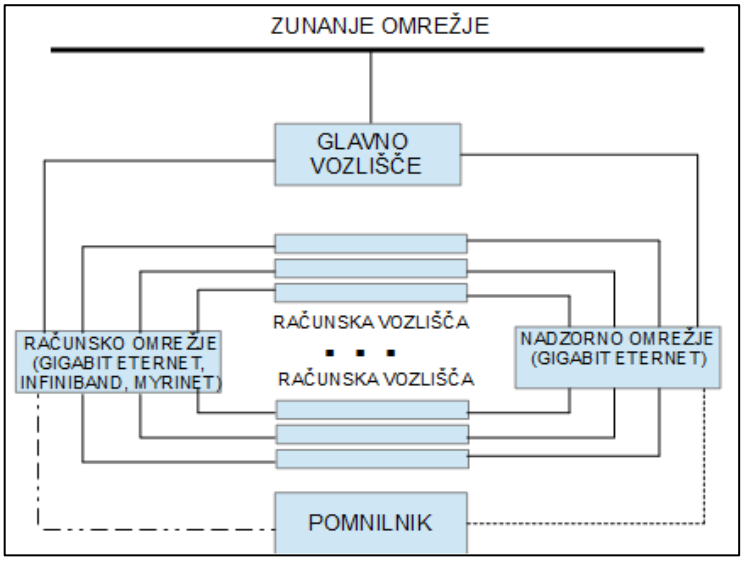


Slika 1: Razvoj hitrosti visoko zmogljivih računalnikov [2]. Graf s kvadrati prikazuje 500. najhitrejši HPC, graf s trikotniki najhitrejšega, graf s krogi pa seštevek 500 najhitrejših.

2.2 Vrste visoko zmogljivih računalnikov

Računalniška gruča (HPC cluster)

Računalniška gruča je skupek med seboj povezanih računalnikov (vozlišč – nodes), ki so sposobni delovati tako, da so uporabniku videti kot en sam sistem. Posamezna vozlišča so med seboj povezana s hitrimi lokalnimi povezavami in imajo lasten operacijski sistem. Običajno vsa vozlišča uporabljajo enako strojno opremo in operacijski sistem. Na Sliki 2 vidimo tipično arhitekturo gruče.



Slika 2: Arhitektura gruče.

Oblak (Cloud)

Računalništvo v oblaku je model računalništva, ki omogoča dostop do množice vseprisotnih, priročnih in nastavljivih računalniških virov, ki so v skupni rabi. Ti viri so lahko strežniki, aplikacije, omrežja ali storitve in so zlahka določeni ter dosegljivi. Uporabnik dostopa do virov z minimalnim sodelovanjem in interakcijo s ponudnikom storitev [3].

Oblak igra pomembno vlogo v znanosti predvsem v tem, da omogoča znanstvenikom, da sami določijo in nastavijo virtualne naprave tako, da so primerne za vsak posamezen primer posebej. Primeren je za začetne raziskave, testiranja in manj obsežne eksperimente ter za paralelno reševanje problemov, ki ne potrebujejo veliko medsebojnega komuniciranja (in s tem veliko prenašanja podatkov preko interneta). Ravno hitrost prenosa podatkov je glavna slabost oblaka v primerjavi z gručami [4].

Definicija oblaka obsega njegove bistvene značilnosti, modele storitev, ki jih oblak ponuja in možne modele postavitve infrastrukture [3] [5].

Bistvene značilnosti računalništva v oblaku so:

Storitve na zahtevo: Uporabnik lahko po potrebi dostopa do mnogih računalniških storitev (kot so uporaba strežnika ali dostop do omrežne shrambe) avtomatsko, brez potrebe po interakciji z vsakim posameznim ponudnikom storitev.

Širok dostop do omrežja: Zmožnosti so dosegljive preko omrežja, dostop je odjemalcu omogočen z različnih platform, kot so mobilni telefoni, tablice, prenosni računalniki ali delovne postaje.

Združevanje virov: Ponudnik storitev lahko svoje vire dinamično združuje ali razdružuje tako, da zadovolji uporabnikove zahteve. Viri niso omejeni z lokacijo, tudi uporabniku so ti podatki neznani in nepomembni. Primeri takih virov so shramba podatkov, procesna moč in pomnilnik.

Hitra prilagodljivost: Viri so lahko določeni ali sproščeni elastično, običajno avtomatično in se širijo ali krčijo sorazmerno z zahtevami. Uporabniku se zmožnosti sistema pogosto zdijo neomejene in dostopne kadarkoli.

Modeli storitev:

Programska oprema kot storitev: Uporabnik uporablja ponudnikove aplikacije, ki tečejo na oblaku. Aplikacije so dostopne z različnih naprav, lahko preko spletnega brskalnika (kot na primer spletna elektronska pošta) ali pa preko programskega vmesnika.

Platforma kot storitev: Uporabniku je ponujena možnost, da na oblaku postavi (narejeno ali kupljeno) aplikacijo, narejeno s programskim jezikom, knjižnicami, viri in orodji, ki jih podpira ponudnik. Uporabnik ne upravlja oblakove infrastrukture, ima pa kontrolo nad postavljenimi aplikacijami in njihovimi nastavitvami.

Infrastruktura kot storitev: Sposobnost ponujena uporabniku, da določa procesorsko moč, pomnilnik, mrežo in druge osnovne računalniške vire, na katerih lahko uporabnik postavlja in uporablja svojo programsko opremo, lahko tudi operacijske sisteme.

Modeli postavitve:

Zasebni oblak: Infrastruktura oblaka je določena izključno za uporabo ene organizacije, znotraj katere je več uporabnikov. Oblak si lahko lasti in upravlja organizacija sama ali pa nekdo tretji. Obstaja lahko znotraj ali zunaj prostorov organizacije.

Oblak skupnosti: Infrastruktura oblaka je določena izključno za uporabo skupnosti uporabnikov iz različnih organizacij, ki pa imajo skupne skrbi (npr. varnostne zahteve, poslanstva, politiko delovanja,…).

Javni oblak: Infrastruktura je primerna za javno uporabo. Lahko je v lasti in upravljanju podjetja, akademske ali vladne organizacije. Oblak obstaja v prostorih ponudnika.

Kombinirani oblak: Infrastruktura je sestavljena iz dveh ali več različnih oblakov (zasebnih, skupnih ali javnih), ki ostajajo samostojni, vendar jih povezujejo tehnološki standardi, ki omogočajo pretok podatkov in aplikacij.

Grid

Grid običajno pomeni skupek različnih virov, ki se raztezajo na več lokacijah. Velikost grida je lahko zelo različna - lahko pokriva različne oddelke v organizaciji, ali pa se razteza preko velikega števila virov na različnih lokacijah, razkropljenih po celem svetu [6]. Viri so običajno

heterogeni (torej različne arhitekture, operacijski sistemi, omrežja). Uporabljajo se večinoma v različnih raziskovalnih skupnostih, združenih v virtualne organizacije. V večini primerov so javno financirani, lahko na lokalni, državni ali mednarodni ravni. Velikokrat se uporabljajo podatkovni modeli, ki jih je težko reproducirati (kot na primer medicinski podatki, dostopni samo za bolnišnice) [7].

Glavna lastnost grida je sposobnost združevanja virov iz različnih organizacij s ciljem reševanja skupnih nalog [5]. To združevanje bistveno poveča velikost virov (viri so lahko tako programski kot strojni), ki so na voljo organizacijam. Storitvam grida so dodani vmesniki, ki skrijejo razlike med vsebovanimi viri, kar pomeni, da grid ponuja možnost virtualizacije vseh različnih delov v en velik skupek virov. Virtualizacija pokriva podatke (datoteke, baze podatkov) in računalniške vire [7].

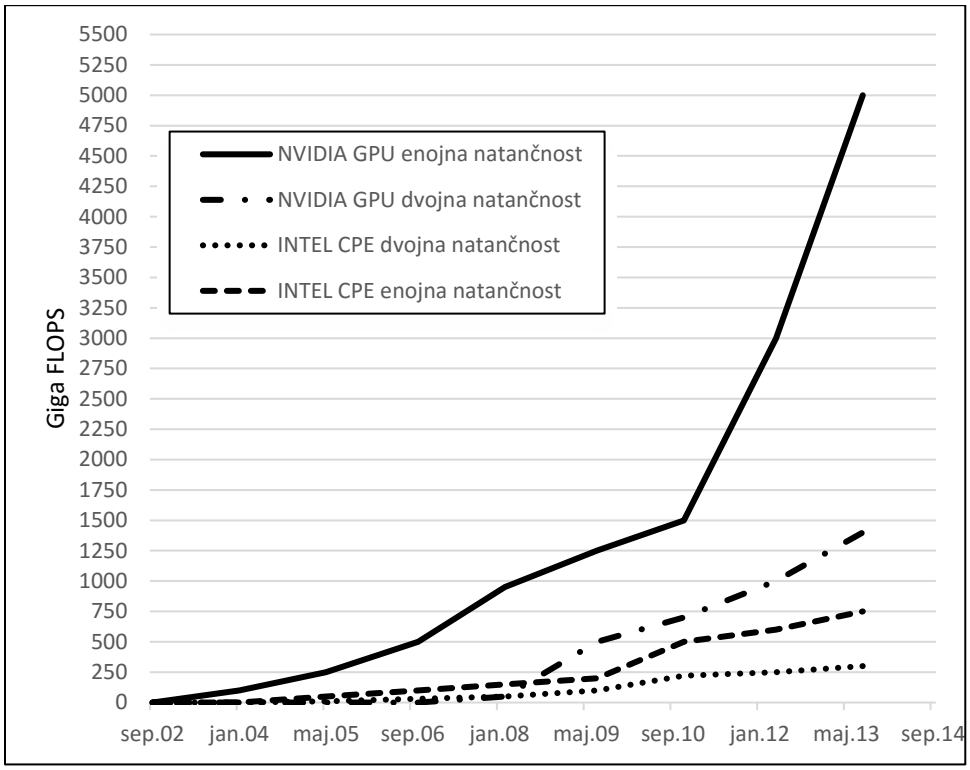
Skalabilnost grida je v glavnem omogočena s povečevanjem delovnih vozlišč. S tega vidika se končnim uporabnikom ni potrebno ukvarjati s problemom skalabilnosti, kot bi bilo potrebno na enem, tradicionalnem računalniku [6].

2.3 Grafična procesna enota

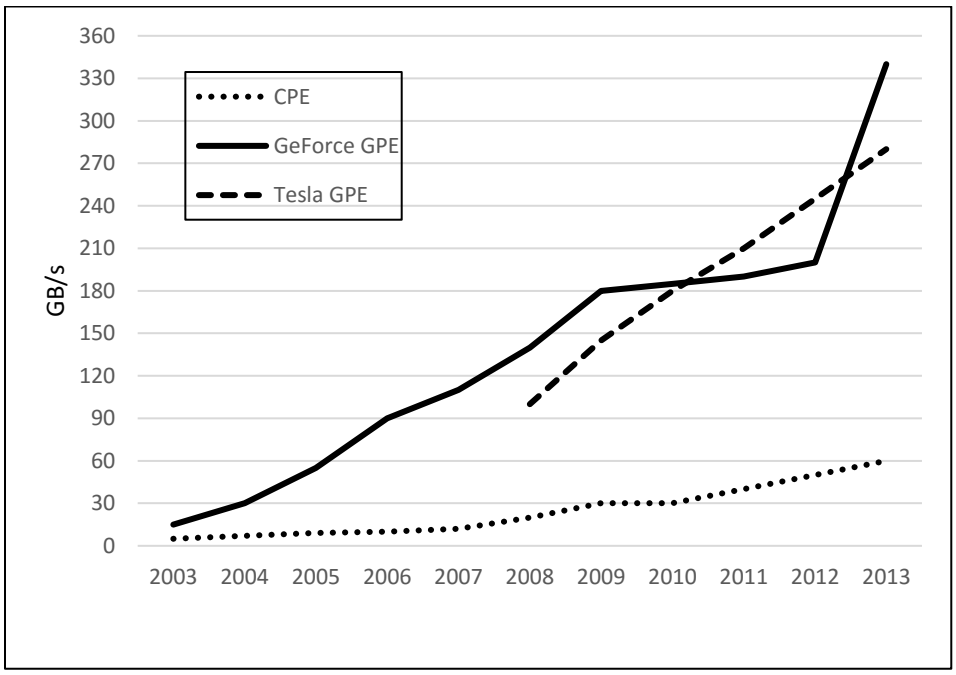
Grafična procesna enota (GPE) je posebno tiskano vezje, zasnovano tako, da pospešuje izris slik za prikaz na računalniškem monitorju. GPE najdemo v osebnih računalnikih, tablicah, telefonih, igralnih konzolah, …

V prvih letih računalništva je izračune za izris slik opravljala centralna procesna enota (CPE), vendar se je z razvojem grafično zahtevnih aplikacij izkazalo, da le-te zasedajo vse večji delež virov CPE. Tako je prišlo do razvoja GPE, ki je prevzela te naloge od CPE.

Razvoj GPE je z letom 1999 začelo podjetje NVIDIA, ki ima še danes vodilno vlogo pri razvoju GPE [8]. Slika 3 prikazuje razvoj hitrosti računanja GFLOPS pri najhitrejših predstavnikih GPE in CPE, Slika 4 pa razvoj hitrosti pretoka podatkov preko pomnilniškega vodila GPE napram CPE [9].



Slika 3: Razvoj hitrosti GPE in primerjava s CPE.



Slika 4: Primerjava razvoja hitrosti prenosa podatkov GPE in CPE.

Na začetku razvoja je bila uporaba GPE omejena na hitro procesiranje slik in geometrijskih informacij. Ker večina grafičnih algoritmov uporablja paralelne podatke, je tudi razvoj GPE

potekal v smeri strojnih in programskih arhitektur, ki so izrabljale paralelizem za doseganje višjih hitrosti računanja [10]. To je tudi razlog za tako razliko med CPE in GPE, saj izris slike na ekran predstavlja veliko podobnih operacij z ogromnimi količinami podatkov – primerno za paralelno obdelavo. Sčasoma je to pripeljalo do uporabe GPE za t.i. splošne namene (GPGPU). To pomeni, da GPE izrabimo za računanja in procese, za katere je tipično zadolžena CPE. Z uporabo več GPE ali kombinacij GPE in CPE lahko dosežemo še dodatno stopnjo paralelnosti [11].

GPE za splošne namene se med drugim uporabljajo tudi za reševanje znanstvenih problemov, kot je napoved vremena, raziskovanje vremena, molekularno modeliranje, ...

Popularnost GPE pri uporabi za reševanje znanstvenih problemov lahko pripišemo nizkim cenam (v primerjavi z CPE), hitrosti pri računanju operacij s plavajočo vejico in visokim hitrostim prenosa podatkov. Napredek pri razvoju GPE je tako pripeljal do tega, da se GPE uporabljajo v visoko zmogljivem računalništvu, prinaša pa tudi nižje stroške, nižjo uporabo električne energije, lažje hlajenje in tudi manj uporabljenega prostora [12].

Arhitektura GPE

Računalniška arhitektura GPE, imenovana Tesla, ki jo je leta 2006 uvedlo podjetje NVIDIA, močno razširi možnost uporabe GPE [9]. Najmanjša enota v GPE, ki je tudi sposobna izvajati ukaze, je procesna enota, več procesnih enot pa se združuje v pretočni multiprocesor (PM). Ti procesorji so združeni v veliko matriko, ki postane visoko učinkovita platforma, tako za izračun grafike, kot za uporabo za splošne namene [13]. Implementacije GPE lahko vsebujejo več tisoč sočasno izvajajočih se niti, zato je glavni cilj transparentno skaliranje in razporejanje niti glede na visoko možnost paralelizma.

Te niti upravlja PM, z uporabo SIMT arhitekture [14]. To je arhitektura, kjer posamezni ukaz podamo več različnim nitim. PM lahko izstavi le en ukaz naenkrat, zato je pomembno, da se niti razporedijo tako, da čim več procesnih enot izvaja isti ukaz.

Niti lahko med izvajanjem dostopajo do podatkov na treh različnih pomnilniških mestih. Vsem nitim je na voljo skupni, globalni pomnilnik. Različni bloki niti ga uporabljajo za deljenje podatkov. Je največji, lociran na kartici, a dostopi do njega imajo največjo zakasnitev. Je tudi edini pomnilnik na GPE, ki prenaša podatke do CPE in obratno [13].

Vsak blok niti ima na voljo tudi deljeni pomnilnik, do njega lahko dostopajo samo niti, ki se izvajajo na istem PM. To je pomnilnik z nizko zakasnitvijo in je namenjen visokozmogljivi komunikaciji in delitvi podatkov med nitmi. Fizično gledano je lociran na samem čipu. Po koncu izvajanja zaporedja ukazov se njegova vsebina prekopira na globalni pomnilnik.

Vsaka nit ima zaseben lokalen pomnilnik. Ta se uporablja samo za uporabo lokalnih spremenljivk, ki jih nit uporablja med izvajanjem ukaza [13].

Primernost problema

Za izvajanje na GPE niso primerni vsi problemi. Sodobne GPE so sestavljene iz velikega števila procesnih enot, zato so primerne za probleme, ki se jih da učinkovito paralelizirati. Značilnosti takih problemov so [15]:

- Visoke računske zahteve. Izrisovanje slike na ekran zahteva milijarde operacij vsako sekundo. Če želimo izkoristiti moč GPE, morajo tudi problemi ponujati visoke računske zahteve.
- Visoka stopnja paralelnosti.
- Pretok podatkov je pomembnejši kot časovni zamik. Grafični algoritmi dajejo prednost pretoku podatkov, saj zamik posamezne operacije iz vidika prikaza na ekranu ni pomemben.

GPE se veliko uporablja v znanstvene namene, eden od takih problemov so tudi simulacije v biokemiji, med katere spada molekulska dinamika, katero simuliramo za potrebe diplomske naloge.

Uporaba GPE za simulacije molekulske dinamike

Simulacije molekulske dinamike so pomembno orodje pri razlagi obnašanja biomolekul. Razvile so se že do te mere, da so simulacije molekul z milijoni atomov postale nekaj običajnega [16]. Take simulacije pa so računsko zelo zahtevne in če za njihovo reševanje uporabljamo tradicionalne infrastrukture, ki temeljijo na CPE, potrebujemo dostop do velikih superračunalnikov. Več kot uspešna alternativa se je izkazala uporaba GPE, ki se poleg nizkih cen (v primerjavi s skupki CPE) ponašajo s hitrim naraščanjem računske moči (Slika 3) in pasovne širine pomnilnika (Slika 4). Posledično lahko vrhunske GPE štejemo med standardno opremo v znanstvenih ustanovah in so na voljo praktično vsem raziskovalcem [17].

Velika računska moč na majhni GPE prinaša druge probleme, kot je zmanjšana fleksibilnost in višja kompleksnost programiranja v primerjavi s CPE. Glavne probleme lahko strnemo v naslednjih točkah [17]:

- Vektorizacija: GPE uporablja SIMT arhitekturo, ki je sposobna visoke paralelnosti. V nasprotju s CPE, ki običajno uporablja do 8 paralelnih niti, GPE tipično procesirajo bloke niti, ki vsebujejo do 128 niti. Vsaka nit v bloku mora izvršiti isti ukaz v istem urinem ciklu, zato je za optimizacijo pomembno, da je koda vektorizirana tako, da se ujema z velikostjo bloka.
- Pomnilniški model: Visoko število aritmetičnih enot na GPE pomeni, da je manj prostora za predpomnilnik in kontrolne enote. Vsa jedra, ki sestavljajo PM, imajo na voljo manjše število registrov, lokalni pomnilnik, ki je omejen na posamezno nit in deljeni pomnilnik, omejen na PM. Večino pomnilnika predstavlja globalni pomnilnik in

je na voljo vsem PM. Kljub temu, da je dostop do globalnega pomnilnika hitrejši kot pri CPE, je še vedno počasen v primerjavi z dostopom do lokalnega predpomnilnika. Ta edinstven pomnilniški model je potrebno upoštevati pri optimizaciji zmogljivosti GPE, potrebno je optimizirati dostope do globalnega pomnilnika, uporabo deljenega pomnilnika za shranjevanje vmesnih rezultatov in uporabo hitrega lokalnega pomnilnika za shranjevanje pogosto uporabljenih podatkov (kot so na primer parametri polja sil pri molekulski dinamiki).

- Komunikacija med GPE in CPE: Pomnilnika GPE in CPE sta na različnih lokacijah, kar pomeni, da je potrebno poskrbeti, da so podatki sinhronizirani. To pa seveda pomeni velik padec zmogljivosti, saj komunikacija poteka preko vodila PCIe in je zelo zamudna.
- Komunikacija med GPE in GPE: Tradicionalna metoda za programiranje vzporednih znanstvenih algoritmov uporablja vmesnik MPI, kjer vsaka nit uporablja svoj pomnilniški prostor. Pri paralelni uporabi GPE je potrebno pri prenosu podatkov med dvema GPE, najprej podatke iz pomnilnika pošiljajoče GPE prenesti v pomnilnik CPE, ki skrbi za nit MPI pošiljajoče GPE. Nato se prenesejo podatki v pomnilnik CPE, ki skrbi za MPI nit prejemne GPE in šele na koncu se podatki prenesejo na pomnilnik prejemne GPE. Pri maksimizaciji zmogljivosti vzporednega izvajanja je tako potrebno zagotoviti čim manjše število prenosov med GPE.

S simuliranjem molekulske dinamike na več GPE so se ukvarjali v več različnih študijah [17] [18] [19] [20]. Ugotovitve kažejo, da komunikacija GPE – CPE – GPE predstavlja ozko grlo. Ker se sistem med simulacijo molekulske dinamike stalno spreminja, je potreba po komunikaciji med GPE stalno prisotna, zato je pohitritev, ko večamo število GPE, razmeroma majhna. Ena od rešitev se kaže v direktnih povezavah med GPE. V primeru, da so GPE v enem vozlišču, je možnost direktne povezave z vodilom PCIe, povezave med različnimi vozlišči pa lahko realiziramo s povezavo Infiniband. Seveda je ob vsem tem potrebna ustrezna programska oprema, primerna za tak pristop.

2.4 Skalabilnost

Skalabilnost pomeni sposobnost prilagoditve sistema ob povečanem številu elementov, ki so v računalništvu procesorji, deloma pa tudi velikost pomnilnika. Sistem mora biti sposoben prilagajanja na povečane delovne naloge in primerno zasnovan, da je mogoča njegova razširitev. Skalabilnost je zaželen atribut vsakega sistema, omrežja ali procesa.

Skalabilnost prav tako pomeni, da je sistem, ne samo sposoben delovati ob povečanem številu elementov, ampak je sposoben izkoristiti dodane vire v svojo korist in s tem izboljšati delovanje. Primer je izboljšano delovanje računalniškega sistema ob dodajanju dodatnega

pomnilnika ali povečana hitrost reševanja računskih problemov ob dodajanju novih procesorjev [21].

Skalabilnost običajno govori o ugodni primerjavi med obstoječim sistemom in večjo verzijo enakega sistema. V primeru vzporednega sistema je smiselno definirati skalabilnost kot pohitritev. Če je *t (n, x)* čas, ki ga za rešitev velikosti problema *x* porabi *n* procesorjev, potem je *pohitritev(n, x)* razmerje med časom za rešitev problema na enem procesorju in časom na *n* procesorjih [22]:

$$pohitritev(n, x) = \frac{t(1,x)}{t(n,x)} \tag{4}$$

Iz pohitritve izpeljemo učinkovitost, kar je pohitritev deljena s številom procesorjev:

$$učinkovitost(n, x) = \frac{pohitritev(n,x)}{n} = \frac{t(1,x)}{t(n,x)*n} \tag{5}$$

Najboljša možna učinkovitost je 1, kar pomeni linearno pohitritev: *pohitritev (n, x)* = *n*.

V realnosti sistem praktično ne more doseči linearne pohitritve, saj obstajajo omejitve. Ena takih je v tem, da ima večina paralelnih algoritmov zaporedno (ali vsaj ne popolnoma paralelno) komponento, kar pripelje do slabše učinkovitosti ob zadostnem številu procesorjev. Druga taka omejitev je v velikosti problema. Če je ta konstanten, ob vse večjem številu procesorjev učinkovitost pada (ko na primer število procesorjev preseže velikost problema) [22].

V kontekstu visoko zmogljivega računalništva skalabilnost delimo na:

(1) Močno skalabilnost – velikost problema ostane enaka, število procesorjev se poveča. Običajno se uporablja pri problemih, ki potrebujejo veliko časa za računanje. V teh primerih želimo poiskati najugodnejšo točko, v kateri se računanje konča razmeroma hitro in hkrati ne porabimo veliko računskih ciklov za koordinacijo paralelizma. V idealnem problemu je pohitritev enaka faktorju dodanih procesorjev.

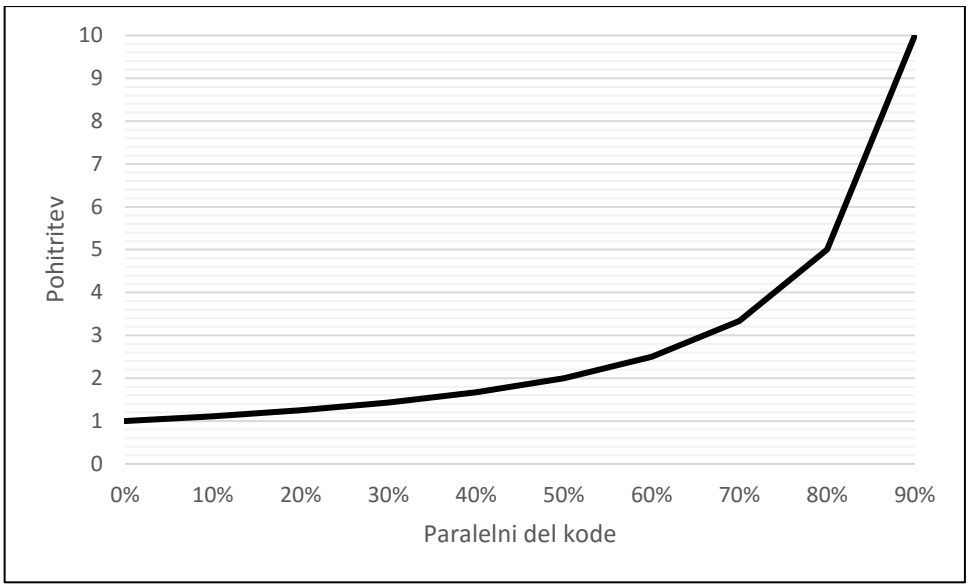
(2) Šibko skalabilnost – hkrati s procesorsko močjo se poveča tudi velikost problema, ki ga rešujemo. Cilj v tem primeru je, da čas za reševanje problema ostaja konstanten [23].

Amdahlov zakon

Amdahlov zakon povezuje skalabilnost z deležem vzporednega računanja. Skalabilnost je v veliki meri odvisna od problema. Če je problem primeren, torej če je možnost paralelizacije visoka, potem je tudi možnost pohitritve pri dodajanju procesorske moči velika. O tem govori Amdahlov zakon, ki pravi, da je pohitritev odvisna od deleža programa, ki se lahko računa vzporedno.

$$pohitritev = \frac{1}{1-P} \tag{6}$$

P predstavlja delež vzporedne kode. Če je *P*=0, potem je pohitritev 1, kar pomeni da pohitritve ni. Če je celotna koda primerna za vzporedno računanje (*P*=1), je v teoriji možna neskončna pohitritev. Slika 5 prikazuje graf pohitritve v odvisnosti od paralelnega dela kode [24].

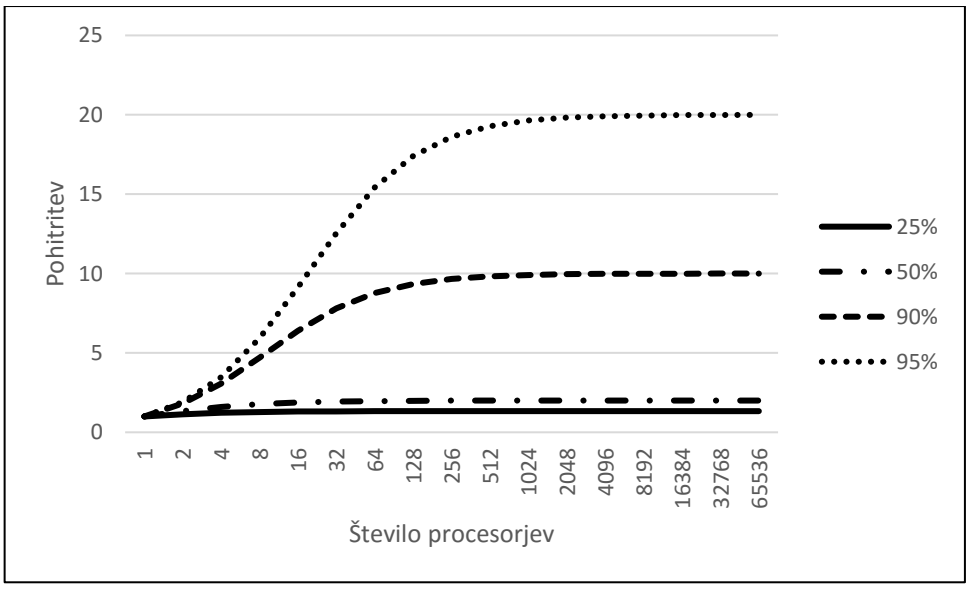


Slika 5: Pohitritev reševanja problema glede na delež programa, ki se lahko procesira vzporedno.

V Amdahlov zakon lahko vključimo tudi število procesorjev, v tem primeru dobimo:

$$pohitritev = \frac{1}{\frac{P}{N}+(1-P)} \tag{7}$$

P je del kode, ki lahko teče vzporedno, *N* pa število procesorjev. Slika 6 prikazuje graf pohitritve v odvisnosti od števila procesorjev.



Slika 6: Maksimalna možna pohitritev glede na število procesorjev pri različnih deležih paralelizacije.

2.5 Infrastruktura na Kemijskem inštitutu

Biomolekularne simulacije predstavljajo ključno računsko podporo strukturni biologiji in molekularni medicini. Ker imamo opravka z velikim številom ogromnih molekularnih sistemov, tovrstnih simulacij ni možno izvajati na delovni postaji, ampak je potrebna velika računska moč, ki jo zagotavlja skupek računalnikov ali grafični procesorji. Zaradi velikega oddajanja toplotne energije moramo strojno opremo namestiti v primerno hlajen računalniški prostor.

Prostor

Prostor (Slika 7) za HPC se nahaja na Kemijskem inštitutu v Ljubljani. Opremljen je s 175 kW hladilne moči v obliki tekoče hladne vode in dovolj energije za napajanje predvidenih 16 omar. Strojne inštalacije omogočajo modularno nadgradnjo opreme brez večjih izpadov delovanja.

Slika 7: Računalniški prostor na Kemijskem inštitutu.

Strežniške omare

V sestav so trenutno vključene tri strežniške omare APC Netshelter SX 42U (AR3300), opremljene s pametnimi APC RACKS PDU 2G električnimi razdelilci, ki nudijo vso potrebno zaščito in omogočajo spremljanje vseh parametrov porabe in delovanja priključene opreme.

Hlajenje

Za hlajenje skrbita dva APC IN ROW RC Chilled Water pametna izmenjevalca, nominalne hladilne moči 18,20 kW. Dodatno skrbi za hlajenje zaprta vroča cona, ki preprečuje mešanje hladnega in vročega zraka.

Strojna oprema

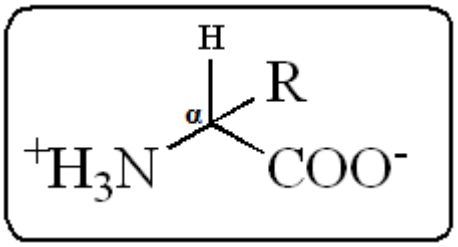
Strojna oprema ponuja vse skupaj 1280 CPE jeder, 2,6 TB pomnilnika in za 166 TB prostora na trdih diskih.

Poleg tega pa še osem GPE enot Nvidia GeForce GTX 780, vsak od njih ima 2304 računskih jeder in 3072 MB pomnilnika.

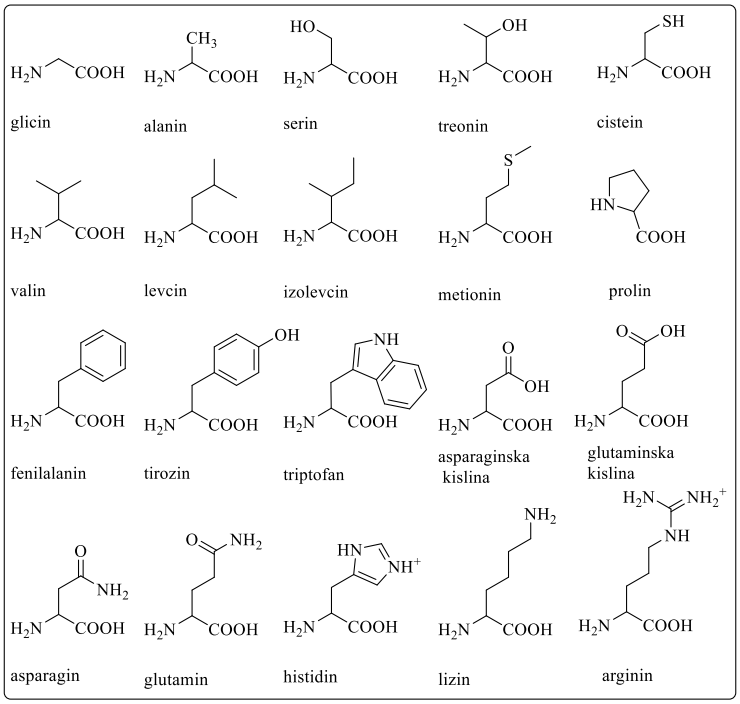
3 Biološke molekule

3.1 Proteini

Proteini so linearni polimeri, zgrajeni iz monomerih enot, ki se imenujejo aminokisline. Poznamo 20 proteinogenih aminokislin. Zgradba aminokisline je predstavljena na Sliki 8, kjer Cα predstavlja α ogljik, NH_3^+ aminsko skupino, COO^- karboksilno, R pa stransko verigo [25] [26]. Stranske verige aminokislin so si med seboj različne in določajo značilne biološke in kemijske lastnosti vsake aminokisline [26]. V fizioloških pogojih so lahko hidrofobne, polarne, pozitivno nabite in negativno nabite. Predstavljene so na Sliki 9.



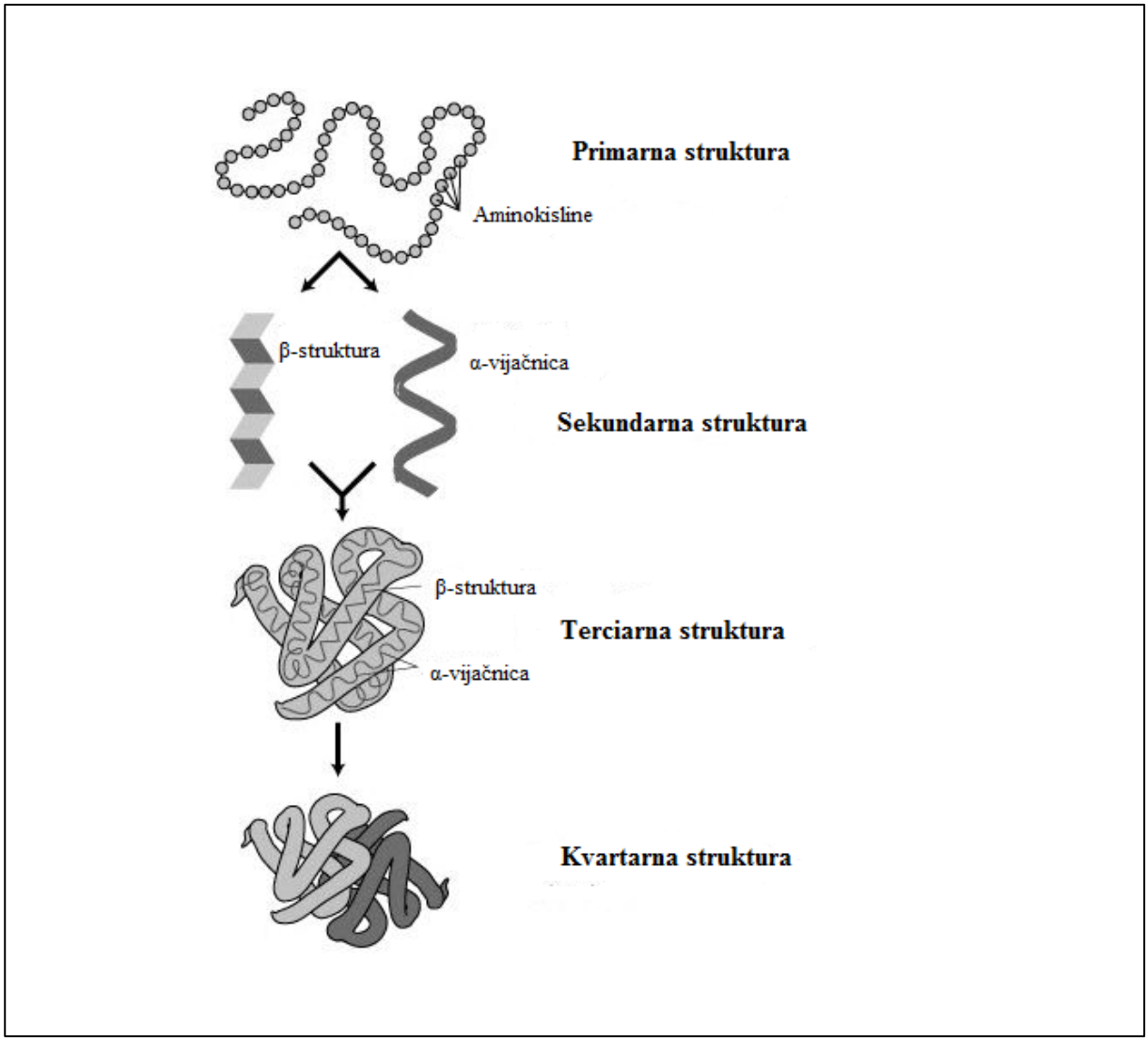
Slika 8: Zgradba aminokisline.



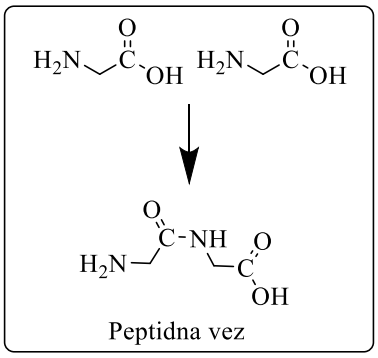
Slika 9: Aminokisline.

Zgradbo proteinov lahko opišemo na štirih ravneh (Slika 10). Poleg primarne strukture tudi na nivoju sekundarne, terciarne in pri nekaterih kvartarne strukture [26]. Primarno strukturo določa zaporedje aminokislin, ki so med seboj povezane s peptidno vezjo (Slika 11) [27] in je osnova

za naslednje tri ravni. Sekundarna struktura je posledica tvorbe vodikovih vezi med bližnjimi aminokislinami. Najpogosteje se povežejo v α-vijačnico ali β-strukturo. Ko se elementi sekundarne strukture povežejo med seboj in zvijejo, dobimo terciarno strukturo proteina. Veliko proteinov ima še kvartarno strukturo, za katero je značilno povezovanje dveh ali več polipeptidnih verig v funkcionalen protein z več podenotami [25] [26].



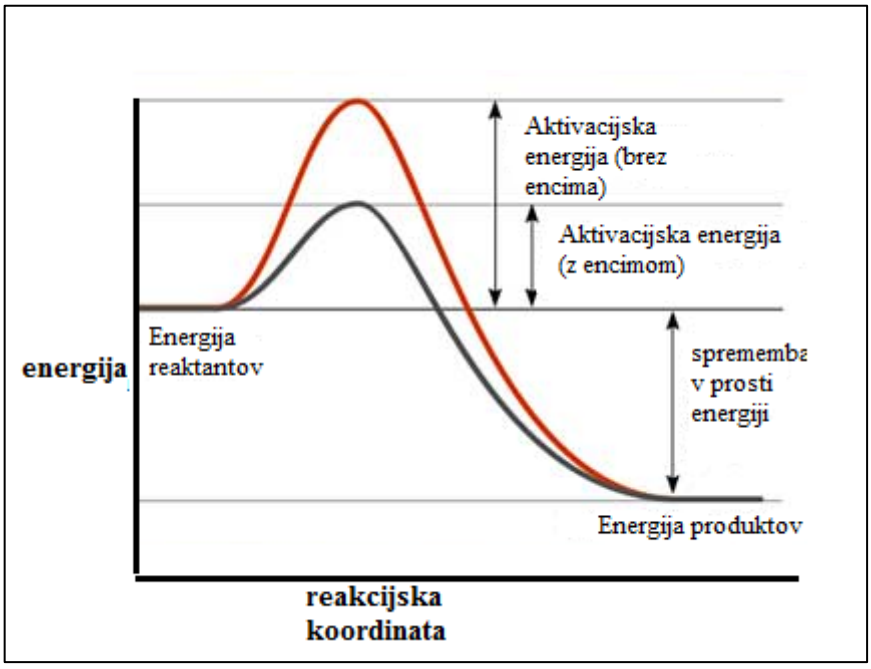
Slika 10: Primarna, sekundarna, terciarna, kvartarna struktura proteina [15].



Slika 11: Peptidna vez.

3.2 Encimi

Encimi so proteini, ki delujejo kot biološki katalizatorji in pospešijo hitrost kemijske reakcije, tako da znižajo aktivacijsko energijo (Slika 12). Potrebni so za nemoteno delovanje celic, saj brez njih reakcije v celici ne bi mogle zadovoljivo potekati. Nekateri encimi imajo v svoji sestavi poleg proteinskega dela še drugo komponento – kofaktor. Kofaktor je lahko organska spojina ali kovinski ion.



Slika 12: Znižanje aktivacijske energije s pomočjo encima.

3.3 Encim monoamin-oksidaza

Monoamin-oksidaza (MAO) je encim, ki sodeluje v presnovi živčnih prenašalcev serotonina, noradrenalina in dopamina [28]. S tem uravnava koncentracije živčnih prenašalcev v

perifernem in centralnem živčnem sistemu ter posledično vpliva na gibanje, spanje, mišljenje, počutje, krvni pritisk in srčno frekvenco [29]. MAO encimi se poleg centralnega živčnega sistema nahajajo tudi drugje v telesu (v jetrih, gastrointestinalnem traktu, trombocitih, …) [28] [29]. Za svoje delovanje MAO potrebuje kofaktor - flavin-adenin dinukleotid (FAD).

Obstajata dve izoobliki MAO encimov - MAO A in MAO B. Razlikujeta se po aminokislinski sestavi, razporeditvi v telesu, imunoloških lastnostih ter specifičnosti do substratov [28].

Vloga monoamin-oksidaz v medicini

Inhibitorji MAO se uporabljajo za zdravljenje depresije in Parkinsonove bolezni. Depresija je motnja razpoloženja, ki se kaže s simptomi, kot so občutki krivde, žalost, apatija, sprememba spanja in apetita, bolečina. Obstaja več hipotez o nastanku depresije. Najstarejša hipoteza pravi, da je za nastanek depresije odgovorno pomanjkanje živčnih prenašalcev noradrenalina, serotonina in v manjši meri tudi dopamina v centralnem živčnem sistemu [30]. Parkinsonova bolezen je degenerativna bolezen možganov, pri kateri propadejo nevroni, ki proizvajajo dopamin. Bolezen se kaže predvsem z motnjami gibanja, pa tudi z motnjami spanja, spomina, razpoloženja, spolnih funkcij [31].

Z inhibicijo encimov MAO lahko vplivamo na koncentracije živčnih prenašalcev, zato inhibitorje MAO encimov uporabljamo za zdravljenje depresije in Parkinsonove bolezni. MAO A presnavlja predvsem serotonin in je glavna tarča za zdravljenje depresije, medtem ko MAO B presnavlja predvsem dopamin ter feniletilamin (PEA) in je zato glavna tarča za zdravljenje Parkinsonove bolezni. Primera zdravilnih učinkovin, ki delujeta selektivno, sta moklobemid kot inhibitor MAO A in selegilin kot inhibitor MAO B [28].

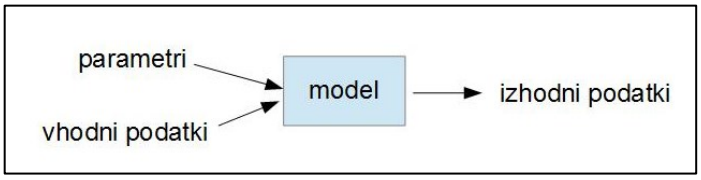
Feniletilamin (PEA)

Kot smo že omenili, je eden od substratov za MAO B feniletilamin, ki se tudi nahaja v našem telesu. Feniletilamin je živčni modulator in živčni prenašalec v centralnem živčnem sistemu. V telesu se sintetizira iz aminokisline fenilalanin [32].

4 Molekulske simulacije

4.1 Računalniške simulacije

Računalniške simulacije uporabljamo kot orodje za teoretične raziskave na različnih znanstvenih področjih. Kot navidezni poskusi so lahko komplementarne ali celo nadomestijo dejanske eksperimente. Pravi eksperimenti namreč niso vedno izvedljivi, bodisi zaradi fizikalnih, tehničnih ali finančnih omejitev. Računalniške simulacije teh omejitev povečini nimajo, zato jih uporabljamo v takih primerih in tako razjasnimo delovanje realnega sistema.



Slika 13: Ponazoritev računalniške simulacije.

Zakonitosti simuliranega sistema vključimo v model. S parametri mu spreminjamo lastnosti, tako da se čim bolje ujema z realnim sistemom. Za uspešno izvedbo podamo tudi začetno stanje sistema, ki ga simuliramo [33]. Ponazoritev računalniške simulacije je prikazana na Sliki 13.

4.2 Simulacija molekulske dinamike

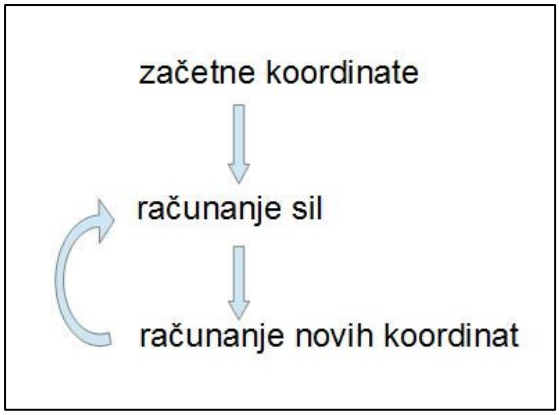
Simulacije molekulske dinamike (MD) omogočajo teoretske raziskave na mnogih področjih znanosti, med drugim v molekularni kemiji. MD omogoča raziskave bioloških in kemijskih sistemov na nivoju atomov v časovnih intervalih od femto (10^{-15}) do mili (10^{-3}) sekund. Dopolnjuje eksperimente in hkrati ponuja možnost slediti procesom, ki jih je težko zaznati z eksperimenti [34]. V splošnem simulacija MD pomeni reševanje problema *n* delcev, zato so zanjo razvite metode prenosljive tudi na druge vrste simulacij [35]. Ker pa ta problem v splošnem ni analitično rešljiv, ga rešujemo numerično z reševanjem sistema gibalnih enačb za vsak posamezen atom [33].

Računanje simulacije MD: Reševanje diferencialnih gibalnih enačb 2. reda:

$$m_i \frac{d\vec{v}_i}{dt} = \sum_j \overrightarrow{f_{i,j}} \tag{8}$$

$$\frac{d\vec{r}_i}{dt} = \overrightarrow{v_i} \tag{9}$$

Kjer je m_i masa atoma *i*, $\vec{v}_i$ hitrost, $\overrightarrow{f_{i,j}}$ sila atoma *j* na atom *i*, $\vec{r}_i$ pa koordinate atoma *i*.



Slika 14: Simulacija MD - glavna zanka.

Glavna zanka računanja MD je prikazana na Sliki 14. Vsak korak je sestavljen iz dveh delov, in sicer iz računanja sil in računanja novih koordinat. Te nato uporabimo pri računanju sil v naslednjem koraku [33].

4.3 Polje sil

Polje sil je skupek atomskih tipov in njihovih veznih in neveznih parametrov, ki omogočajo simuliranje kateregakoli sistema. V praksi je polje sil potrebno skrbno izbrati glede na raziskovani problem. Parametre se običajno pridobi s poskusi ali z zahtevnejšimi kvantno-kemijskimi izračuni [29].

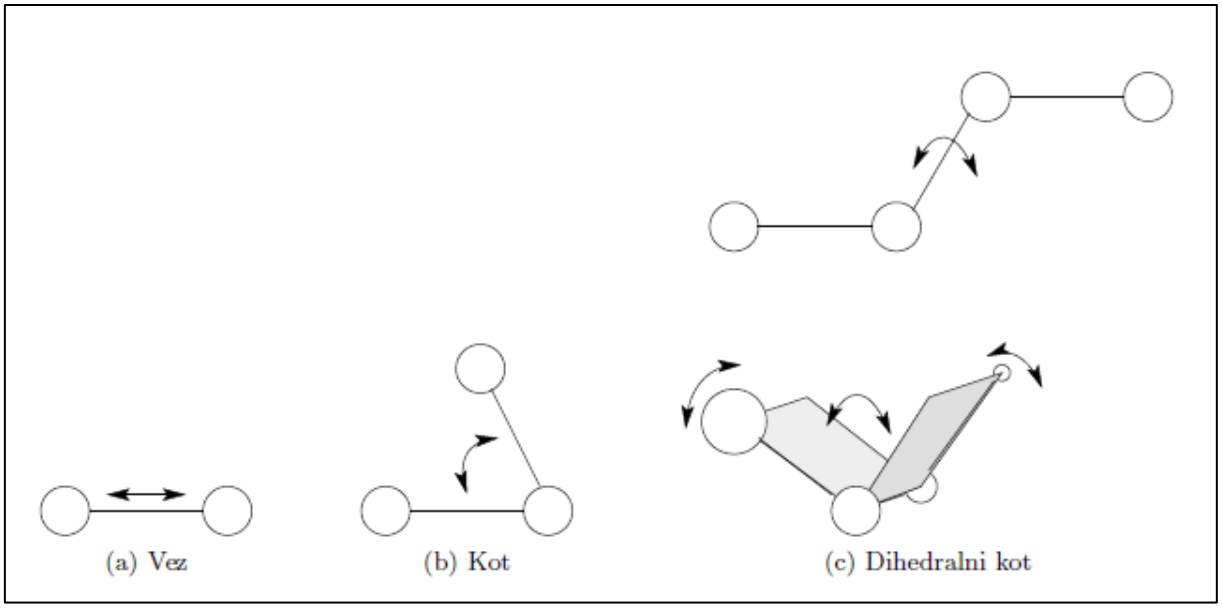
Med parametre za vezne interakcije prištevamo parametre za raztezanje vezi, zvijanje vezi, rotacijo okoli vezi in za nihanja izven ravnine. Parametri za nevezne interakcije pa so disperzijske sile (Lennard-Jonesov potencial) in Coulombove elektrostatične interakcije.

Polje sil opišemo z veččlensko enačbo [36]:

$$V(\vec{r}^N) = \sum_i \frac{k_i}{2}(l_i - l_{i,0})^2 + \sum_{koti} \frac{k_i}{2}(\theta_i - \theta_{i,0})^2 + \sum_{torzije} \frac{V_n}{2}(1 + cos(n\omega - \gamma)) + \sum_{i=1}^{N}\sum_{j=i+1}^{N}\left(4\epsilon_{ij}\left[\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}}\right)^{6}\right] + \frac{q_i q_j}{4\pi\epsilon_0 r_{ij}}\right) \quad (10)$$

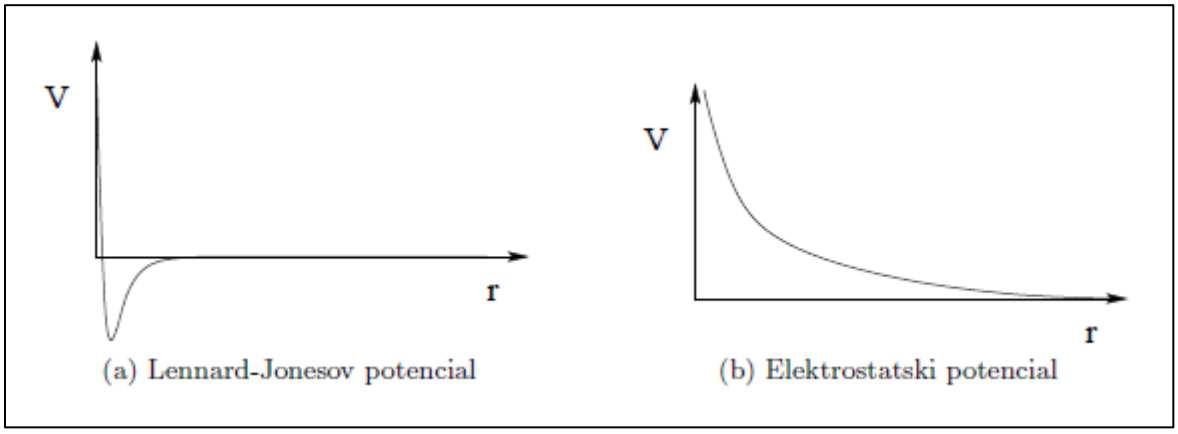
$V(\vec{r}^N)$ prikazuje celotni potencial v odvisnosti vektorja koordinat vseh atomov. Prvi člen prikazuje potencial raztezanja vezi v odvisnosti njune razdalje. Modeliran je s harmonskim

potencialom, kar si lahko predstavljamo kot kroglici povezani z vzmetjo (Slika 15a). Drugi člen prikazuje prispevek potenciala kotov vseh trojk vezanih atomov, prav tako modeliran s harmonskim potencialom (Slika 15b). Tretji člen opisuje potencial rotacije glede na torzijske (dihedralne) kote med štirimi vezmi. Dihedralni kot je kot med ravninama, določenima s prvimi in zadnjimi tremi atomi (Slika 15c). Ti trije členi opisujejo vezne interakcije med atomi.



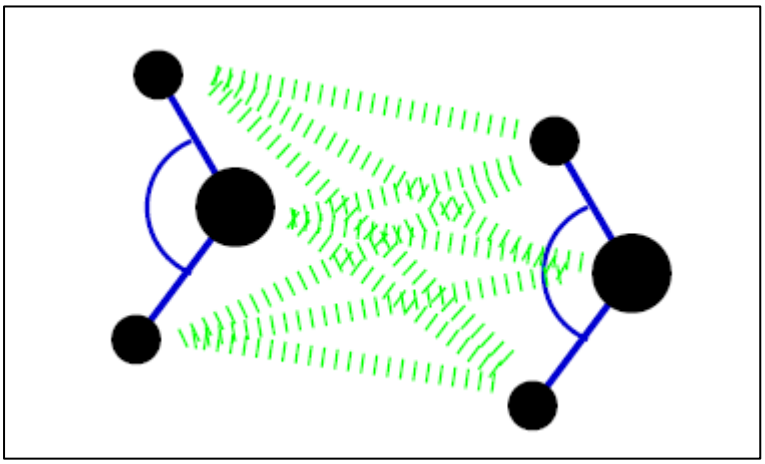
Slika 15: Vezne interakcije med atomi.

Zadnji člen opisuje prispevek van der Waalsovih in elektrostatskih interakcij med atomi k skupnemu potencialu. Van der Waalsove interakcije so opisane z Lennard-Jonesovim potencialom, ki opisuje nevezne interakcije med atomi glede na njihovo medsebojno razdaljo (Slika 16a). Elektrostatske interakcije pa nastanejo zaradi privlaka nasprotno nabitih ionov, ali obratno zaradi odboja enako nabitih ionov (Slika 16b).



Slika 16: Potencial neveznih interakcij med atomi.

Slika 17 prikazuje vezne in nevezne interakcije sistema z dvema molekulama vode, torej šestih atomov. Za pare atomov, ki jih obravnavamo z veznimi interakcijami, neveznih interakcij ne računamo, saj je njihov vpliv že zajet v izračunih veznih interakcij.



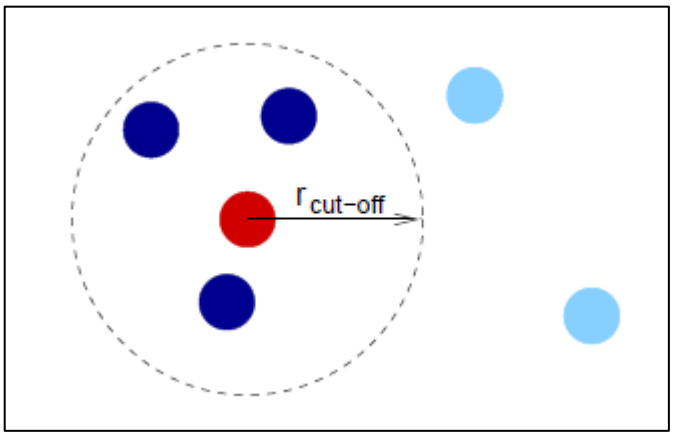
Slika 17: Vezne in nevezne interakcije med dvema molekulama vode [22].

Že na tako majhnem sistemu vidimo, da je neveznih interakcij mnogo več kot veznih. V sistemu z N atomi je veznih interakcij O(N), saj ima vsak atom navadno do tri vezi. Neveznih interakcij pa je O(N^2), saj ima vsak atom nevezne interakcije z vsemi atomi v sistemu in je v principu potrebno izračunati nevezne interakcije za vsak par atomov v sistemu [36]. Izračun neveznih interakcij je tako med večjimi omejitvami računalniške simulacije MD. Omejena je tako dolžina simulacije, kot tudi velikost simuliranega sistema.

Za pohitritev izračuna in povečanje sistema je bilo razvitih več metod. Najbolj razširjena je uporaba radija cut-off.

Cut-off

Metoda cut-off spada med najosnovnejše metode zmanjšanja računske zahtevnosti neveznih interakcij [36]. Uporaba le-te upošteva to, da se vrednosti neveznih interakcij med dvema atomoma manjšajo z večanjem razdalje med njima. Na tak način se obnašata tako Lennard-Jonesov potencial kot tudi elektrostatski potencial. Oba se z večanjem razdalje približata ničli, zato lahko določimo, da je razlika zanemarljiva nad neko razdaljo. To razdaljo imenujemo cut-off, prikazuje jo Slika 18. Nevezne interakcije računamo samo do atomov, ki so znotraj radija $r_{cut-off}$, za ostale pa predpostavimo, da je vrednost potenciala enaka 0 in jih ne računamo.



Slika 18: Razdalja cut-off [22].

Primernost te metode se pokaže zlasti pri večjih sistemih. Pri sistemu z N atomi in atomsko gostoto ρ (število atomov na prostorsko enoto) ni potrebno računati N^2 interakcij ampak le n_{calc} interakcij, kjer je n_{calc} [33]:

$$n_{calc} = N \times \frac{4}{3} \times \pi \times r_{cut-off}^3 \times \rho \tag{11}$$

4.4 Računanje energije med molekulami vode

Namen izračuna skupne energije sistema n molekul vode je prikazati obsežnost in časovno potratnost že samo v eni časovni točki, enem koraku molekulske dinamike. Sistem se z molekulsko dinamiko neprestano spreminja, pri čemer se spreminjajo koordinate atomov, kar vpliva na različne energije interakcij v različnih časovnih točkah. Poleg tega je voda osnova za molekulsko dinamiko, saj vse simulacije makromolekul potekajo v vodi.

Najprej je potrebno izračunati razdalje med vsemi kombinacijami atomov iz različnih molekul vode (Slika 17). Podatek o razdalji je potreben za izračun interakcijske energije med posameznimi kombinacijami teh atomov. Posamezne izračunane interakcijske energije

seštejemo in jim dodamo Lennard-Jonesov potencial, ki pripada interakcijam med kisikovimi atomi različnih molekul vode. S tem dobimo skupno energijo sistema *n* molekul vode v določenem trenutku.

Razdaljo med atomoma različnih molekul *i* in *j* izračunamo po formuli (Enačba 12) za razdaljo v tridimenzionalnem koordinatnem sistemu:

$$razdalja = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2} \tag{12}$$

Interakcijska energija med posameznimi pari atomov se izračuna po formuli v Enačbi 13. Za naboj atoma vodika smo uporabili konstanto 0,41, za naboj kisika pa -0,82.

$$E_{ij} = \frac{naboj_i \times naboj_j \times e_0}{4 \times \pi \times \varepsilon_0 \times razdalja_{ij}} \tag{13}$$

V Enačbi 13 sta spremenljivki naboj in razdalja med atomoma, ostale vrednosti so konstantne in njihova skupna vrednost je izračunana v Enačbi 14. Pri tem je e_0 osnovni naboj atoma, ε_0 pa dielektrična konstanta.

$$\frac{e_0}{4 \times \pi \times \varepsilon_0} = 332{,}04 \tag{14}$$

Lennard-Jonesov potencial se zaradi poenostavitve izračuna le med kisikoma in sicer po formuli v Enačbi 15:

$$E_{LJ} = 4\epsilon_{ij}\left[\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}}\right)^{6}\right] \tag{15}$$

Skupna interakcijska energija med molekulama je seštevek posameznih interakcijskih energij vseh kombinacij parov atomov z dodanim Lennard-Jonesovim potencialom:

$$E_{total} = \sum E_{ij} + E_{LJ} \tag{16}$$

Za poenostavitev izračuna lahko v tem koraku uporabimo radij cut-off, ki nam izračun interakcijskih energij med dvema atomoma omeji na razdaljo 10 Å. Angstrem je enota, ki predstavlja 10^{-10} metra. Zadnji korak je seštevek interakcijskih energij vseh kombinacij atomov.

4.5 Amber

Amber je skupno ime za skupek programov, ki omogočajo simulacije molekulske dinamike, še posebej simulacijo bioloških molekul. Poleg tega izraz Amber prav tako predstavlja polja sil, ki jih implementirajo programi v paketu Amber. Programi in polja sil sta ločen izdelek. Polja sil Amber so v javni rabi in jih implementirajo tudi mnogi drugi programi za molekulske simulacije, medtem ko je programski paket plačljiv [37] [38]. Za potrebe diplomske naloge bomo uporabljali različico Amber2015, ki sestoji iz dveh delov. Glavnega, plačljivega dela

(Amber14), ki je osnovan okoli simulacijskega programa *pmemd* in iz paketa programov AmberSuite15, ki je na voljo brezplačno.

Pri izvajanju simulacij za potrebe diplomske naloge bomo uporabili simulacijski program *pmemd* in njegovo različico za grafične procesne enote.

Pmemd

Pmemd je simulacijski program in je osrednji del paketa Amber. Idealen je za izvajanje dolgotrajne molekulske dinamike velikih sistemov. Poleg tega je eden prvih programov za izvajanje molekulske dinamike s pomočjo grafično procesnih enot. Pmemd podpira več polj sil, tako proteinov in nukleinskih kislin, kot tudi modelov vod in njihovih organskih topil [37]. Verzija za več procesorjev se imenuje *pmemd.MPI*, verzija za GPE *pmemd.cuda*, verzija za več GPE pa *pmemd.cuda.MPI.*

Pmemd za svoje delovanje potrebuje tri tipe vhodnih datotek [39]:

1. *Prmtop* – datoteka, ki opisuje parametre in topologijo molekul v sistemu
2. *Inpcrp* – datoteka začetnih koordinat sistema
3. *Mdin* – datoteka z nastavitvami za izvajanje molekulske dinamike

V tem diplomskem delu smo se posvetili molekulski simulaciji encima MAO B, ki je odgovoren za razgradnjo dopamina in predstavlja ključno farmakološko tarčo za zdravljenje Parkinsonove bolezni. Omejili smo se na strukturne aspekte, kemijskih korakov pa se nismo dotikali. Strukturni aspekti so pomembni pri načrtovanju inhibitorjev in pri razumevanju točkovnih mutacij, ki vodijo do nevropsihiatričnih obolenj.

5 Izvedba

Pri biomolekularnih simulacijah izjemno težaven del simulacije predstavlja gradnja molekulske topologije, ki mora biti usklajena z uporabljenim poljem sil in molekulsko geometrijo. Zaradi kompleksnosti bioloških makromolekul je ta del še vedno deloma treba opraviti ročno in tudi najboljši programski paketi vedno zahtevajo nekaj časa za iskanje napak in njihovega odpravljanja. Tipično dobimo geometrijo biološke makromolekule iz spletne baze struktur proteinov (Protein Data Bank - PDB), kjer so deponirane eksperimentalne strukture. Strukturo predstavljajo kartezične koordinate vseh atomov, zapisane v tekstovni datoteki. Manjkajo jim vodikovi atomi, saj rentgenska difrakcija s svojo omejeno resolucijo ne omogoča določitve njihovih koordinat. Manjkajoči vodikovi atomi se dodajo z orodjem AmberTools15.

Koordinate manjkajočih atomov dobimo iz eksperimentalnih podatkov za homologne gradnike s predpostavko, da se dolžine vezi in vezni koti med homolognim modelom in našim proteinom ne spreminjajo. Za to smo uporabili programski paket VMD (Visualisation of Molecular Dynamics) in orodje AmberTools15.

Ko smo s strukturo proteina zadovoljni, mu dodamo molekule vode, ki so potrebne za uspešno izvedbo simulacije, saj predstavljajo naravno okolje, v katerem se nahaja protein v biološkem sistemu. Tudi molekule vode dodamo z orodjem AmberTools15. Rezultat tega je datoteka z začetnimi koordinatami atomov.

Sledi določitev polja sil, kjer so ključnega pomena nevezne interakcije, zlasti atomski naboji. Polje sil prav tako določimo z orodjem AmberTools15. Rezultat tega je datoteka z opisi atomskih nabojev in začetnih medsebojnih sil.

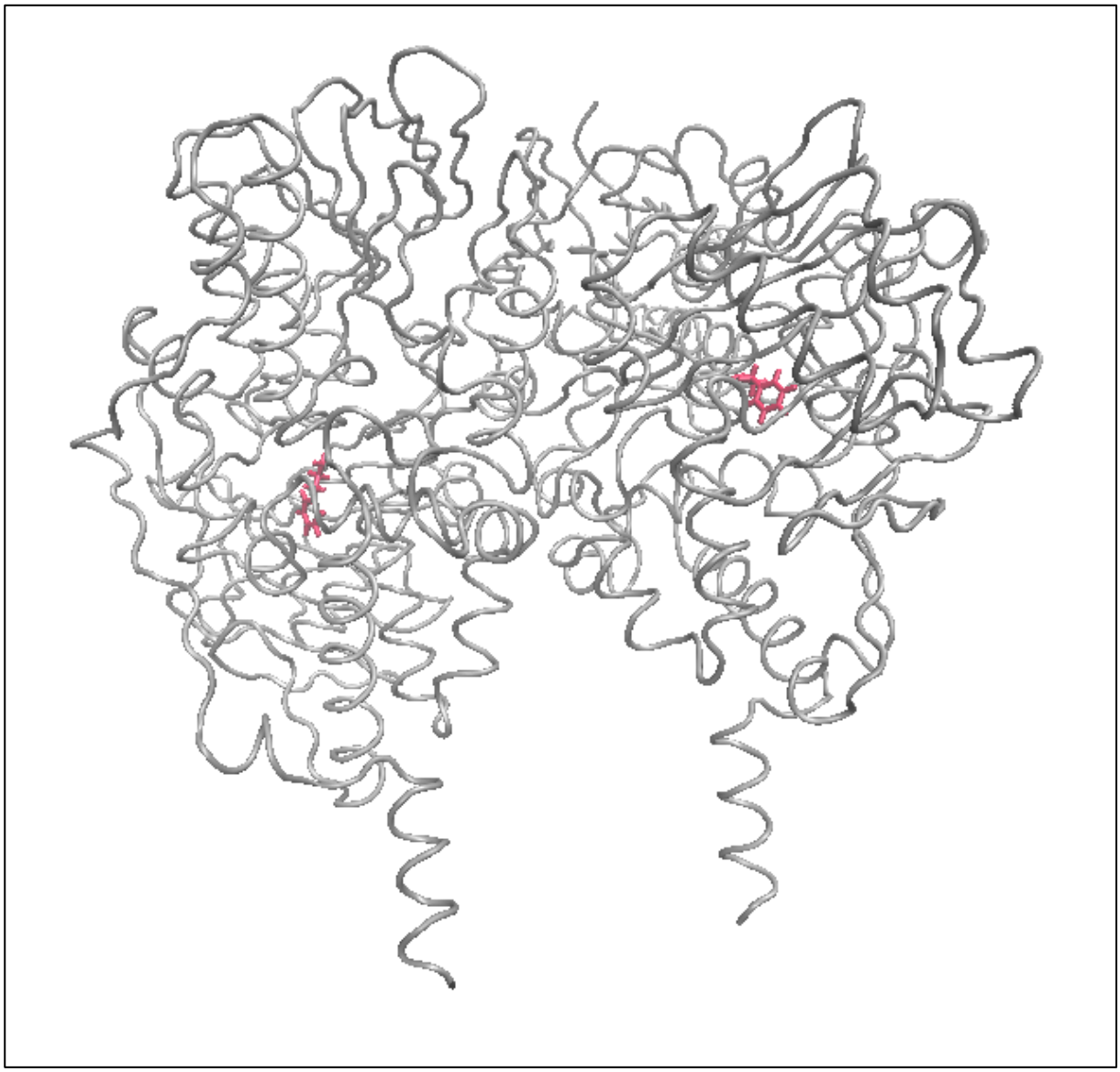
Nato tipično opravimo energijsko minimizacijo, ki ji sledi uravnoteženje in na koncu še produkcijski del molekulske dinamike. Po končani simulaciji sledi analiza in interpretacija rezultatov.

Če povzamemo, za uspešen potek simulacije molekulske dinamike sledimo naslednjim korakom:

1. Pridobitev geometrije (koordinat) proteina iz PDB, dodamo manjkajoče atome in molekule ter določimo polje sil
2. Priprava vhodnih parametrov
3. Izvedba molekulske dinamike
4. Analiza in interpretacija rezultatov simulacije

5.1 Priprava geometrije

Za simulacijo smo uporabili encim MAO B v kompleksu z feniletilaminom. Začetno strukturo smo naložili iz PDB (koda 1S2Q). V tej kristalni strukturi protein obstaja kot homodimer (2 enaki podenoti) s kovalentno vezanim kofaktorjem FAD in vezanim inhibitorjem encima rasagilinom, katerega smo za simulacijo odstranili. Strukturo prikazuje Slika 19.

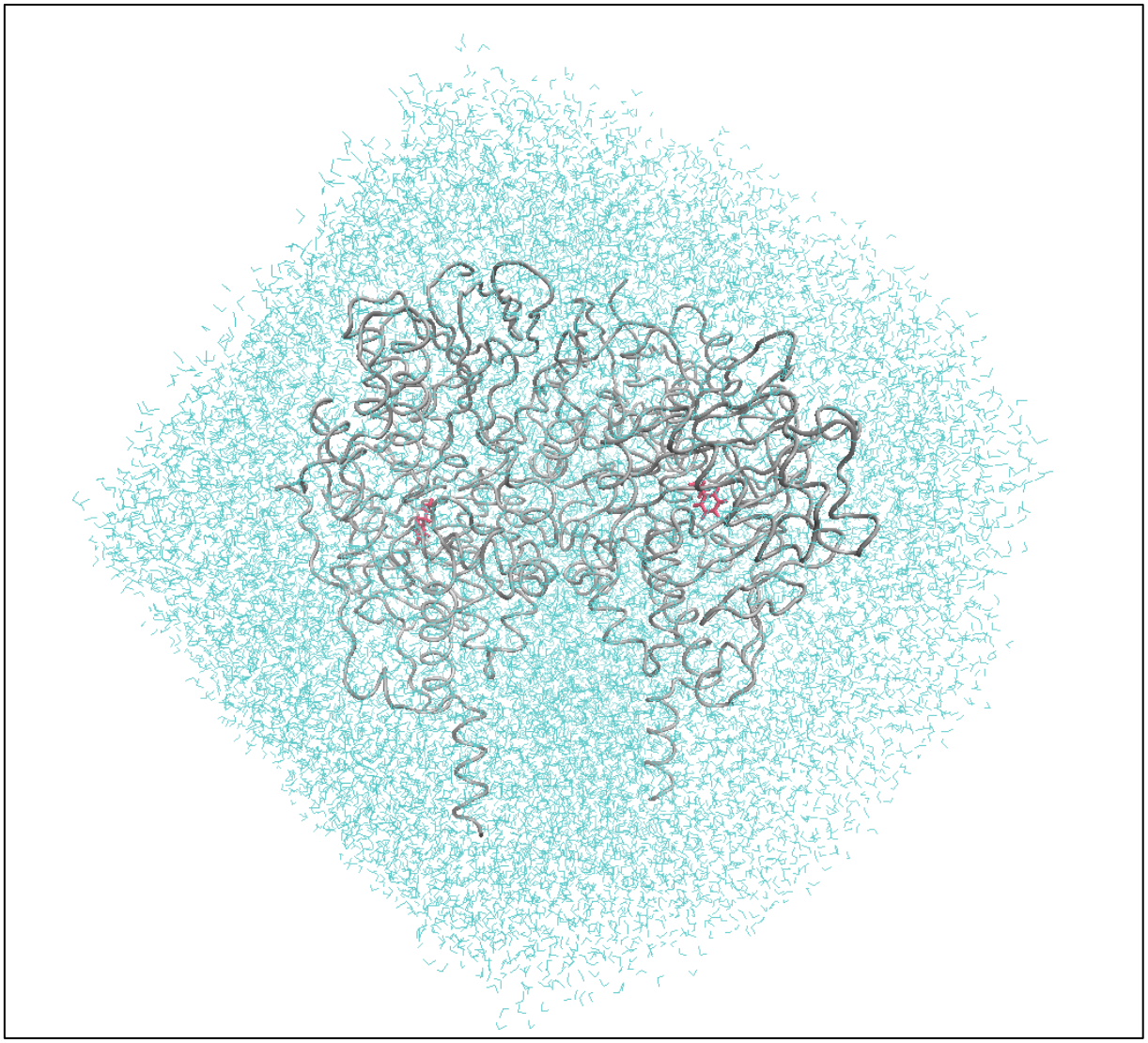


Slika 19: Encim MAO B v kompleksu s feniletilaminom in kofaktorjem FAD (rdeči molekuli).

Protein zalijemo z uravnoteženo strukturo vode, tako da ta zapolni celotno simulacijsko celico, medtem ko je protein v središču. Dodajanje molekul vode je pomembno, saj s tem oslabimo premočne interakcije med nabitimi skupinami. Če molekul vode ne bi dodali, bi se zaradi premočnih interakcij med nabitimi skupinami razdalje med atomi zmanjšale, kar bi imelo za rezultat preveč kompaktno strukturo s premajhnim radijem vrtenja. Dodane vode so nujne tudi zaradi primerjave z eksperimentom, saj proteinski kristali vsebujejo veliko število molekul

vode. Pri numeričnem postopku je ključnega pomena algoritem, ki odstrani molekule vode, ki bi bile v pretesnem kontaktu s proteinom. Za ta postopek smo uporabili orodje AmberTools15. Po dodajanju manjkajočih atomov in molekul vode je sledila pazljiva vizualizacija začetne strukture, da bi preprečili napake, kot so slabi kontakti ali manjkajoče vezi. Za to smo uporabil programski paket VMD.

Rezultat je hidratiran protein, grafični prikaz je na Sliki 20.



Slika 20: Hidratirani protein MAO B. Topilo in protein tvorita oktaedersko simulacijsko celico. Skupno število atomov v simulacijski celici je 75445.

5.2 Priprava vhodnih parametrov

Potek priprave vhodnih parametrov za simulacijo molekulske dinamike sestoji iz treh delov. Prvi del je energijska minimizacija, s katero optimiziramo geometrijo našega sistema. Energijska minimizacija fizikalno pomeni, da odstranimo kinetično energijo in sistem

pripeljemo v minimum. Odstranjene so vse napetosti in prevelika približanja atomov. Praktično energijsko minimizacijo izvedemo tako, da algoritem toliko časa spreminja koordinate atomov, da sile padejo pod predpisano vrednost, kar pomeni, da smo dosegli energijski minimum. Minimizacija je integrirana v paketu Amber, potrebno je pravilno nastaviti vhodno datoteko in parametre. Z vidika procesorske zahtevnosti je enaka produkcijski molekulski dinamiki.

Sledi uravnoteženje sistema, izhajamo iz minimuma in počasi segrevamo sistem, da se v prvi stopnji najprej premešajo vode, čemur sledi druga stopnja, »dihanje« proteina, do njegove naravne lege. S segrevanjem dosežemo, da se poviša kinetična energija. Takrat se atomi začnejo premikati. Uravnoteženje poteka enako kot produkcijska dinamika, potrebno je zato, da začnemo s produkcijsko dinamiko, ko je protein v naravni legi. S tem zagotovimo točnost rezultatov. Tudi uravnoteženje je z vidika procesorske zahtevnosti enako kot produkcijska molekulska dinamika.

Ko končamo, je na vrsti še dejanska produkcijska molekulska dinamika.

Minimizacija

Minimizacija je sestavljena iz štirih ciklov. Prvi traja 1500 korakov. Molekule vode in substrat pustimo, da se prosto gibajo, medtem ko na protein in kofaktor vplivamo s harmonskim potencialom s konstantno silo 30 $\frac{kcal}{mol*Å^2}$. Slika 21 nam kaže vhodne parametre za prvi cikel minimizacije. Pomen pomembnejših nastavitev:

Imin = 1	pomeni, da bo potekala minimizacija
Maxcyc = 1500	maksimalno število korakov minimizacije
Ncyc = 470	prvih 470 korakov se za minimizacijo uporabi metoda najhitrejšega spusta, v ostalih korakih pa metoda gradientnega spusta
Ntmin = 1	če je 1, se metodi minimizacije zamenjata po *ncyc* korakih
Ntb = 1	spremenljivka, ki določa obnašanje sistema na njegovih mejah
Cut = 10	razdalja *cut-off* v angstremih
Ntr = 1	če je 1, nam omogoča, da omejimo določene elemente s harmonskim potencialom

```
Input min1.in
&cntrl
imin=1, maxcyc=1500, ncyc=470, ntmin=1, nsnb=15, ntb=1, cut=10.0, ntr=1
/
Hold the protein constrained
30  // sila harmonskega potenciala
RES 1 1002  // definiramo, na katere dele našega sistema vplivamo s harmonskim p.
END
END
```

Slika 21: Parametri za prvi cikel minimizacije.

Drugi cikel minimizacije traja 2500 korakov, razlika pa je v tem, da s harmonskim potencialom s konstantno silo 100 $\frac{kcal}{mol*Å^2}$ vplivamo samo na glavno verigo proteina. Parametre za drugi cikel prikazuje Slika 22.

Tretji in četrti cikel minimizacije sta enaka kot drugi, s tem da harmonski potencial zmanjšamo na 10 $\frac{kcal}{mol*Å^2}$. Tretji cikel poženemo za 1500, četrti pa za 5000 korakov.

```
Input min2.in
&cntrl
imin=1, maxcyc=2500, ncyc=470, ntmin=1, nsnb=15, ntb=1, cut=10.0, ntr=1
/
Hold the protein BB constrained
100      // sila harmonskega potenciala
FIND  // določimo, katere atome vključimo v delovanje harmonskega p.
C * * *
O * * *
N * * *
CA * * *
SEARCH       // določimo v katerem delu sistema iščemo zgornje atome
RES 1 1002    //
END
```

Slika 22: Parametri za drugi cikel minimizacije.

Uravnoteženje

Minimizaciji sledita dva kratka cikla uravnoteženja sistema. V prvem ciklu simuliramo 50 pikosekund molekulske dinamike sistema. Med dinamiko simuliramo ogrevanje sistema iz 0 na 300 K. Čas 50 pikosekund je iz kemijskega vidika potreben zato, da se molekule vode, ki obkrožajo naš protein, dobro premešajo. 50 pikosekund namreč ustreza 10-kratnemu korelacijskemu času rotacijskega časa vode. Iz računalniškega vidika je uravnoteženje sistema enako glavni molekulski dinamiki. Med tem na protein in kofaktor vplivamo s harmonskim potencialom 100 $\frac{kcal}{mol*Å^2}$. Slika 23 prikazuje vhodne parametre za prvi cikel.

```
Input eq1.in
&cntrl
imin=0, irest=0, ntx=1, ntb=1, cut=10.0, ntr=1, ntc=2, ntf=2,
tempi=0.0, temp0=300.0, ntt=3, gamma_ln=1.0,
nstlim=50000, dt=0.001, ntpr=100, ntwx=100, ntwr=1000
/
Hold the protein constrained
100
RES 1 1002
END
```

Slika 23: Vhodni parametri za prvi cikel uravnoteženja.

Pomen dodatnih nastavitev pri uravnoteženju:

Ntx=1	če je 1, se preberejo samo koordinate
Tempi=0.0	začetna temperatura sistema, 0 K
Temp0=300.0	končna temperatura sistema, 300 K
Ntt=3	uporaba Langevinovega termostata za kontrolo temperature

V drugem ciklu ponovno simuliramo 50 pikosekund molekulske dinamike, s tem da simulacija poteka na konstantni temperaturi 300 K. S tem dosežemo, da je sistem pred začetkom produkcijske dinamike v čim bolj naravnem stanju. Slika 24 kaže potrebne parametre za drugi cikel uravnoteženja.

```
Input eq2.in
&cntrl
imin=0, irest=1, ntx=5, iwrap=1, ntb=2, ntp=1, cut=10.0, ntc=2, ntf=2, ig=-1,
tempi=300.0, temp0=300.0, ntt=3, gamma_ln=1.0,
nstlim=50000, dt=0.001, ntpr=100, ntwx=100, ntwr=1000
/
```

Slika 24: Vhodni parametri za drugi cikel uravnoteženja.

Produkcijska molekulska dinamika

Uravnoteženju sledi glavni del naše molekulske dinamike. To je simulacija 5 nanosekund našega sistema, s korakom 1 femtosekunde, kar pomeni 5 milijonov korakov. Slika 25 prikazuje vhodne parametre.

```
Input md1.in
&cntrl
imin=0, irest=1, ntx=5, iwrap=1,
ntb=2, ntp=1, cut=10.0, ntc=2, ntf=2, ig=-1,
tempi=300.0, temp0=300.0, ntt=3, gamma_ln=1.0,
nstlim=5000000, dt=0.001, ntpr=1000, ntwx=1000, ntwr=10000
/
```

Slika 25: Vhodni parametri za produkcijsko molekulsko dinamiko.

5.3 Izvedba molekulske dinamike

Ko pripravimo vse vhodne datoteke, lahko poženemo simulacijo. Začnemo z minimizacijo, ukaze izvajamo v ukazni vrstici. Ukazi za minimizacijo so prikazani na Sliki 26.

```
pmemd -O -i min1.in -o MAO_PEA_min1.out -p MAO_PEA.prmtop -r MAO_PEA_min1.rst -c MAO_PEA.inpcrd -ref MAO_PEA.inpcrd

pmemd -O -i min2.in -o MAO_PEA_min2.out -p MAO_PEA.prmtop -r MAO_PEA_min2.rst -c MAO_PEA_min1.rst -ref MAO_PEA_min1.rst

pmemd -O -i min3.in -o MAO_PEA_min3.out -p MAO_PEA.prmtop -r MAO_PEA_min3.rst -c MAO_PEA_min2.rst -ref MAO_PEA_min2.rst

pmemd -O -i min4.in -o MAO_PEA_min4.out -p MAO_PEA.prmtop -r MAO_PEA_min4.rst -c MAO_PEA_min3.rst -ref MAO_PEA_min3.rst
```

Slika 26: Ukazi za minimizacijo.

Pomen nastavitev:

-O	v primeru, da izhodna datoteka že obstaja, se jo prepiše
-i min1.in	vhodna datoteka minimizacije
-o MAO_PEA_min1.out	izhodna datoteka minimizacije

-p MAO_PEA.prmtop	vhodna datoteka s parametri in topologijami
-r MAO_PEA_min1.rst	izhodna datoteka za ponovni zagon
-c MAO_PEA_inpcrd	datoteka s koordinatami
-ref MAO_PEA.inpcrd	datoteka z referenčnimi koordinatami, ki jih uporabimo pri simulaciji z omejitvami

Lahko vidimo, da vsak naslednji korak zamenja datoteko s koordinatami z izhodno datoteko za ponovni zagon koraka pred seboj. Enako bo pri vhodnih datotekah uravnoteženja in dinamike. Ukaza za uravnoteženje sta prikazana na Sliki 27, ukaz za dinamiko na Sliki 28.

```
pmemd -O -i eq1.in -o MAO_PEA_eq1.out -p MAO_PEA.prmtop -r MAO_PEA_eq1.rst -c MAO_PEA_min4.rst -ref MAO_PEA_min4.rst -x MAO_PEA_eq1.xcrd

pmemd -O -i eq2.in -o MAO_PEA_eq2.out -p MAO_PEA.prmtop -r MAO_PEA_eq2.rst -c MAO_PEA_eq1.rst -ref MAO_PEA_eq1.rst -x MAO_PEA_eq2.xcrd
```

Slika 27: Ukaza za uravnoteženje.

```
pmemd -O -i md1.in -o MAO_PEA_md1.out -p MAO_PEA.prmtop -r MAO_PEA_md1.rst -c MAO_PEA_eq2.rst -ref MAO_PEA_eq2.rst -x MAO_PEA_md1.xcrd
```

Slika 28: Ukaz za produkcijsko molekulsko dinamiko.

Dodatna nastavitev je *–x*, kar pomeni izhodno datoteko s trajektorijo simuliranega sistema.

5.4 Analiza in interpretacija rezultatov simulacije

Med dinamiko program vsakih nekaj korakov zapisuje koordinate in hitrosti, kar imenujemo trajektorija. Trajektorije ne zapisujemo pri vsakem koraku, saj bi bile koordinate preveč korelirane. V 1 femtosekundi, ki predstavlja v našem primeru korak za numerično integracijo enačb gibanja, se zgodi zelo malo. Najhitrejši dogodek pri dinamiki hidratiranega proteina je natezno nihanje NH vezi, kjer je za en nihaj potrebno čakati približno 10 femtosekund. Dodaten problem bi bili daljši časi računanja in mnogo večja poraba pomnilnika. Na koncu dinamike trajektorija predstavlja ogromno datoteko, ki je uporabna za analizo in vizualizacijo rezultatov. Prepoznamo jo po končnici *.xcrd*. Ostale komponente (energijske komponente, tlak, temperatura,…) program zapisuje v izhodno datoteko (prepoznamo jo po končnici *.out*), kar omogoča kontrolo in analizo**.** Iz trajektorije in izhodne datoteke izluščimo rezultate za kemijsko analizo, ki so predstavljeni v poglavju 6.2. Frekvenco teh zapisov določimo s parametri NTPR

(izhodni tok), NTWX (trajektorija) in NTWR (datoteka za ponovni zagon, če pride do napake in za vhodne koordinate naslednjega koraka simulacije).

6 Rezultati in diskusija

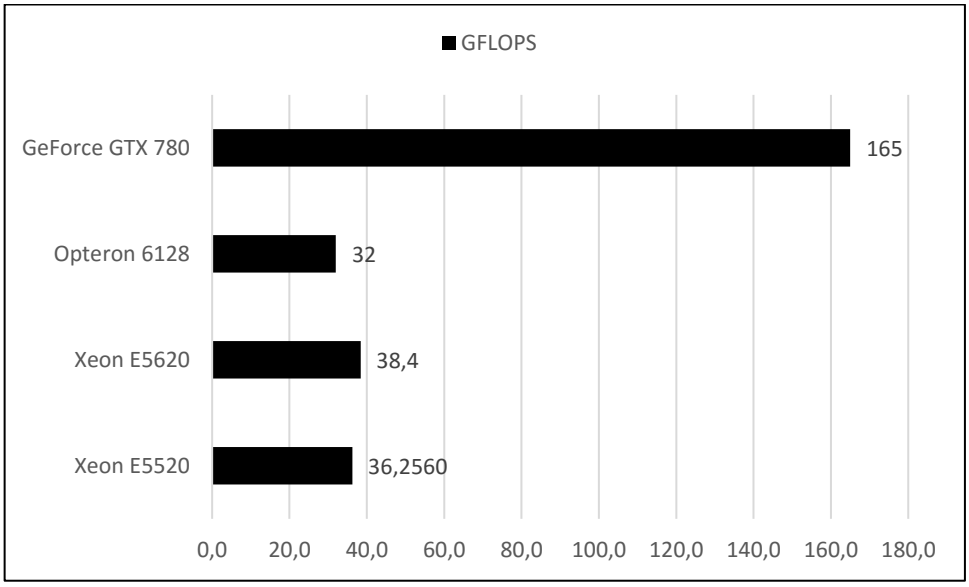
6.1 Analiza in rezultati računanja na različnih platformah

Molekulsko dinamiko smo pognali na več različnih platformah. Za primerjavo hitrosti procesorjev smo uporabili 3 različne CPE, na voljo smo imeli po dva procesorja vsakega tipa:

- AMD Opteron 6182: 8 jedrni procesor, 2.0 GHz.
- Intel Xeon E5620: 4 jedrni procesor z možnostjo večnitenja (*hyper-threading*), kar omogoča 4 dodatna navidezna jedra. Dela s hitrostjo 2.4 GHz
- Intel Xeon E5520: 4 jedrni procesor z možnostjo večnitenja, kar omogoča 4 dodatna navidezna jedra. Dela s hitrostjo 2.3 GHz

Poleg tega smo imeli na razpolago tudi dve grafični kartici, obe Nvidia GeForce GTX 780, ki vsebujeta vsaka 2304 računskih jeder in 3072 MB pomnilnika.

Slika 29 kaže največjo teoretično hitrost računanja v GFLOPS, podatki so pridobljeni s strani proizvajalcev.



Slika 29: Podatki o največji teoretični hitrosti računanja testiranih platform, v GFLOPS.

Na vseh CPE smo simulacijo molekulske dinamike pognali na 1, 4, 8 in 16 jedrih. Kot merilo primerjave smo uporabili podatek, koliko nanosekund simulacije je posamezna platforma sposobna izračunati v enem dnevu. Radij cut-off je bil nastavljen na 10 angstremov.

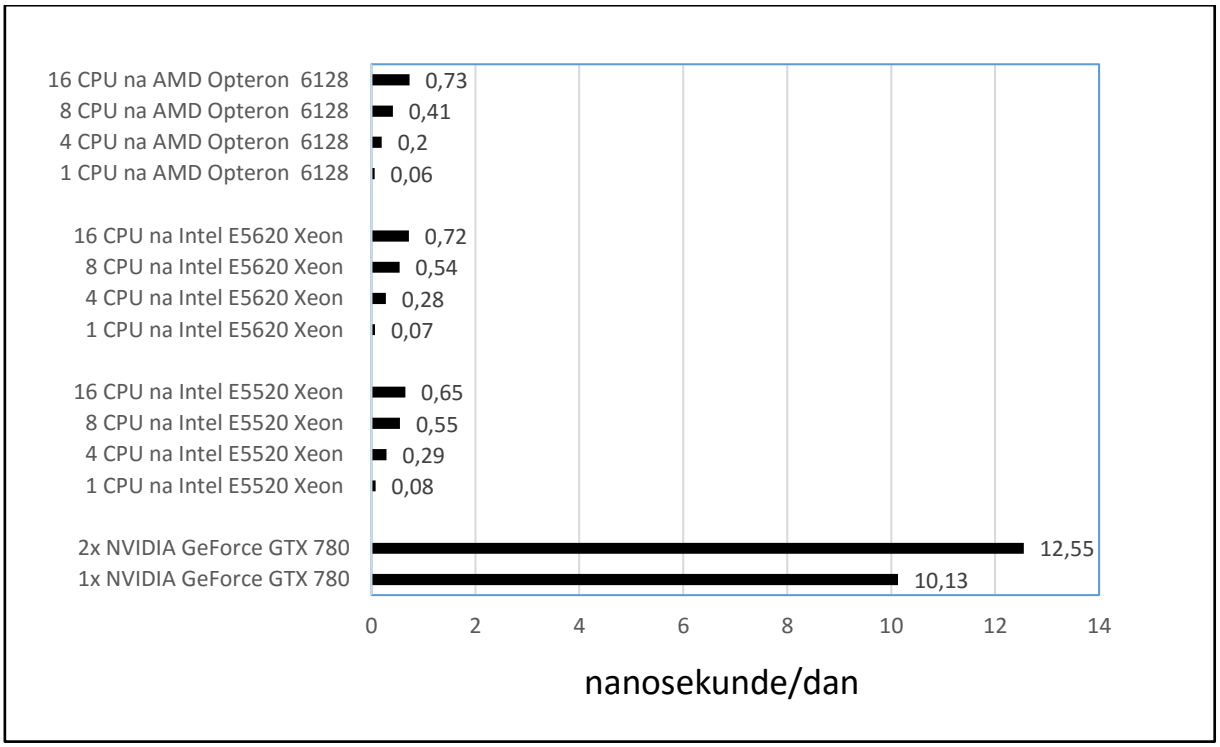
Rezultate časa računanja preberemo v izhodni datoteki simulacije. Primer izseka izhodne datoteke je prikazan na Sliki 30.

```
| NonSetup CPU Time in Major Routines:
|  Routine      Sec    %
|  ------------------------------
|  Nonbond    846.00   83.38
|  Bond         0.00  0.00
|  Angle        0.00  0.00
|  Dihedral     0.00  0.00
|  Shake        4.23    0.42
|  RunMD      164.26   16.19
|  Other        0.14    0.01
|  ------------------------------
|  Total      1014.65

| Final Performance Info:
|  -----------------------------------------------------
|  Average timings for all steps:
|    Elapsed(s) =    1064.70 Per Step(ms) =     10.68
|       ns/day =    8.11   seconds/ns =   10646.98
|  -----------------------------------------------------
```

Slika 30: Del izhodne datoteke simulacije.

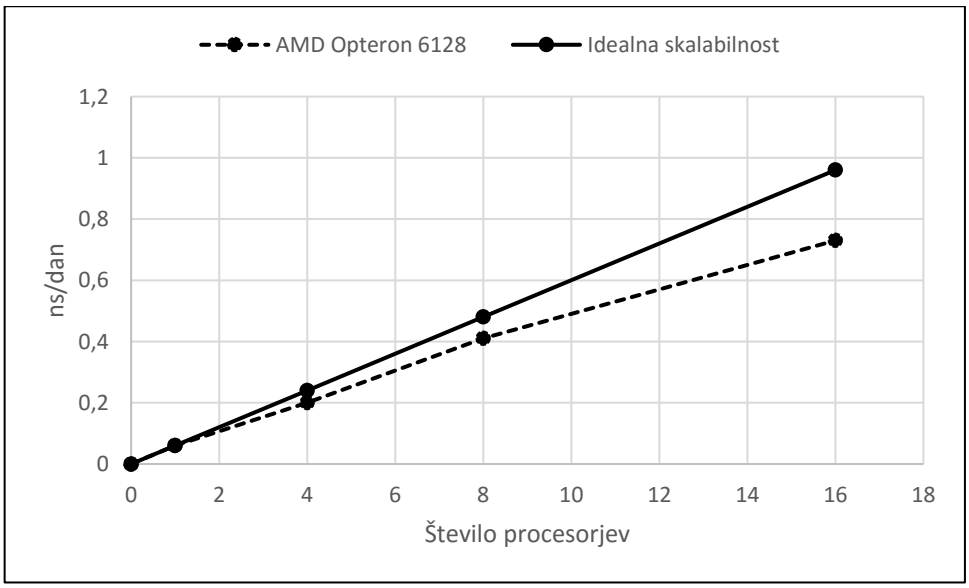
Rezultati vseh simulacij so prikazani na Sliki 31 in jasno je razvidno, da je simuliranje molekulske dinamike hidratiranega proteina daleč najhitrejše na GPE. Razliko v hitrosti lahko pripišemo veliki računski moči GeForce GTX 780, saj je s svojimi 2304 majhnimi procesorskimi jedri zelo primeren za izračun velikega števil neveznih interakcij, ki se računajo v vsakem koraku molekulske dinamike.



Slika 31: Hitrost simulacije na različnih platformah.

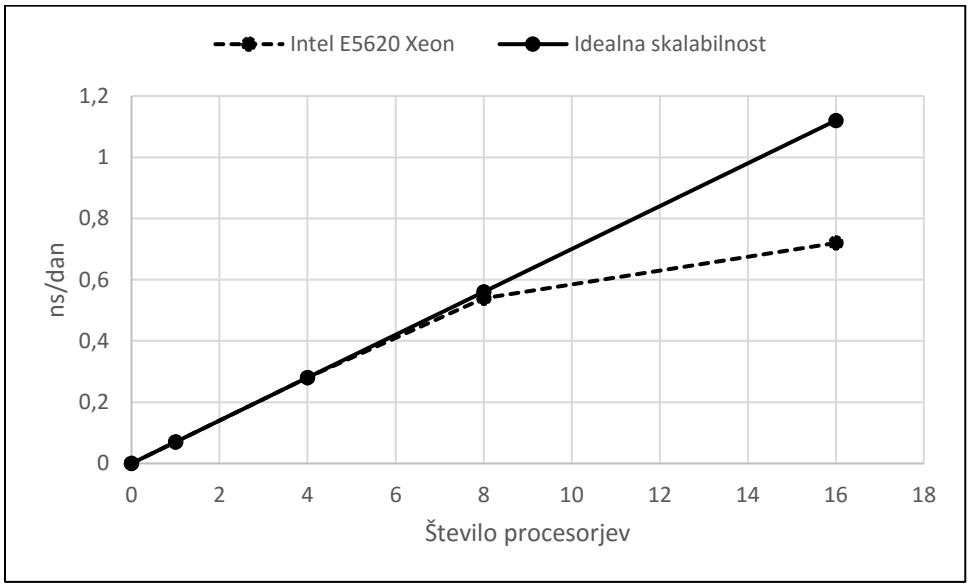
Skalabilnost

Pogledali smo učinkovitost računanja glede na skalabilnost sistema ob povečevanju števila procesorjev. Na sliki 32 je prikazana učinkovitost glede na procesor AMD Opteron 6182, na Sliki 33 glede na Intel E5620, na Sliki 34 glede na Intel E5520 in na Sliki 35 glede na GeForce GTX 780 GPE.

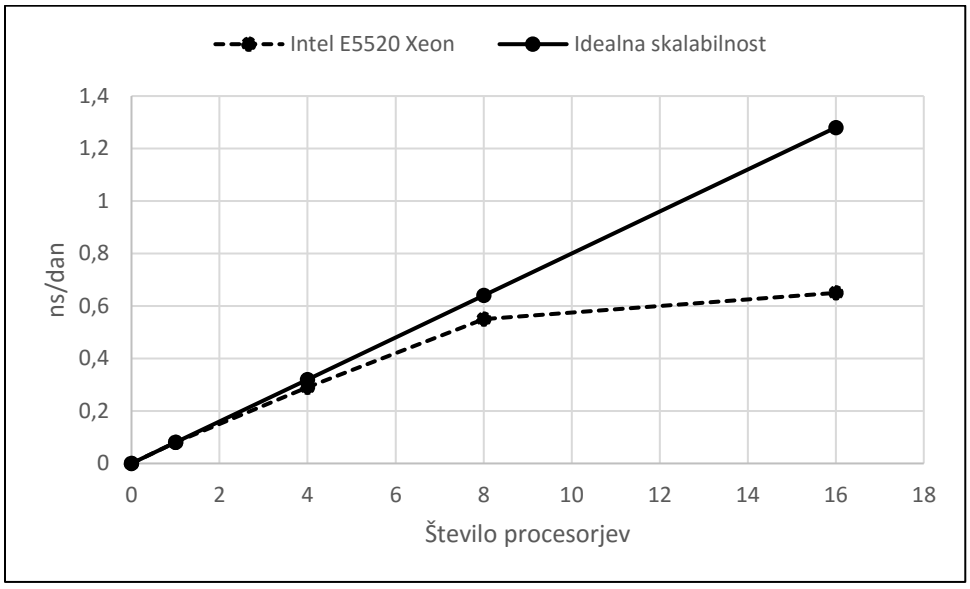


Slika 32: Skalabilnost na AMD Opteron 6128.

V premeru AMD Opterona vidimo, da je učinkovitost računanja z uporabo več jeder zelo dobra, še posebej ko uporabimo 8 procesorskih jeder. Pri uporabi 16 jeder se že vidi manjši padec učinkovitosti na jedro, kar je razumljivo, glede na to, da so jedra na dveh različnih procesorjih. Kljub temu je skalabilnost v tem primeru še vedno zadovoljiva. Predvidevamo lahko, da bi lahko dodali še vsaj 2 procesorja (16 jeder) za dodatno pohitritev in pričakovali dobro učinkovitost.



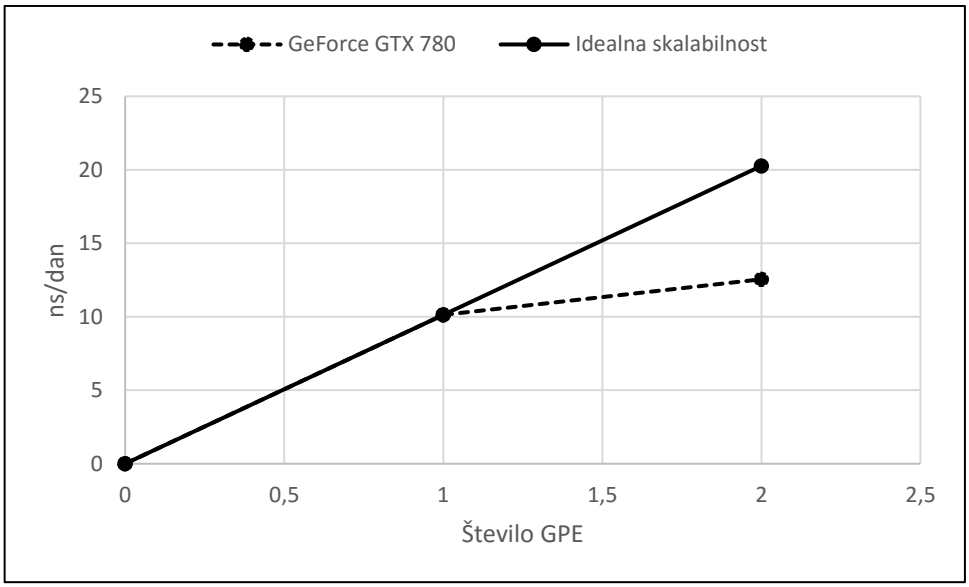
Slika 33: Skalabilnost na Intel E5620 Xeon.



Slika 34: Skalabilnost na Intel E5520 Xeon.

V primeru obeh Intelovih Xeonov vidimo, da je skalabilnost pri uporabi 8 jeder zelo dobra, vendar pa ob uporabi 16 jeder drastično pade. To lahko pripišemo temu, ker ima vsak procesor samo 4 fizična jedra in ob uporabi dodatnih navideznih (večnitenje) učinkovitost močno pade. Večnitenje namreč deluje tako, da se dodatna (navidezna) jedra dodelujejo samo, ko so vsa

fizična jedra zasedena [40] [41]. V našem primeru se ob uporabi 8 jeder simulacija izvaja na vseh osmih fizičnih jedrih, ob simuliranju na 16 jedrih pa se uporabi večnitenje. Zato je pohitritev iz 8 na 16 jeder tako majhna. Kljub slabši pohitritvi v tem primeru, lahko predvidevamo dobro učinkovitost pri dodajanju dodatnih fizičnih jeder.



Slika 35: Skalabilnost na GeForce GTX 780.

Vidimo, da je skalabilnost pri povečanju števila GPE zelo slaba, izračunan čas se namreč ob uporabi dveh GPE namesto ene poveča samo za 24% (10.13 ns/dan – 12.55 ns/dan).

Zanimalo nas je, zakaj je pohitritev računanja ob uporabi dveh GPE v primerjavi z uporabo ene GPE le 24%. Računanje z GPE poteka tako, da CPE deluje kot glavno procesorsko jedro, ki dodeluje ukaze in podatke GPE. Pri uporabi dveh GPE, se vsaki od njih dodeli lasten procesor, ki skrbi za dodeljevanje podatkov in ukazov. V tem smislu z uporabo dveh GPE pričakujemo večjo pohitritev. A ker je med simulacijo potrebna komunikacija med obema GPE, ki poteka preko pomnilnika obeh sodelujočih procesorskih jeder, predstavlja ta komunikacija ozko grlo. S programom Intel Performance Counter Monitor smo med izvajanjem simulacije spremljali dogajanje na CPE. Na Sliki 36 vidimo zasedenost procesorjev, levo pri simuliranju z eno GPE, desno pri simuliranju z dvema GPE. Vidimo, da je pri simuliranju z eno GPE aktivno samo eno procesorsko jedro (v danem primeru jedro 14), pri simuliranju z dvema GPE pa dve jedri (jedri 0 in 6), ki usklajujeta potek simulacije. IPC pomeni izvedeno število ukazov v urinem ciklu. Vidimo lahko, da je število ukazov v urinem ciklu pri uporabi dveh GPE višje, dodatno delo lahko pripišemo usklajevanju podatkov na pomnilniku posameznih jeder. S tem se zagotovi pravilnost podatkov, potrebna za izvajanje izračunov na GPE.

Jedro	IPC	Ukazi	Cikli
0	0.26	5135 K	19 M
1	0.20	195 K	959 K
2	0.24	104 K	432 K
3	0.12	108 K	893 K
4	0.06	1819 K	31 M
5	0.08	89 K	1184 K
6	0.28	105 K	377 K
7	0.13	172 K	1316 K
8	0.04	208 K	4947 K
9	0.37	136 K	368 K
10	0.28	84 K	301 K
11	0.38	1953 K	2508 K
12	0.27	441 K	1657 K
13	0.12	77 K	619 K
14	1.32	3474 M	2538 M
15	0.21	161 K	780 K
*	1.33	3485 M	2606 M

Jedro	IPC	Ukazi	Cikli
0	1.40	3554 M	2538 M
1	0.33	14 M	44 M
2	0.11	43 K	387 K
3	0.11	37 K	346 K
4	0.20	1846 K	9136 K
5	0.38	711 K	1852 K
6	1.39	3530 M	2538 M
7	0.27	139 K	512 K
8	0.19	59 K	308 K
9	0.17	249 K	1442 K
10	0.33	139 K	420 K
11	0.33	148 K	444 K
12	0.23	803 K	3504 K
13	0.46	872 K	1884 K
14	0.15	63 K	427 K
15	0.28	116 K	423 K
*	1.42	7105 M	5142 M

Slika 36: Levo dogajanje ob uporabi ene GPE, desno ob uporabi dveh GPE

Za izboljšanje učinkovitosti ob dodajanju GPE bi bilo potrebno izboljšati komunikacijo med GPE. Ena od možnosti je vzpostaviti direktno povezavo med GPE. V našem primeru, ko sta obe GPE v enem vozlišču (torej v eni delovni postaji), obstaja možnost direktne povezave med karticama z vodilom PCIe. Povezave med morebitnimi različnimi vozlišči pa bi lahko realizirali s povezavo Infiniband. Ob spremembah infrastrukture bi bila potrebna prilagoditev programske opreme, tako da bi paket Amber hitrejše povezave ustrezno izkoriščal.

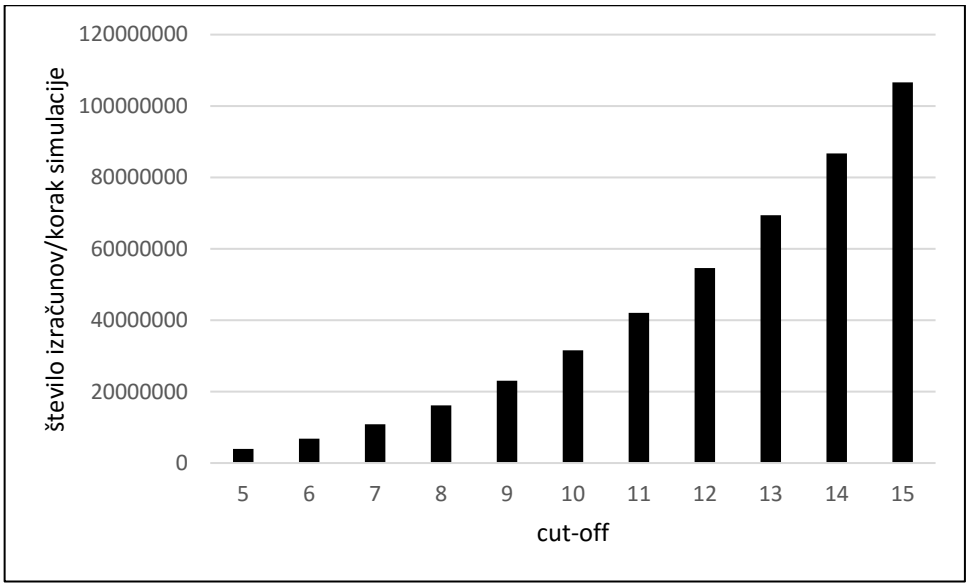
Spreminjanje velikosti problema s spreminjanjem radija cut-off

V nadaljevanju smo se zaradi veliko večje hitrosti računanja osredotočili na simuliranje molekulske dinamike z GPE. Z uporabo radija cut-off smo povečevali in zmanjševali potrebna števila izračunov. Z uporabo Enačbe 17 smo izračunali število potrebnih izračunov v posameznem koraku molekulske dinamike. Potrebne izračune prikazujeta Tabela 1 in Slika 37.

$$n_{calc} = N \times \frac{4}{3} \times \pi \times r_{cut-off}^3 \times \rho \qquad (17)$$

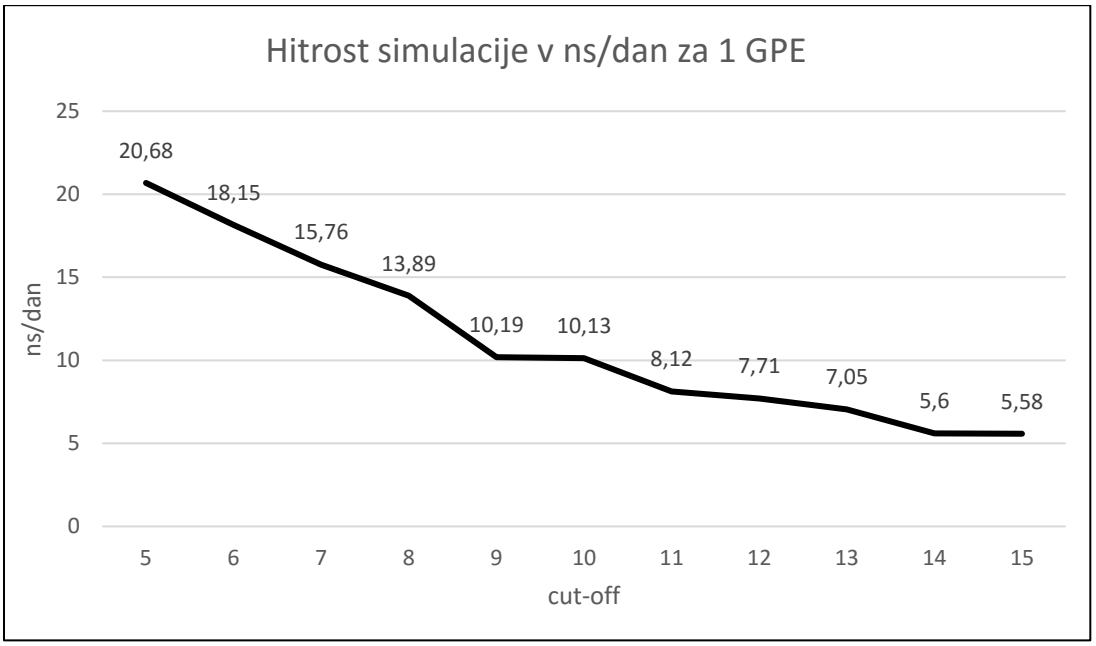
Radij cut-off	Število izračunov 1 korak (v tisočih)
5	3950
6	6826
7	10840
8	16181
9	23039
10	31604
11	42065
12	54612
13	69434
14	86721
15	106663

Tabela 1: Potrebno število izračunov v enem koraku simulacije



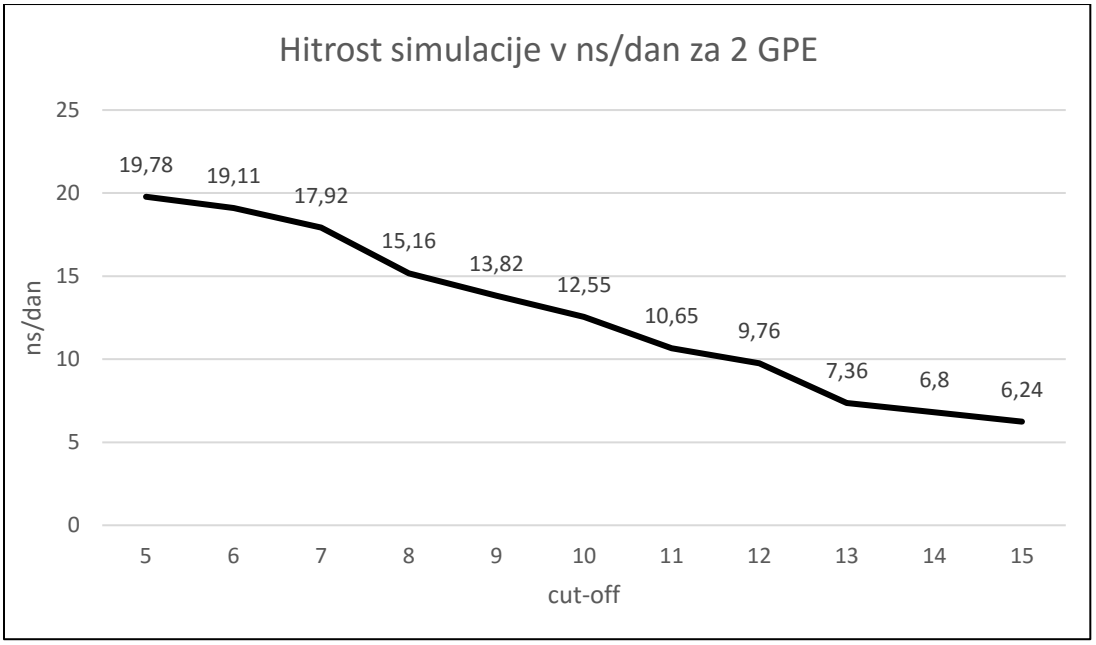
Slika 37: Število izračunov v vsakem koraku simulacije glede na radij cut-off

Radij cut-off pomeni, do katere razdalje izračunavamo nevezne interakcije. Koncept se je uveljavil, da omejimo procesorski čas, saj je večina procesorskega časa povezanega z molekulsko dinamiko porabljenega za računanje neveznih interakcij. V praksi uporabljamo cut-off med 8 in 15 angstremi. Hitrost simulacije v odvisnosti od radija cut-off na eni GPE je prikazana na Sliki 38.



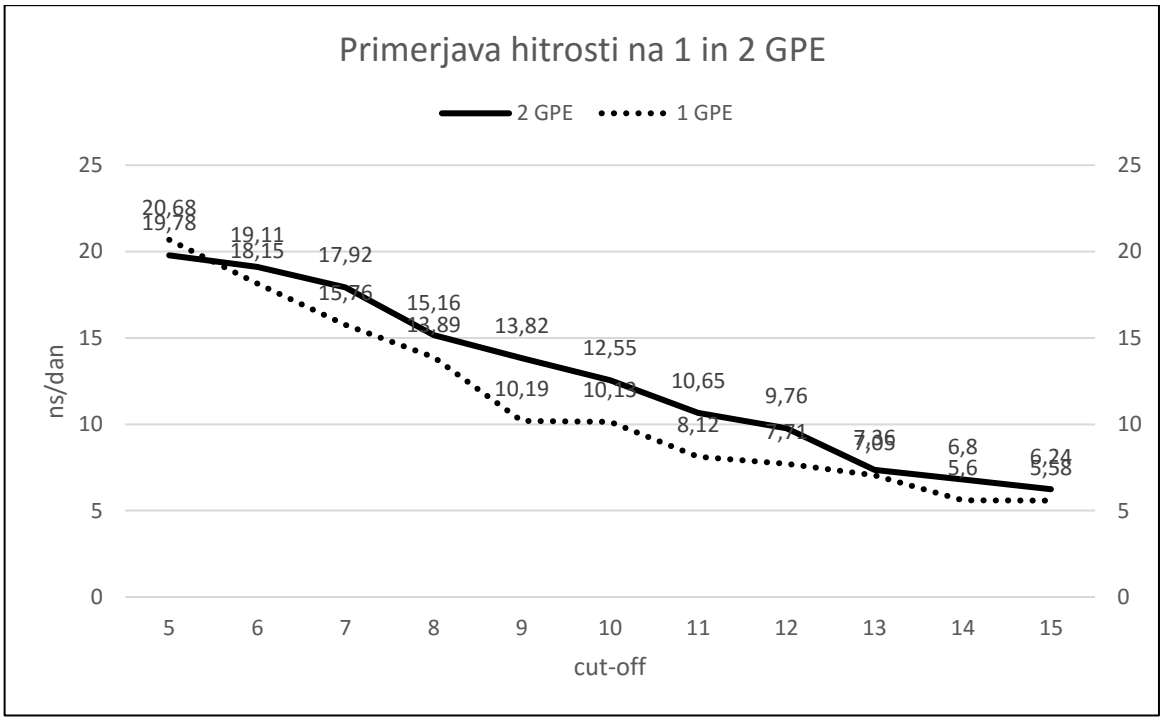
Slika 38: Hitrost simulacije na eni GPE za različne vrednosti cut-off radija.

Odvisnost hitrosti simulacije od radija cut-off za dve uporabljeni GPE je prikazana na Sliki 39. Hitrost procesiranja se tako kot v primeru ene GPE zmanjšuje s povečevanjem radija cut-off.



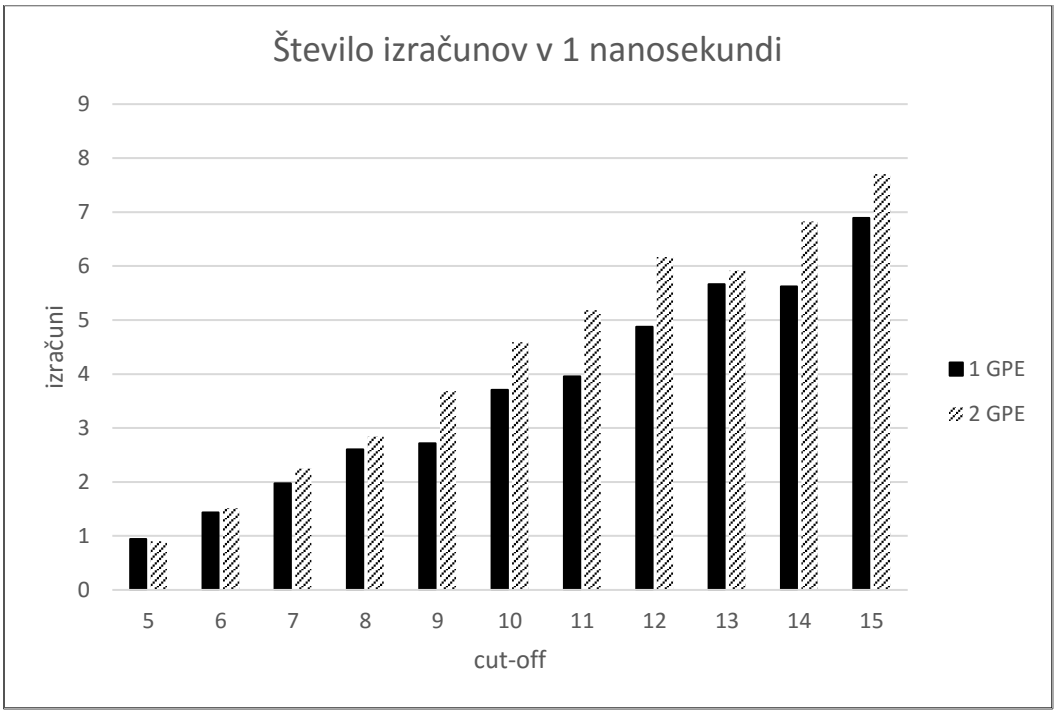
Slika 39: Hitrost simulacije na dveh GPE za različne vrednosti cut-off radija.

Superpozicijo hitrosti računanja za različne vrednosti radija cut-off za eno in dve uporabljeni GPE prikazuje Slika 40. Sporočilo je, da je pohitritev ob hkratni uporabi dveh GPE v primerjavi z eno GPE razmeroma majhna.



Slika 40: Primerjava hitrosti simulacije na 1 in 2 GPE za različne vrednosti cut-off radija.

V obeh primerih vidimo, da število izračunov v ns/dan pada počasneje, kot raste skupno število izračunov. Na primer pri vrednosti 15 radija cut-off, je potrebnih kar 27 krat več izračunov na korak simulacije kot pri vrednosti 5 radija cut-off. Število izračunov v ns/dan pa se zmanjša za samo 3,7 krat (iz 20,68 – 5,58). Glede na število potrebnih izračunov in na čas trajanja simulacije, smo izračunali, koliko izračunov se opravi v vsaki nanosekundi simulacije. Rezultati so prikazani na Sliki 41.



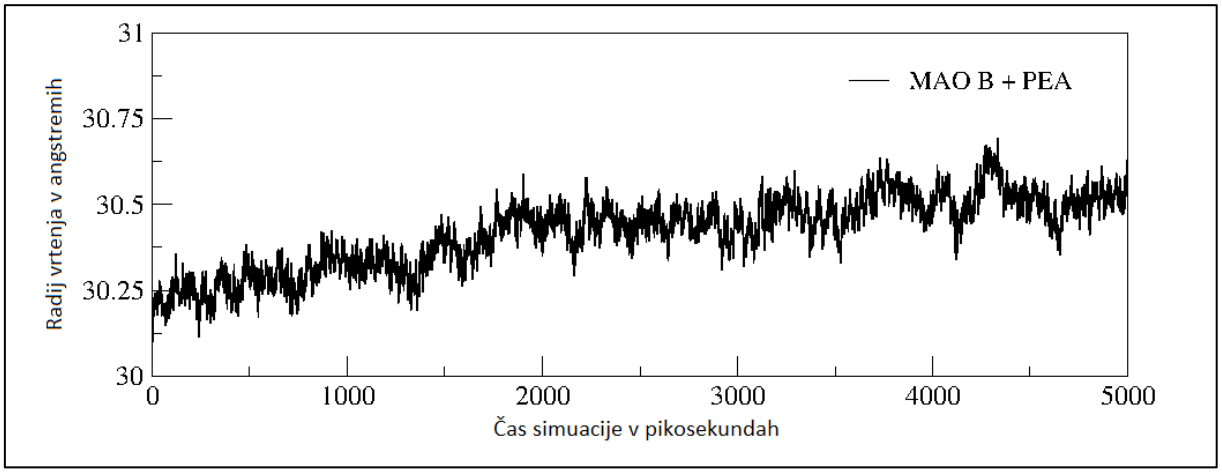
Slika 41: Število opravljenih izračunov v 1 nanosekundi ob različnih vrednosti radija cut-off.

Rezultati kažejo, da se učinkovitost računanja povečuje, ko GPE naložimo večje količine računov. Iz tega sklepamo, da so računske kapacitete GPE pri manjšem številu izračunov slabše izkoriščene, najverjetneje zaradi več potrebnih dostopov do pomnilnika in komunikacije med CPE in GPE.

6.2 Analiza rezultatov molekulske dinamike

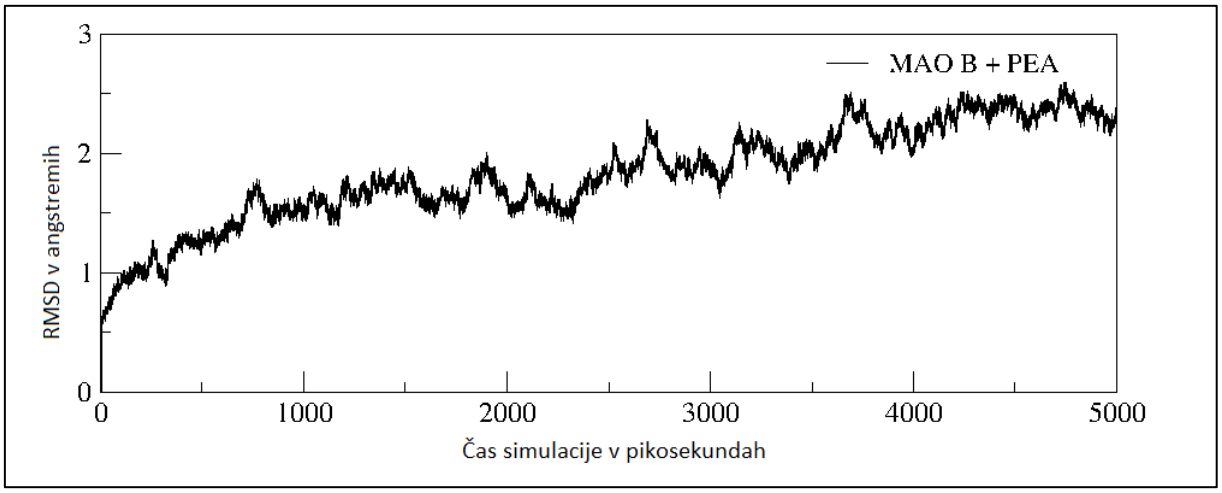
V tem poglavju predstavljamo vsebinske rezultate simulacije, zanimive iz vidika kemije.

Analiza trajektorije je ključni element simulacije molekulske dinamike. Zelo ilustrativna količina je radij vrtenja (*radius of gyration*). Radij vrtenja je merilo za velikost proteina, ki ga aproksimiramo s kroglo. Majhen radij vrtenja pomeni majhno in kompaktno strukturo, zelo velik radij pa predstavlja razvito strukturo. Če radij vrtenja narišemo kot funkcijo časa, pričakujemo, da se po določenem času stabilizira in fluktuira okrog srednje vrednosti. Slika 42 kaže, da je simulacija postala nekoliko stabilnejša po 3 nanosekundah, za natančnejši rezultat bi bila potrebna daljša simulacija.



Slika 42: Radij vrtenja kot funkcija časa. Od 0-3000 pikosekunde se je radij vrtenja počasi povečeval, od 3000-5000 pikosekunde pa bolj ali manj fluktuira okrog srednje vrednosti.

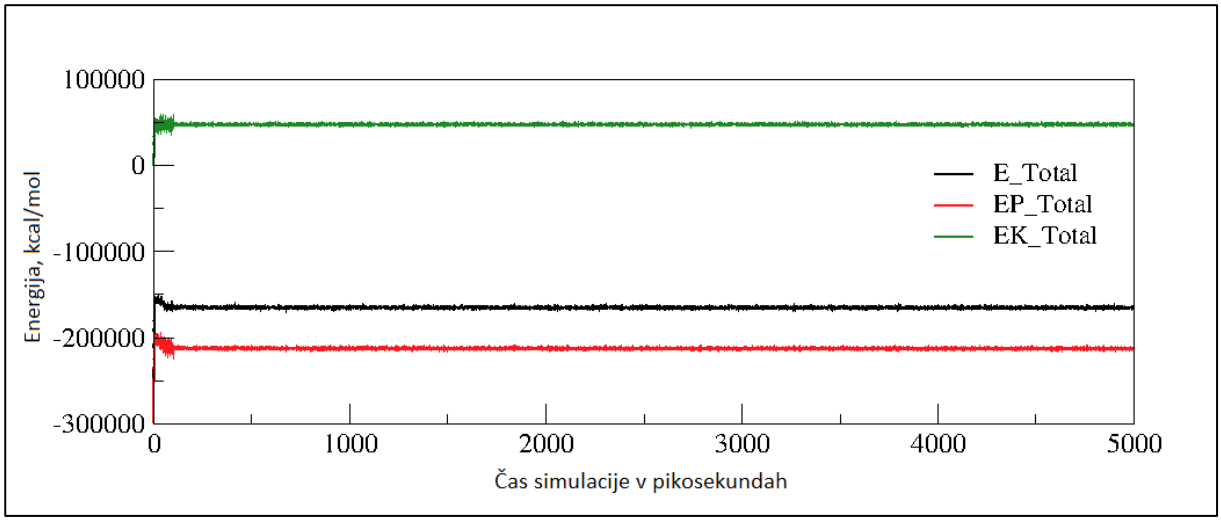
Zelo podobna količina kot je radij vrtenja je srednja vrednost odklona glede na začetne koordinate (*Root Mean Square Deviation*, RMSD). Za razliko od radija vrtenja je ta odvisna tudi od rotacijskega gibanja okrog centra mase. Iz trajektorije smo izračunali to količino in jo narisali kot funkcijo simulacijskega časa (Slika 43).



Slika 43: Srednja vrednost odklona od začetne vrednosti koordinat kot funkcija časa.

RMSD se je med simulacijo počasi povečevala, a v nobeni točki vrednost ni bistveno presegla 2,5 Å, kar kaže na uravnoteženje sistema. Med 3000-5000 pikosekundo se je RMSD za razliko od radija vrtenja še vedno rahlo povečevala, kar kaže na rotacijsko gibanje proteina.

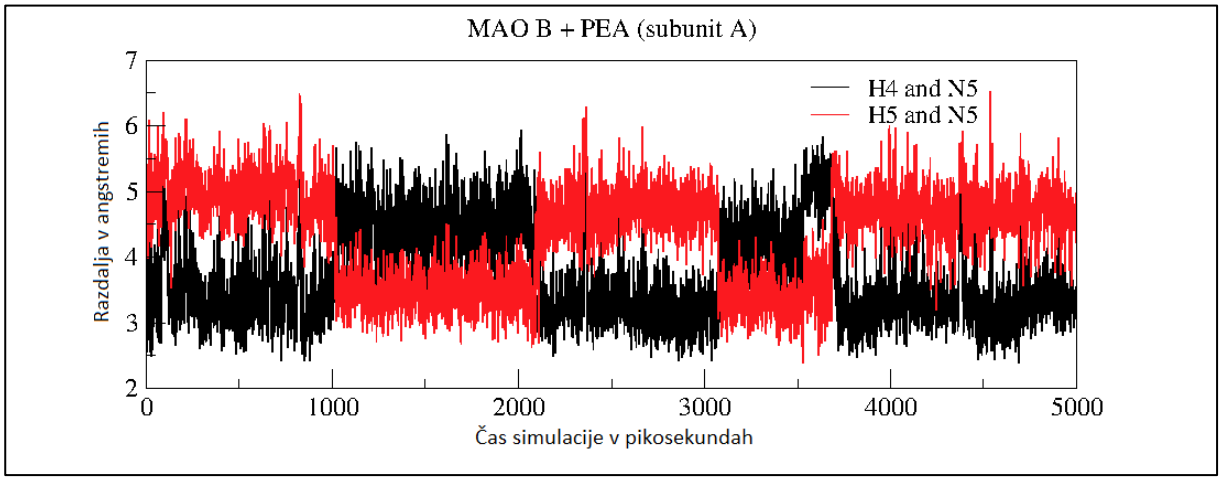
Komponente energij so pomemben parameter stabilnosti simulacije. Ker je naš simulirani sistem ogromen, pričakujemo majhne fluktuacije kinetične in potencialne energije. Kinetična, potencialna in celokupna energija sistema so prikazane na Sliki 44.



Slika 44: Kinetična, potencialna in celotna energija sistema kot funkcija časa.

Energijske komponente prikazane na sliki kažejo popolno stabilizacijo po 100 pikosekundah. Stabilnost celokupne energije navaja na misel, da je uporabljeni integrator dal stabilno trajektorijo, da je obravnava daljnosežne elektrostatike dobra in da lahko uporabljeni simulacijski protokol uporabljamo za simulacije vezave ligandov. Primerjava med energijo kot funkcijo časa in geometričnimi parametri kot funkcijo časa kaže, da slednji počasneje konvergirajo kot energija.

Razdalja kot funkcija časa med vodikovimi atomi reaktivne metilenske skupine feniletilamina in reaktivnim N5 atomom lumiflavinskega kofaktorja je prikazana na Sliki 45.



Slika 45: Razdalja kot funkcija časa med vodikovimi atomi reaktivne metilenske skupine fenilatilamina in reaktivnim N5 atomom lumiflavinskega kofaktorja.

Slika 45 kaže, da se reaktivna vodikova atoma v približno enaki meri približujeta reaktivnemu atomu N5. Nizka razdalja je pogoj za kemijsko reakcijo. Ker sta obe razdalji nizki v primerljivi količini časa, to navaja na misel, da ne moremo predvideti stereoselektivnega produkta.

7 Zaključki

V diplomskem delu smo izvedli simulacijo molekulske dinamike hidratiranega proteina MAO B, kompleksiranega z endogenim substratom feniletilaminom. Simulirani sistem je izjemno relevanten za področje nevroznanosti, saj ta encim regulira nivoje biogenih aminov, ki imajo ključno vlogo pri prenosu živčnega signala. Naš namen je bil kritična primerjava različnih računalniških platform pri izvedbi simulacije molekulske dinamike.

Rezultati iz vidika kemije nam na osnovi spremljanja radija vrtenja pokažejo, da je potrebno za uravnoteženje hidratiranega encima MAO B 3 nanosekunde dolgo uravnovešenje, ki mu sledi produkcijski del simulacije. Rezultat je pomemben za računsko obravnavo mutantov tega encima, ki imajo različne konstante reakcijske hitrosti. Razumevanje povezave med strukturo encima, kemijsko reaktivnostjo pri razgradnji biogenih aminov in klinično sliko spada že v področje genomske medicine, ki prvič v zgodovini medicine daje možnost personaliziranega zdravljenja (Precision Medicine). Biomolekularne simulacije bodo nedvomno odigrale zelo pomembno vlogo pri razvoju tega področja.

Na osnovi rezultatov meritev smo ugotovili, da je visoko zmogljiva GeForce GTX 780 GPE z vidika hitrosti najbolj primerna platforma za izvajanje klasičnih molekulskih simulacij. Pokazali smo, da je smiselna uporaba le ene GPE hkrati, saj je pohitritev ob uporabi dveh GPE samo 24%. Trenutno je komunikacija med različnimi GPE, ki poteka s pomočjo CPE, ozko grlo pri simulacijah molekulske dinamike. V prihodnosti se kažejo rešitve v obliki direktnih povezav med GPE. Skalabilnost je pri uporabi CPE boljša, vendar bi za dosego podobnih rezultatov potrebovali veliko število procesorskih jeder, kar je bistveno dražja rešitev.

Velikost našega sistema smo nato spreminjali s pomočjo radija cut-off in ugotovili, da so GPE pri manjših obremenitvah manj učinkovite in obratno: več kot je potrebnih izračunov, bolj učinkovite so.

Na osnovi te ugotovitev so se sodelavci Odseka za računsko biokemijo in načrtovanje učinkovin odločili, da bodo v prihodnosti večje količine sredstev namenili za investicijo v GPE, kar bo izjemno pohitrilo molekulske simulacije hidratiranih bioloških makromolekul.

Literatura

[1] L. Amundsen, M. Landro in B. Arntsen, „Supercomputing,“ *GEO ExPro Volume 12,* Izv. 5, pp. 50-53, 2015.

[2] „Top500 Project,“ [Elektronski]. Available: www.top500.org. [Poskus dostopa April 2016].

[3] T. G. P. Mell, „The NIST Definition of Cloud,“ *NIST Special Publication 800-145,* Izv. Peter Mell, 2011.

[4] D. Milojicic, „High Performance Computing (HPC) in the Cloud,“ 2012. [Elektronski]. Available: https://www.computer.org/web/computingnow/archive/september2012. [Poskus dostopa april 2016].

[5] L. M. Vaquero, L. Rodero-Merino, J. Caceres in M. Lindner, „A break in the clouds: towards a cloud definition,“ *ACM SIGCOMM Computer Communication Review,* Izv. Volume 39 Issue 1, pp. 50-55, 2009.

[6] P. Strong, „Enterprise Grid Computing,“ *Distributed Computing Volume 3,* Izv. 6, pp. 50-59, 2005.

[7] B. Jones, „CERN Accelerating science,“ 2008. [Elektronski]. Available: https://edms.cern.ch/ui/#!master/navigator/document?D:1752431380:1752431380:subDocs. [Poskus dostopa junij 2016].

[8] „Nvidia visual computing,“ [Elektronski]. Available: http://www.nvidia.com/object/visual-computing.html. [Poskus dostopa april 2016].

[9] „CUDA C Programming guide,“ [Elektronski]. Available: https://docs.nvidia.com/cuda/cuda-c-programming-guide/. [Poskus dostopa april 2016].

[10] J. E. Stone, J. C. Phillips, P. L. Freddolino, D. J. Hardy, L. G. Trabuco in K. Schulten, „Accelerating Molecular Modeling Applications with Graphics Processors,“ *Journal of Computational Chemistry,* Izv. 28, p. 2618–2640, 2007.

[11] J. Fung in S. Mann, „Using Multiple Graphics Cards as a General Purpose Parallel Computer: Applications to Computer Vision,“ v *Pattern Recognition, 2004. ICPR 2004*, 2004.

[12] M. W. Brown, P. Wang, S. J. Plimpton in A. N. Tharrington, „Implementing molecular dynamics on hybrid high performance computers – short range forces,“ *Computer Physics Communications Volume 128,* Izv. 4, pp. 898-911, 2011.

[13] J. Nickolls, I. Buck, M. Garland in K. Skadron, „Scalable parallel programming with cuda,“ *Queue,* Izv. 6, pp. 40-53, 2006.

[14] E. Lindholm, J. Nickolls, S. Oberman in J. Montrym, „NVIDIA Tesla: A Unified Graphics and Computing Architecture," *IEEE Micro, Volume 28,* Izv. 2, pp. 39-55, 2008.

[15] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone in J. C. Phillips, „GPU Computing," *Proceedings of the IEEE, volume 96,* Izv. 5, pp. 879 - 899, 2008.

[16] K. Y. Sanbonmatsu, S. Joseph in C.-S. Tung, „Simulating movement of tRNA into the ribosome during decoding," *Proceedings of the National Academy of Sciences of USA,* Izv. 102, p. 15854–15859, 2005.

[17] A. W. Götz, M. J. Williamson, D. Xu, D. Poole, S. Le Grand in R. C. Walker, „Routine Microsecond Molecular Dynamics Simulations with AMBER on GPUs. 1. Generalized Born," *Journal of Chemical Theory and Computation,* Izv. 8, p. 1542–1555, 2012.

[18] N. Pydiura, P. Karpov in Y. Blume, „On the Efficiency of CPU and Hybrid CPU-GPU Systems," *Computer Science and Applications,* Izv. 1, pp. 48-59, 2014.

[19] J. Glaser, T. D. Nguyen, J. A. Anderson, P. Lui, F. Spiga, J. A. Millan, D. C. Morse in S. C. Glotzer, „Strong scaling of general-purpose molecular dynamics simulations on GPUs," *Computer Physics Communications,* Izv. 192, pp. 97-107, 2015.

[20] Q. Hou, M. Li, Y. Zhou, J. Cui, Z. Cui in J. Wang, „Molecular dynamics simulations with many-body potentials on multiple GPUs—The implementation, package and performance," *Computer Physics Communications Volume 184,* Izv. 9, p. 2091–2101, 2013.

[21] A. B. Bondi, „Characteristics of Scalability and Their Impact on Performance," *Proceedings of the 2nd international workshop on Software and performance,* pp. 195-203, 2000.

[22] M. D. Hill, „What is scalability?," *ACM SIGARCH Computer Architecture News,* Izv. Volume 18 Issue 4, pp. 18-21, 1990.

[23] J. Kowalik, „Efficiency and Scalability in High Performance Computing," April 2008. [Elektronski]. Available: http://www.hpcwire.com/2008/04/18/intel_ibm_speed_through_economic_slowdown-1/. [Poskus dostopa Maj 2016].

[24] „Livermore Computing Center," [Elektronski]. Available: https://computing.llnl.gov/tutorials/parallel_comp/. [Poskus dostopa 23 Maj 2016].

[25] J. M. Berg, J. L. Tymoczko in L. Stryer, Biochemistry, W.H. Freeman and Company, 2002.

[26] R. F. Boyer, Temelji biokemije, Študentska založba, 2008.

[27] „Boundless, Boundless Biology,“ 8 marec 2016. [Elektronski]. Available: https://www.boundless.com/biology/textbooks/boundless-biology-textbook/biological-macromolecules-3/proteins-56/amino-acids-303-11436/. [Poskus dostopa april 2016].

[28] H. P. Rang, J. M. Ritter, R. J. Flower in G. Henderson, Rang and Dale's Pharmacology, 8th Edition, Churchill Livingstone, 2015.

[29] M. Repič, „Računalniško modeliranje katalitičnega mehanizma in zaviranja monoamin-oksidaze,“ *Doktorska disertacija,* 2014.

[30] M. L. 6. š. 2. Anderluh, „Pregled zdravilnih učinkovin za zdravljenje depresije,“ *Farmacevtski vestnik,* Izv. 2, pp. 66-72, 2010.

[31] M. Trošt, „Parkinsonova bolezen,“ *Farmacevtski vestnik,* Izv. 2, pp. 60-63, 2008.

[32] A. Szabo, B. E in T. J, „Phenylethylamine, a possible link to the,“ *Br J Sports Med,* Izv. 35, pp. 342-343, 2001.

[33] U. Borštnik, Vzporedne računalniške simulacije na gručah osebnih računalnikov - doktorska disertacija, 2006.

[34] R. Salomon-Ferer, D. A. Case in R. C. Walker, „An overview of the Amber biomolecular simulation package,“ *WIREs Comput Mol Sci,* 2012.

[35] S. Pfalzner in P. Gibbon, Many-Body Tree Methods in Physics, Cambridge: Cambridge University Press, 1996.

[36] A. R. Leach, Molecular Modelling: Principles and Applications, Essex: Addison Wesley Longman Limited, 1996.

[37] D. Case, J. Berryman, R. Betz, D. Cerutti, I. T.E. Cheatham, T. Darden, R. Duke in T. Giese, Amber 2015 Reference Manual, University of California, San Francisco, 2015.

[38] „Amber Home Page,“ [Elektronski]. Available: http://ambermd.org/. [Poskus dostopa 14 Maj 2016].

[39] B. D. Madej in R. Walker, „An Introduction to Molecular Dynamics Simulations using AMBER,“ [Elektronski]. Available: http://ambermd.org/tutorials/basic/tutorial0/. [Poskus dostopa 17 maj 2016].

[40] D. T. Marr, F. Binns, D. L. Hill, G. Hinton, D. A. Koufaty, J. A. Miller in M. Upton, „Hyper-Threading Technology Architecture and Microarchitecture.,“ *Intel Technology Journal Vol. 6,* Izv. 1, pp. 1-12, 2002.

[41] Y. Wakisaka, N. Shibata, K. Yasumoto in M. Ito, „Task Scheduling Algorithm for Multicore Processor Systems with Turbo Boost and Hyper-Threading,“ v *The Steering Committee of The*

World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA), 2014.