Univerza v Ljubljani

Fakulteta za računalništvo in informatiko

Urban Borštnik

Vzporedne računalniške simulacije na gručah osebnih računalnikov

DOKTORSKA DISERTACIJA

Mentor: prof. dr. Borut Robič

Somentorica: univ. znan. svet. dr. Dušanka Janežič

Ljubljana, 2007

Zahvala

Za nastanek doktorskega dela se moram zahvaliti mnogim, ki so me že od nekdaj navdušili, da svet opazujem, preučujem in poskušam razumeti. K tej radovednosti so mi vedno sodila vprašanja, kako kaj deluje in zakaj, pa naj bo to pojem ali naprava, in moji neuspešni in uspešni poskusi, da kakšno stvar izboljšam. Vedno pa sem se kaj novega naučil, četudi so bili izidi proti mojim pričakovanjem.

Za mentorstvo pri nastanku disertacije dolgujem zahvalo mentorju prof. dr. Borutu Robiču. Velika zahvala pripada somentorici univ. znan. svet. dr. Dušanki Janežič za vodenje pri doktorskem delu in mentorstvo pri disertaciji, za vpeljavo v raziskovalno in znanstveno delo in za številne ideje pri raziskavah. Zahvaliti se moram tudi Kemijskemu inštitutu, kjer sem opravljal doktorsko delo in Javni agenciji za raziskovalno dejavnost Republike Sloveniji za moje financiranje po programu mladega raziskovalca.

Brez zgodnjih vtisov računalnikov se morda ne bi odločil raziskovati računalništvo. Mag. Damir Cibic mi je prvi pokazal, da je računalnik več kot le »škatla«, ko me je v neki majhni univerzitetni sobi očaral z grafičnem prikazom simulacije Sončnega sistema. Za navdušenje nad simulacijo molekulske dinamike se moram pa zahvaliti predvsem dr. Milanu Hodoščku, ki mi jo je predstavil in na ravno pravi način razložil, kasneje pa vpeljal v svet vzporednega računanja in računalniških gruč. Poleg tega mi je bil vedno pripravljen razložiti odgovore na moja raznolika vprašanja. Hkrati gre zahvala tudi dr. Bernardu R. Brooksu, ki mi je omogočil, da sem v času šolanja prihajal v njegov laboratorij in se spoznaval s simulacijami in računalniki za njihovo računanje; najbolj pa za izjavo »Let's build a force decomposition machine!«, saj sem s tem lahko povezal svoje dosedanje raziskovalno delo v celoto.

Ker je namen simulacije molekulske dinamike razumevanje kemije, podlaga fizika, računanje pa domena računalništva in matematike, se moram za obširne razlage zahvaliti predvsem sodelavcem na Kemijskem inštitutu, ki so mi sproti dopolnjevali znanje teh in drugih ved: poleg dr. Janežič še dr. Franci Merzel, dr. Matej Praprotnik, dr. Janez Mavri, dr. Jernej Stare, dr. Franc Avbelj, Urban Bren in Andrej Perdih.

Snov narejena brez duše ni zanimiva, prav tako pa so mi bližnji obogatili čas med šolanjem in raziskovanjem. Zahvaliti se moram svojim staršem, Ireni in Miklavžu, ki sta mi vedno omogočala sledenje svoji radovednosti, ker sta mi nudila podporo, ko sem jo potreboval in ker sta me spodbujala, ko sta vedela, da jo potrebujem. Brez njiju ne bi mogel napisati prvega odstavka zahvale. Kristina mi je kot starejša sestra marsikdaj nevede dala zgled in mi omogočala, da sem v življenju videl dlje. Prijatelji pa so mi omogočili »pokukati« izza ograje znanost in z delitvijo izkušenj in mnenj so mi pomagali reševati težave na svoji poti.

Joži, kakšna zahvala sploh še ostane tebi? Del vsake do sedaj naštete, skupaj pa največja. S svojo ljubeznijo si mi omogočila, da sem si razširil življenje izven meja znanosti, s tem sem si pa obseg znanosti le še povečal. Hvala, Joži.

Povzetek

Vzporedni računalniški programi se pogosto uporabljajo za pohitritev izvajanja računsko zahtevnih znanstvenih nalog, kot so na primer računalniške simulacije. Take simulacije omogočajo teoretske raziskave v molekularni kemiji, strukturni biologiji in drugih področjih znanosti. Simulacija molekulske dinamike predstavlja reševanje diferencialnih enačb drugega reda, s katerimi opisujemo gibanje atomov simuliranega sistema. Ker ta problem v splošnem ni analitično rešljiv, ga obravnavamo numerično. Rešiti moramo sistem gibalnih enačb za vsak atom posebej. Osnovni rezultat simulacije je trajektorija gibanja atomov.

Metode za simulacijo molekulske dinamike so časovno zahtevne, kar omejuje tako dolžino simulacije kot velikost obravnavanega sistema. Računsko najbolj zahtevni del simulacij MD je računanje interakcij med atomi. Vseh interakcij v sistemu z N atomi je N^2. Na vsakem integracijskem koraku moramo izračunati te interakcije oziroma sile, ki delujejo na atome. Obstaja več načinov, da zmanjšamo računsko zahtevnost računanja interakcij. Med njimi je upoštevanje razdalje *cut-off*: kadar je razdalja med dvema atomoma večja od razdalje *cut-off*, njune interakcije ne upoštevamo in je ne računamo.

Vzporedne metode za simulacijo molekulske dinamike porazdelijo po $|P|$ procesorjih vzporednega računalnika računanje sil med atomi. Glede na način porazdelitve računanja sil med atomi delimo te vzporedne metode na tri vrste: replicirani podatki (ang. replicated data), delitev sil (ang. force decomposition) in prostorsko delitev (ang. space decomposition). Pri vseh teh metodah si morajo procesorji med posameznimi koraki simulacije izmenjevati podatke.

Pri *repliciranih podatkih* procesorji usklajujejo vse podatke med celotno simulacijo. Procesorjem enakomerno porazdelimo $N/|P|$ atomov, katerim računajo koordinate. Vsak procesor računa delež $1/|P|$ vseh interakcij; če ne upoštevamo *cut-off*, vsak procesor računa $N^2/(2|P|)$ interakcij. Na vsakem koraku simulacije si mora vseh $|P|$ procesorjev izmenjati podatke o vseh N atomih.

Pri metodi *delitev prostora* vsakemu procesorju dodelimo en del simulacijskega prostora. Za računanje sil mora vsak procesor izmenjavati podatke z najmanj 26 naj-

bližjimi sosedi, izmenjujejo pa podatke o $(N/|P|)^{2/3}$ atomih, če je razdalja *cut-off* dovolj majhna. Če ne upoštevamo razdalje *cut-off* in zahtevamo računanje vseh interakcij, potem morajo procesorji komunicirati z več kot le 26 sosednjimi procesorji. To lahko privede celo do enakih komunikacijskih zahtev kot pri repliciranih podatkih, ker si potem vsi procesorji izmenjujejo podatke o vseh atomih.

Pri metodi *delitev sil* ima vsak procesor podatke o $2N/\sqrt{|P|}$ atomih. Procesor komunicira le z $2\sqrt{|P|}$ drugimi procesorji in na vsakem koraku z njimi izmenjuje podatke o $2N/\sqrt{|P|}$ atomih. Več ko je procesorjev, manj podatkov izmenjujejo.

Razvil sem novo vzporedno metodo za simulacijo molekulske dinamike, *delitev sil s porazdelitvijo diagonale*. N^2 interakcij med vsemi atomi ponazorimo z matriko sil velikosti $N \times N$. Atome razdelimo po $|B|$ blokih; tako dobimo $|B|^2$ produktov blokov, ki predstavljajo interakcije med atomi dveh blokov. Ti produkti blokov tvorijo mrežo velikosti $|B| \times |B|$ v matriki sil. Produkte porazdelimo po procesorjih, tako da vsak računa interakcije v svojem produktu. Polovico interakcij, tiste v zgornjem trikotniku matrike sil, ne računamo, saj so obratno zrcalne interakcijam v spodnjem trikotniku. Sile ob diagonali matrike sil ustrezajo interakcijam med atomi istega bloka. Računanje teh sil porazdelimo po preostalih procesorjih, ki imajo ustrezne podatke. Metoda delitev sil s porazdelitvijo diagonale tako omogoča uporabo manjšega števila procesorjev pri danem številu blokov. Ker se čas komunikacije manjša z večanjem števila blokov in manjšanjem števila procesorjev, ima metoda manjšo komunikacijsko zahtevnost.

Kadar uporabimo metodo *cut-off*, lahko postane računanje sil neuravnoteženo. Različna obremenjenost procesorjev privede do neizkoriščenega časa čakanja, s čimer se zmanjša vzporedna učinkovitost računanja. Za novo metodo delitev sil s porazdelitvijo diagonale sem razvil način za dinamično uravnoteženje računanja. Ob upoštevanju trenutnih računskih obremenitev procesorjev diagonalo ponovno porazdelimo tako, da izenačimo računsko obremenitev procesorjev.

Pri vzporednih računalnikih se vse bolj uveljavljajo tisti s porazdeljenim pomnilnikom, pri čemer prednjačijo gruče osebnih računalnikov. Ena glavnih težav pri vzporednem računanju je komunikacija med procesorji, ker čas komunikacije zmanjšuje učinkovitost vzporednega računanja. Čas komunikacije je načeloma daljši pri računalnikih s porazdeljenim pomnilnikom kot pri tistih s skupnim pomnilnikom. Topologija povezav med procesorji vpliva na učinkovitost vzporednih algoritmov, ki se izvajajo na vzporednem računalniku. Uskladitev komunikacijskih vzorcev vzporednih programov in topologij zmanjša čas komunikacije med procesorji in poveča učinkovitost vzporednega računanja.

Zasnoval sem gručo osebnih računalnikov, katerih povezovalna topologija je prilagojena novo razviti vzporedni metodi delitev sil s porazdelitvijo diagonale. Gruča

uporabi več mrežnih stikal z manjšim številom priključkov. Za vsak blok $|B|$ uporabimo svoje stikalo, na katerega povežemo $|B|-1$ procesorjev, ki računajo interakcije z atomi tega bloka. Vsak procesor je povezan z dvema stikaloma. Ker procesorji izmenjujejo podatke le z drugimi procesorji, s katerimi imajo skupen blok, tak način povezave poveča prepustnost povezovalne topologije gruče.

Namenski procesorji so procesorji, ki so zasnovani za hitro izvajanje omejenih vrst računov. Procesor MDGRAPE-II je narejen za hitro računanje interakcij med atomi in služi za pohitritev računanja simulacij molekulske dinamike. V program za simulacijo molekulske dinamike sem dodal možnost vzporedne uporabe več takih procesorjev z metodo delitev sil, kar dodatno pohitri računanje simulacije molekulske dinamike.

S simulacijami molekulske dinamike in drugimi računskimi modeli realnih molekulskih sistemov prikažem uporabo in pomen vzporednih metod za izvajanje simulacij molekulske dinamike.

Abstract

Parallel computer programs are often used to speed up computationally intensive scientific tasks, such as computer simulations. Simulations enable theoretical research in molecular chemistry, structural biology, and other fields. Molecular dynamics simulation is the computational problem of solving a second-order system of differential equations that describe the motion of atoms of the simulated system. The problem must be solved numerically, since it generally can not be solved analytically; as such, the simulation is a sequence of consecutive time steps. The equations of motion must be solved for every atom. The basic output of a molecular dynamics simulation is the trajectory of atomic motions.

Molecular dynamics simulation methods are computationally demanding, which limits both the length of simulations and the size of the simulated molecular system. The most computationally demanding aspect is the calculation of the interactions. There are N^2 interactions in a system of N atoms. The interactions, or forces acting on every atom, much be calculated in each integration time step. There are several methods to decrease the interaction calculation computational requirements. Among these is the use of a cut-off distance: an interaction is disregarded if the distance between its two atoms is greater than the cut-off distance; the forces for such interactions need not be calculated.

Parallel molecular dynamics simulation methods distribute the interactions calculations among $|P|$ processors of the parallel computer. There are three types of parallel methods, differing in their distribution approach: replicated data, force decomposition, and space decomposition. In each of these methods, the processors must exchange data at every simulation time step. In the *replicated data* method, the latest atomic coordinates must be replicated at each processor. The $N/|P|$ atoms are evenly distributed among the processors, which perform the integration of the assigned atoms. The processors calculate a $1/|P|$ fraction of the force calculations. At every timestep, the processors must exchange data for all N atoms. In the *force decomposition method*, the processors have the data for only $2N/\sqrt{|P|}$ of the atoms and every processor communicates only with $2\sqrt{|P|}$ other processors to ex-

change the data of these atoms. The more processors there are, the less data they must exchange. In the *space decomposition* method, the simulation space is divided into cells, which are assigned to processors along with cells' corresponding atoms. The processors must exchange data with at least their 26 neighbors to successfully calculate atomic interactions. If there is no cut-off distance employed, then the communication requirements equal the replicated data approach.

I have developed a new parallel method for molecular dynamics simulation, the *distributed-diagonal force decomposition* method. The interactions of all atoms for a $N \times N$ force matrix. Distributing the atoms into $|B|$ blocks gives $|B|^2$ block products, which encompass the interactions among the atoms of the two blocks. The products are assigned to processors, which calculate interactions of the product. These block products divide the force matrix into a $|B| \times |B|$ grid. The half of the forces in the upper triangle of the force matrix can be disregarded, since they form mirrored opposites of the forces in the lower triangle. The forces near the diagonal correspond to forces among atoms of the same block. Since there are other processors that have all of the necessary data to calculate these forces, their calculation is distributed to these other processors. The distributed-diagonal force decomposition method therefore uses fewer processors for a given number of blocks. Since the communication time decreases with an increasing number of blocks, the method has a lower communication requirement.

The force calculation may become imbalanced when a cut-off distance is employed. Differing processor loads leads to processors idling as they wait for busier processors, which decreases parallel efficiency. I have developed a dynamic load balancing method for the now distributed-diagonal force decomposition method. By taking into account current processor loads, it re-distributes the force calculations from the diagonal among all of the processors to achieve balanced processor computational loads.

Distributed-memory parallel computers are becoming more prevalent for computer simulations. Most prominent of these are clusters of personal computers. One of the challenges facing parallel computation is the inter-processor communication requirement; distributed-memory machines must use message passing to exchange data, which is generally slower than in shared-memory machines. The processor interconnect topology affects the programs' parallel efficiency. Coupling the parallel programs' communication patterns with the interconnect topology achieves increased parallel efficiency.

I have designed a personal computer cluster whose interconnect is tailored to the new distributed-diagonal force decomposition method for molecular dynamics simulation. A number of smaller network switches are used, each one being assigned

to one of the $|B|$ blocks. The $|B| - 1$ processors that calculated interactions with a blocks are connected to the appropriate switch. Every processor is thus connected to two switches. Since a processor communicate only with the processors sharing the same blocks, this topology enables a higher throughput than a single standard network switch would.

Specialized processors are processors designed for the fast calculation of a limited set of calculations. The MDGRAPE-II processors is designed for calculating the inter-atomic interactions and serves to increase the speed of molecular dynamics simulations. I have enabled a molecular dynamics simulation program to use multiple MDGRAPE-II processors with the force decomposition method, which further speeds up the molecular dynamics simulation.

I have used parallel molecular dynamics simulations and other modeling methods to study real problems, showing the use and importance of parallel methods in computer simulations.

Kazalo

Poglavje 1

Uvod

V uvodnem delu disertacije bom predstavili pojme in metode, ki so ključni za razumevanje doktorskega dela. Predstavil bom vzporedne računalnike, različne vrste vzporednih računalnikov in njihove značilnosti. Nato bom opisal molekulske simulacije, pomen simulacij molekulske dinamike in še posebno izvedbo računanja takih simulacij. Oba pojma združim v predstavitvi vzporednih računalniških simulacij, kjer razjasnim pomen vzporednega računanja za reševanje znanstvenih problemov, opišem tehnike paralelizacije in predstavim delitev podatkov in računanja na vzporednih procesorjih.

1.1 Vzporedni računalniki

Namen uporabe računalnikov je reševanje računskih problemov. Vedno hitrejši računalniki nam omogočajo vse hitrejše računanje [9,74], vendar je pri marsikaterem problemu dobrodošla vsaka dodatna pohitritev. Mnogo takih računskih problemov lahko razstavimo na podprobleme, ki jih rešujemo vzporedno na ločenih procesorjih vzporednega računalnika [32].

1.1.1 Vrste osebnih računalnikov

Vzporedne računalnike delimo na dve vrsti glede na dostopnost pomnilnika vseh procesorjev. Vrsta dostopa do pomnilnika določa, na kakšen način si procesorji lahko izmenjujejo podatke. Od načina izmenjave podatkov zavisi, kako mora biti pisan vzporeden program za računanje vzporednega problema.

Računalniki s skupnim pomnilnikom (ang. shared-memory) omogočajo vsem procesorjem, da dostopajo neposrednega do vseh delov pomnilnika [1]. Dostopni čas je lahko enak za vse procesorje, kot je v primeru simetrično multiprocesorskih

računalnikov (ang. Symmetric Multi-Processing, SMP), ali pa se razlikuje pri procesorjih NUMA (ang. Non-Uniform Memory Access) [1]. Pri računalnikih vrste NUMA ima vsak procesor lokalni pomnilnik, do katerega ima najhitrejši dostop. Do ostalih delov pomnilnika, ki mu niso lokalni, tudi ima dostop, vendar je počasnejši [19]. Program, ki je pisan za računalnik s skupnim pomnilnikom, lahko upošteva možnost neposrednega dostopa do poljubnih podatkov programa na poljubnem drugem procesorju, ki tudi služi izvajanju vzporednega programa.

Na vzporednih računalnikih s porazdeljenim pomnilnikom (ang. distributed-memory) procesorji ne morejo neposredno dostopati do vseh delov pomnilnika [1]. Procesorji morajo imeti način komunikacije, ki služi prenosu podatkov. Program, ki je pisan za računalnike s porazdeljenim pomnilnikom, mora za izmenjavo podatkov med procesorji uporabiti prenos sporočil (ang. message-passing). Vsak manjkajoči podatek, ki ga potrebuje program na enem procesorju, mora preko sporočila dobiti od drugega procesorja, ki mu ga pošlje. Obstaja več računalniških knjižic, ki abstrahirajo komunikacijsko opremo vzporednega računalnika s porazdeljenim pomnilnikom, kar omogoča enostavnejšo pisanje vzporednega programa in lažji prenos med računalniki. Takšne knjižnice so npr. PVM [67] (Parallel Virtual Machine, navidezni vzporeden računalnik) ter LAM-MPI [18] in MPICH [30], ki temeljita na standardu MPI (vmesnik za prenašanje sporočil, Message Passing Interface).

1.1.2 Gruče osebnih računalnikov

Ko se je zmogljivost osebnih računalnikov povečala na nivo delovnih postaj in je prišla na tržišče dovolj zmogljiva mrežna oprema za povezovanje osebnih računalnikov z mrežo, so začeli v računskih centrih povezovati osebne računalnike v gruče (ang. clusters) [65,66]. Za prve gruče so uporabili tehnologijo Ethernet s hitrostjo 100 Mb/s za mrežno povezavo med računalniki, kasneje pa so poleg hitrejše tehnologije Ethernet 1 Gb/s pričeli uporabljati tudi alternativne tehnologije, kot so Myrinet, Infiniband idr. [7, 48], ki obljubljajo višjo hitrost, predvsem pa manjšo zakasnitev prenosov.

Gruča osebnih računalnikov je vzporeden računalnik s porazdeljenim pomnilnikom. Vsak sestavni računalnik ima svoj pomnilnik, do katerega lahko neposredno dostopajo le lokalni procesorji v računalniku. Zato mora vzporeden program za prenos podatkov pri računanju uporabiti prenos sporočil po mreži, za kar ponavadi uporabljajo obstoječe knjižnice, npr. MPI.

Večprocesorski računalniki

Osebni računalniki imajo manjše število procesorjev v primerjavi z močnejšimi superračunalniki [20, 31, 65]; v preteklosti so bili računalniki ponavadi le eno- ali dvoprocesorski. V zadnjem času se tudi v osebnih računalnikih pojavljajo večjedrni procesorji (ang. multi-core processors), na katerih vsako jedro predstavlja suveren računski procesor; v času pisanja je tako mogoče uporabiti računalnike z 8 računskimi procesorji kot gradniki gruče [33].

Gruče, sestavljene iz računalnikov z večimi procesorji, združujejo lastnosti računalnikov s skupnim pomnilnikom in računalnikov s porazdeljenim pomnilnikom. Program sicer lahko izkoristi skupni pomnilnik vsakega sestavnega računalnika za pohitritev nekaterih prenosov, vendar ima gruča kot celota še vedno porazdeljen pomnilnik.

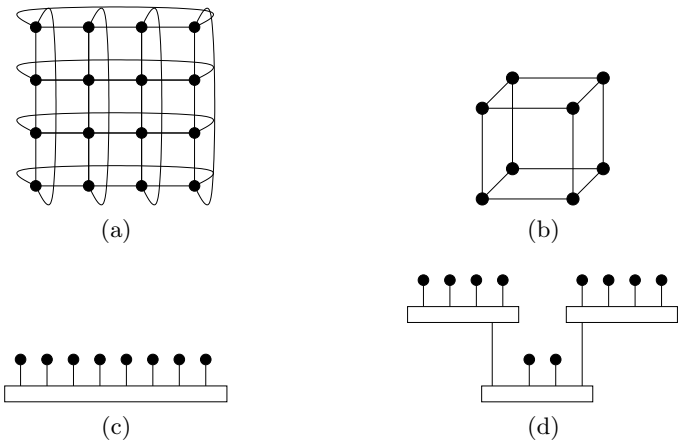
Topologije gruč

Topologija gruče opisuje, kako so sestavni procesorji in računalniki gruče med sabo povezani. Pri prvih gručah so uporabili mrežno stikalo [65, 66], preko katerega so vsi računalniki povezani s topologijo polnega grafa – vsak računalnik lahko neposredno pošlje sporočilo poljubnemu drugemu računalniku, ki je tudi priključen na isto stikalo. A že takrat so iskali načine za pohitritev povezav, npr. z združevanjem mrežnih povezav [66]. Pri nekaterih gručah so načrtovalci uporabili točkovne povezave, ki tvorijo topologijo z manjšo povezljivostjo od polnega grafa (torej računalnik ne more neposredno poslati sporočila poljubnemu drugemu računalniku, le takim, s katerimi je neposredno povezan) [61]. Če domena vzporednega problema dopušča tako topologijo, je lahko prepustnost take mreže s točkovnimi povezavami večja, kot če bi uporabili le skupno mrežno stikalo.

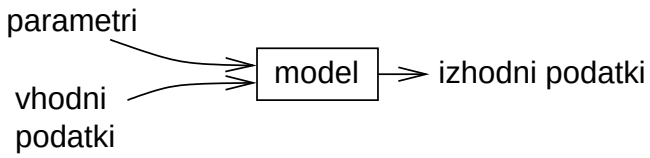
Primeri topologij, ki so predstavljene na sliki 1.1, so 2-D mreže [70], hiperkocka [8], mrežno stikalo, in tudi večnivojsko povezovanje mrežnih stikal [22].

1.2 Računalniške simulacije

Računalniške simulacije uporabljamo za teoretske raziskave na različnih znanstvenih področjih. Kot virtualni poskusi so računalniške simulacije lahko komplementarne ali nadomestne pravim eksperimentom. Pravi eksperiment je lahko neizvedljiv zaradi fizikalnih [38], tehničnih [2] ali finančnih [41] omejitev; v takih primerih se poslužujemo računalniških simulacij, ki nimajo takih omejitev. Z računalniškimi simulacijami lahko razjasnimo delovanje realnega sistema.



Slika 1.1: Prikaz različnih topologij povezav med procesorji. Na podslikah so prikazane topologije; procesorji oziroma računalniki so prikazani s krogi, stikala s kvadrati, povezave pa s črtami. Na podsliki a je 2-D mreža, na podsliki b je kocka, na podsliki c je mrežno stikalo (ki nudi polno povezljivost), na podsliki d pa so mrežna stikala v kaskadi.

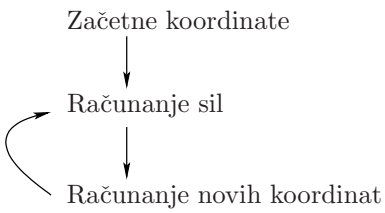


Slika 1.2: Ponazoritev računalniške simulacije. Z modelom in parametri modela opišemo realni sistem tako, da ga lahko z računalnikom simuliramo. Za računanje simulacije moramo podati še začetno stanje kot vhodne podatke.

Zakonitosti realnega sistema, ki ga simulirimo, vključimo v model. S parametri modela mu spreminjamo lastnosti, tako da se ujema z lastnostmi modeliranega sistema. Za izvedbo simulacije moramo podati tudi sistem, ki ga simuliramo in njegovo začetno stanje. Ponazoritev simulacije je prikazana na sliki 1.2.

1.2.1 Simulacija molekulske dinamike

Simulacije molekulske dinamike (MD) omogočajo teoretske raziskave v molekularni kemiji, strukturni biologiji in drugih področjih znanosti. Simulacija MD v splošnem predstavlja reševanje problema n delcev, zato so metode, razvite zanjo, prenosljive tudi na druge vrste simulacij [55]. Ker ta problem v splošnem ni analitično rešljiv, ga obravnavamo numerično. Rešiti moramo sistem gibalnih enačb za vsak atom posebej [29].



Slika 1.3: Glavna zanka računanja simulacije MD. Vsak korak simulacije sestoji iz dveh delov: računanja sil in integracije oziroma računanje novih koordinat. Za računanje novih koordinat moramo imeti izračunane sile. Za izračun sil na naslednjem koraku moramo imeti koordinate, izračunane v prejšnjem koraku.

Računanje simulacije MD je numerično reševanje diferencialnih gibalnih enačb 2. reda,

$$\begin{aligned} m_i \frac{d\vec{v}_i}{dt} &= \sum_j \vec{f}_{i,j} \\ \frac{d\vec{r}_i}{dt} &= \vec{v}_i, \end{aligned} \tag{1.1}$$

kjer je m_i masa atoma i, $\vec{v}_i$ njegova hitrost, $\vec{f}_{i,j}$ sila atoma j na atom i, $\vec{r}_i$ pa so koordinate atoma i.

Več različnih integracijskih metod lahko uporabimo za integracijo gibalnih enačb [29, 35, 36, 47]; nekatere celo analitično obravnavajo gibanja najhitrejših atomov [17, 37, 39, 40, 59, 60].

Z računskega vidika poteka računanje simulacije MD kot zaporedje časovnih korakov (ang. time steps), sestavljenih iz dveh delov,

1. računanje energije oziroma sil

2. integracija gibalnih enačb oziroma računanje novih koordinat atomov

pri čemer moramo podati začetne koordinate atomov. Postopek je shematsko prikazan na sliki 1.3.

Polje sil

S poljem sil (ang. force-field) modeliramo vse interakcije med atomi, služi pa računanju potenciala (energije). Opišemo ga z veččlensko enačbo. Mnogokrat se uporablja polje sil s členi, ki opisujejo prispevek veznih interakcij vezi, kotov in dihedralnih kotov med atomi k celotnemu potencialu, prav tako pa tudi potencial

neveznih interakcij zaradi elektrostatike in van der Waalsovih interakcij [47]:

$$V(\vec{r}^N) = \sum_{\text{vezi}} \frac{k_i}{2}(l_i - l_{i,0})^2 + \sum_{\text{koti}} \frac{k_i}{2}(\theta_i - \theta_{i,0})^2 + \sum_{torzije} \frac{V_n}{2}(1 + \cos(n\omega - \gamma)) + \sum_{i=1}^{N} \sum_{j=i+1}^{N} \left(4\epsilon_{ij} \left[\left(\frac{\sigma_{ij}}{r_{ij}} \right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}} \right)^{6} \right] + \frac{q_i q_j}{4\pi\epsilon_0 r_{ij}} \right), \quad (1.2)$$

kjer je $V(\vec{r}^N)$ celotni potencial v odvisnosti od $\vec{r}^N$, vektorja koordinat vseh atomov. Prvi člen opisuje prispevek vezanih atomskih parov k celotnemu potencialu in je modeliran s harmonskih potencialom, podobno kot je vzmet (slika 1.4a): k_i so konstante vezi, l_i so dolžine vezi, $l_{i,0}$ pa njihove ravnovesne dolžine. Drugi člen opisuje prispevek kotov med vsemi trojicami vezanih atomov k celotnemu potencialu; tudi ta kotni potencial je modeliran s harmonskih potencialom (slika 1.4b); k_i so konstante kotov, θ_i so širine kotov, $\theta_{i,0}$ pa so ravnovesne širine kotov. Tretji člen opisuje prispevek torzijskih (dihedralnih) kotov med štirimi atomi (slika 1.4c) k celotnemu potencialu glede na vrtenje kota; V_n so konstante dihedralnih kotov, n so multitude, ω so vrednosti dihedralnega kota, γ pa ravnovesne vrednosti. Zadnji, četrti člen, opisuje prispevek van der Waalsovih in elektrostatskih interakcij k potencialu. Lennard-Jonesovi potenciali Van der Waalsovih interakcij opisujejo nevezne interakcije med bližnjimi atomi; potencial take interakcije je prikazan na sliki 1.5a. Elektrostatske interakcije nastanejo zaradi odboja enako nabitih atomov ali privlaka nasprotno nabitih atomov; tak potencial je prikazan na sliki 1.5b.

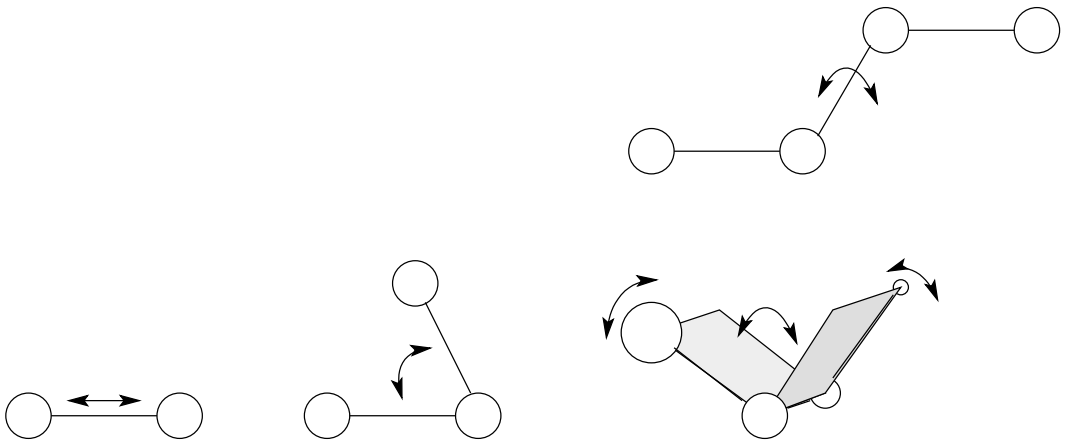
Iz potenciala 1.2 lahko izračunamo silo $\vec{f_i}$, ki deluje na atom i, s parcialnim odvodom po koordinatah atoma i,

$$\vec{f_i} = \frac{\partial V(\vec{r}^N)}{\vec{r_i}}. \quad (1.3)$$

Z vidika računanja je delitev na vezne in nevezne interakcije pomembna. Na sliki 1.6 so prikazane vezne interakcije (polne modre črte) in nevezne interakcije (črtkane zelene črte) v sistemu dveh molekul vod, kar je skupaj 6 atomov. Za pare atomov, ki so vezani ali obravnavani z veznimi interakcijami (sosedje–vez, njihovi sosedje–kot, in sosedje le-teh–dihedralni kot), ne računamo neveznih interakcij, saj je vpliv njihovih neveznih interakcij zajet v opisu veznih interakcij.

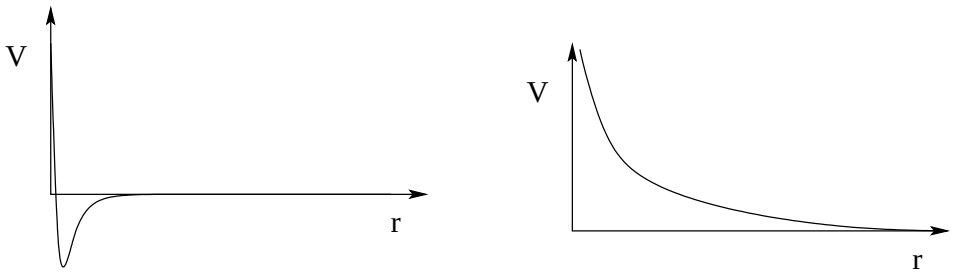
1.2.2 Zmanjšanje računske zahtevnosti

Že na tako majhnem sistemu, kot je prikazan na sliki 1.6, je razvidno, da je neveznih interakcij mnogo več od veznih. V sistemu z N atomi je veznih interakcij $\mathcal{O}(N)$, saj



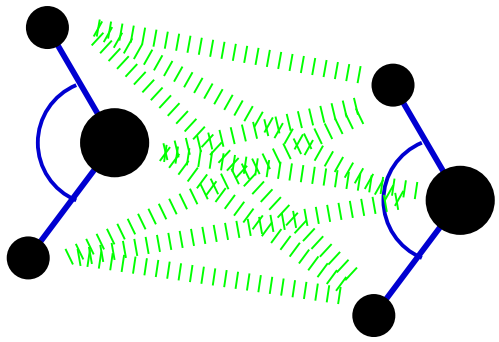
(a) Vez (b) Kot (c) Dihedralni kot

Slika 1.4: Vezne interakcije. Sprememba dolžine vezi (podslika a) spremeni potencial med dvema atomoma. Sprememba kota (podslika b) spremeni potencial med krajnima atomoma. Dihedralni kot med štirimi atomi (podslika c) je kot med ravninama, ki ju določajo prvi trije in zadnji trije atomi. Sprememba dihedralnega kota spremeni potencial med dvema krajnima atomoma.



(a) Lennard-Jonesov potencial (b) Elektrostatski potencial

Slika 1.5: Potenciala van der Waalsovih (podslika a) in elektrostatskih (podslika b) neveznih interakcij. Grafa prikazujeta potencial V v odvisnosti od razdalje r med dvema atomoma.



Slika 1.6: Ponazoritev veznih in neveznih interakcij v sistemu dveh molekul vode (velika črna kroga sta kisika (O), mali črni krogi pa vodike (H). Modre ravne črte predstavljajo vezi med pari kisikov in vodikov (vez O–H). Modra ukrivljena črta predstavlja kot med tremi atomi (H–O–H). Obe vrsti interakcij sta vezni. Zelene črtane črte predstavljajo nevezne interakcije med atomi, ki medsebojno nimajo veznih interakcij. Med dvema molekulama vode je 9 veznih interakcij – vsak izmed atomov ene molekule vode ima nevezne interakcije z vsemi tremi atomi druge molekule vode.

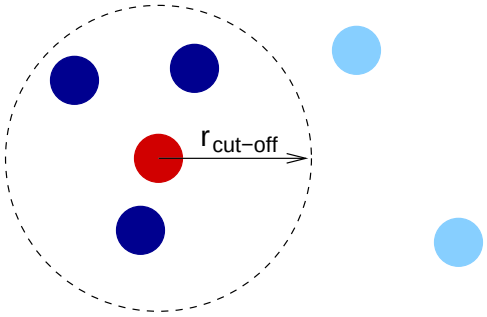
ima vsak atom lahko le nekaj vezi, ponavadi do 3; neveznih interakcij pa je $\mathcal{O}(N^2)$, saj imata oba atoma vsakega izmed $N \times N$ parov atomov interakcijo med sabo (poleg prej omenjenih $\mathcal{O}(N)$ veznih parov). Izračun tako velikega števila neveznih interakcij je med večjimi omejitvami računalniških simulacij. Omejujejo dolžino simulacije, še bolj pa velikost obravnavanih sistemov.

Nekaj metod je bilo razvitih za pohitritev izračuna neveznih interakcij in omogočanja simulacij večjih sistemov. Med te sodi osnovna, uporaba razdalje *cut-off*, prav tako pa drevesna metoda [3] in metoda multipolov [5, 6].

Razdalja *cut-off*

Uporaba razdalje *cut-off* je med najbolj osnovnimi tehnikami za zmanjšanje računske zahtevnost neveznih interakcij [47]. Značilnost neveznih interakcij je, da se njihove vrednosti manjšajo z večanjem razdalje; tako se obnašata tako klasični potencial Lennard-Jones, ki opisuje van der Waalsove interakcije, kot tudi elektrostatski potencial; prvi pada v odvisnosti $1/r^6$ z razdaljo r, drugi pa v odvisnosti $1/r$. Potencial z večanjem razdalje teži proti vrednosti 0. Zato se lahko odločimo, da je razlika zanemarljiva nad neko razdaljo r_{cutoff}; Razdaljo r_{cutoff} imenujemo *cut-off*. Majhna nezveznost, ki se pojavi v potencialu pri prehodu meje r_{cutoff}, ima vpliv na simulacijo, ki ga lahko odpravimo z različnimi metodami [27, 49].

Na sliki 1.7 je ponazorjena uporaba metode z razdaljo *cut-off*. Interakcije upoštevamo le za atome, ki so znotraj radija r_{cutoff}. Ker privzamemo, da je poten-



Slika 1.7: Ponazoritev uvedbe razdalje *cut-off* za upoštevanje in računanje različno oddaljenih interakcij od rdečega atoma. Interakcije s temno modrimi atomi, ki so bližji rdečemu od r_{cutoff}, upoštevamo. Interakcij s svetlo modrimi atomi, katerih razdalja od rdečega atoma je večja od r_{cutoff}, ne upoštevamo.

cial med dvema atomoma nad neko razdaljo enak 0, se na ta način lahko izognemo računanju vseh interakcij med atomi, ki so oddaljeni več kot je razdalja *cut-off*. Če gledamo rdeči atom na sliki 1.7, to pomeni, da računamo interakcije le med med njim in temno modrimi atomi. Svetlo modri atomi so preveč oddaljeni, njihovih interakcij z rdečim ne upoštevamo, zato jih tudi ne računamo.

Primernost uporabe razdalje *cut-off* za zmanjšanje računske zahtevnosti računanja neveznih interakcij se pokaže zlasti pri večjih sistemih. Če imamo molekulski sistem z N atomi z atomsko gostoto ρ (število atomov na enoto prostora), ni potrebno računati vseh N^2 interakcij (z $N^2/2$ računi sil), temveč le

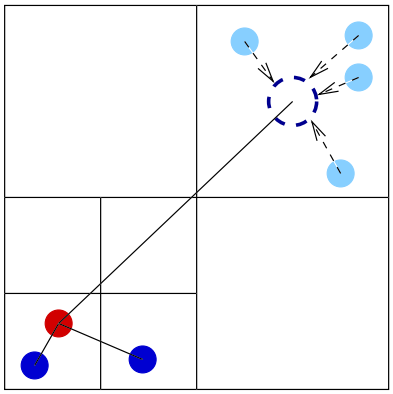
$$n_{\text{calc}} = N \times \frac{4}{3}\pi r_{\text{cutoff}}^3 \, \rho \tag{1.4}$$

interakcij z $n_{\text{calc}}/2$ računi sil.

Drevesna metoda

Drevesna metoda po Barnes in Hutu [3,4] temelji na spoznanju, da lahko interakcije med nekim izbranim atomom in skupino oddaljenih atomov modeliramo kot interakcijo med izbranim atomom in nekim navideznim delcem, ki vsebuje lastnosti celotne skupine oddaljenih atomov. Princip delovanja metode je prikazan na sliki 1.8.

Pri tej metodi prostor hierarhično razdelimo v drevesno strukturo, dokler niso razdelki prostora dovolj majhni, da vsebujejo le nekaj atomov. Za skupino atomov v takem prostorskem razdelku izračunamo lastnosti predstavnika podskupine, npr. težišče povprečnega naboja atomov. Nato za vsak atom a izmed N atomov izvedemo iskanje v globino po zgrajenem drevesu. Če je obiskani razdelek dovolj blizu atomu a, potem izračunamo neposredne interakcije med atomi v razdelku in



Slika 1.8: Drevesna metoda po Barnes in Hutu. Interakcije delimo glede na razdaljo. Interakcije med dvema atomoma, ki sta dovolj blizu, npr. modra atoma do rdečega, računamo neposredno. Pri oddaljenih atomih računamo interakcije posredno. Štirim svetlo modrim atomom, ki so dovolj oddaljeni od rdečega, priredimo navidezen delec, prikazan kot moder črtast krog, ki zajema povprečje štirih atomov. Namesto računanja interakcij posebej za vsakega izmed 4 atomov, izračunamo le eno, med rdečim atomom in navideznim delcem.

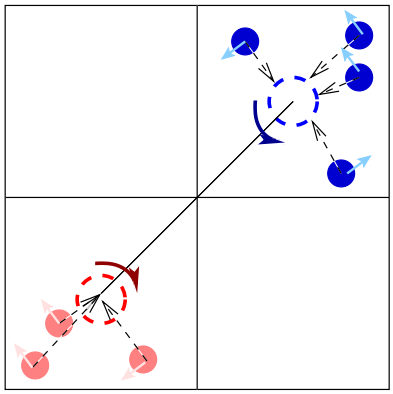
atomom a. Če je razdelek dovolj oddaljen, potem izračunamo le interakcije med atomom predstavnikom razdelka in atomom a.

Časovna zahtevnost drevesne metode za računanje neveznih interakcij je $\mathcal{O}(N \log N)$. Časovna zahtevnost preiskovanja drevesa v globino je $\mathcal{O}(\log N)$, kar moramo storiti za vsak atom, torej N-krat.

Metoda hitrih multipolov

Metoda hitrih multipolov temelji na podobnih načelih kot drevesna metoda po Barnes in Hutu, torej da lahko več interakcij oddaljenih delcev nadomestimo s približkom ene same interakcije [75]. Namesto povprečenja večih interakcij med enim delcem in skupino oddaljenih delcev, povprečimo interakcije med dvema oddaljenima skupinama. Ne uporabimo povprečja prvega reda (npr. povprečna sila) kot približek atomov v prostorskem razdelku, ampak uporabimo povprečje višjih redov (npr. dipol, kvadropol) oziroma multipole. Primer interakcije dveh oddaljenih skupin atomov s približkom je prikazan na sliki 1.9. Vpliv take interakcije dveh skupin se posledično prenese na vse atome dveh skupin. Razdelitev prostora izvedemo na podoben način kot pri drevesni metodi.

Časovna zahtevnost metode hitrih multipolov je le $\mathcal{O}(N)$ [55, 75].



Slika 1.9: Metoda hitrih multipolov. Interakcije med skupinami oddaljenih atomov izračunamo preko ene interakcije njihovih multipolov. Rdeči črtani krog predstavlja multipol rdečih atomov, modri črtani krog pa multipol modrih atomov. Na primer, da je interakcija med multipoloma torzija; prikazana je s temno rdečo oziroma temno modro puščico na ustrezni multipol. Torzija se preko multipola odraža kot sila na atome, ki so sestavni del multipola; te sile na posamezne atome so prikazane s svetlo rdečo oziroma svetlo modro barvo.

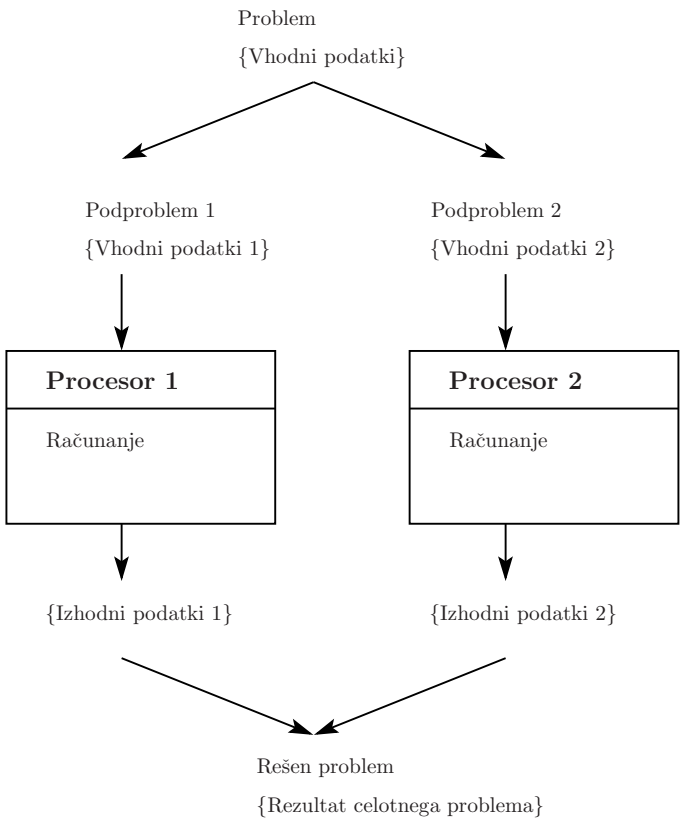
1.3 Vzporedne računalniške simulacije

Pri mnogih vrstah računalniških simulacij lahko računanje simulacij porazdelimo, tako da podprobleme simulacij računamo sočasno na ločenih, vzporednih procesorjih. Pri takih vzporednih računalniških simulacijah moramo določiti, kako porazdeliti podatke in računanje po procesorjih. Vrsta porazdelitve podatkov in računanja zavisi od izbranega vzporednega algoritma za računanje simulacije na vzporednem računalniku.

1.3.1 Delitev podatkov po procesorjih in izmenjava podatkov med procesorji

Vzporeden algoritem določa, katere dele celotnega računa računajo posamezni vzporedni procesorji. Glede na porazdelitev delov računa morajo biti porazdeljeni podatki, ki jih deli računov potrebujejo kot vhodne podatke (ang. domain decomposition) [32]. Poleg tega nam vzporeden algoritem določa, katere podatke morajo procesorji izmenjavati.

Pri najbolj preprostih vzporednih algoritmih lahko celoten račun razdelimo na neodvisne podračune [46]. Vsakemu procesorju dodelimo samostojen račun in začetne podatke zanj. Procesorji računajo neodvisno, po končanih izračunih pa združimo izhodne podatke. Ponazoritev neodvisnih vzporednih računov na dveh procesorjih je na sliki 1.10. Računanja mnogih vrst računalniških simulacij ne mo-



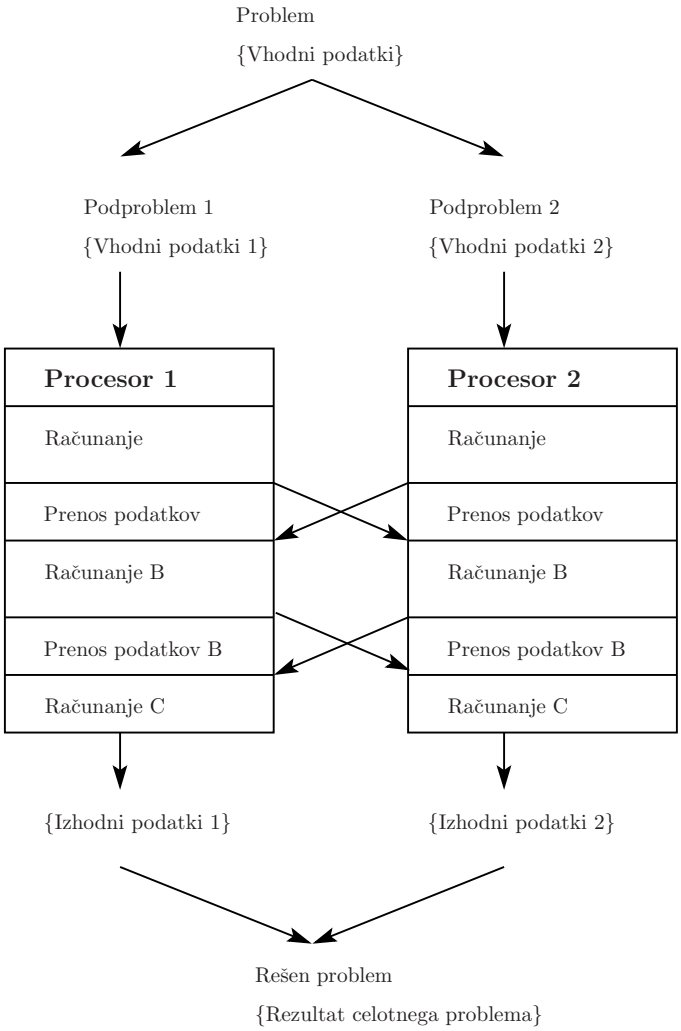
Slika 1.10: Paralelizacija dveh neodvisnih podproblemov enega problema. Procesorja računata neodvisno in edina izmenjava podatkov je začetni prenos vhodnih podatkov in združevanje izračunanih izhodnih podatkov.

remo tako paralelizirati [16, 50], lahko pa več neodvisnih simulacij računamo vzporedno.

Računanja mnogih problemov, kot so tudi simulacije MD, ne moremo razdeliti na neodvisne podračune [32, 50]. Računanje sicer porazdelimo na vzporedne procesorje, vendar račun na enem procesorju potrebuje delno izračunane podatke, ki so prisotni na nekem drugem procesorju. Procesorji morajo zato izmenjevati podatke med posamezni deli računov. Ponazoritev tako odvisnih vzporednih računov, kjer procesorji izmenjajo podatke med posameznimi deli računov, je na sliki 1.11.

1.3.2 Učinkovitost vzporednega računanja

Cilj vzporednega računanja je zmanjšanje časa, ki je potreben za računanje problema, npr. simulacije. Učinkovitost vzporednega računanja merimo na dva načina, s *pohitritvijo računanja* in z *vzporedno učinkovitostjo*. Za izračun teh dveh metrik moramo poznati T, t.j. čas računanja problema na enem procesorju, $|P|$ oziroma



Slika 1.11: Paralelizacija dveh odvisnih podproblemov enega problem. Procesorja računata vsak svoj del podproblema, vendar morata zadnja dva dela računa dobiti podatke prejšnjega dela računa iz drugega procesorja. Procesor 1 mora za računanja računskega dela Računanje B dobiti izračunane podatke računskega dela Računanje od procesorja 1; procesor 2 pa obratno. Podobno mora procesor 1 dobiti podatke po končanem Računanje B od procesorja 2, da lahko izvede Računanje C, obratno pa velja za procesor 2.

število vzporednih procesorjev iz množice procesorjev P ter T_p, čas vzporednega računanja problema.

Pohitritev računanja

Pohitritev računanja S izraža, kolikokrat hitreje poteka vzporedno računanje v primerjavi z računanjem na enem procesorju,

$$S = \frac{T}{T_p} \tag{1.5}$$

Zaželena je čim večja pohitritev, ki je v idealnih pogojih enaka številu procesorjev. Pri nekaterih problemih je pohitritev lahko celo večja od števila procesorjev, pri t.i. nadlinearni pohitritvi (ang. super-linear speedup); nadlinearna pohitritev nastane le zaradi strojne opreme, npr. zaradi vpliva predpomnilnika, saj procesor hitreje računa manj obsežno domeno podatkov podproblema kot obsežnejšo domeno podatkov celotnega problema [72].

Vzporedna učinkovitost

Vzporedna učinkovitost E izraža pohitritev v odvisnosti od števila procesorjev,

$$E = \frac{S}{|P|} = \frac{T}{|P|T_p}. \tag{1.6}$$

Zaželena je učinkovitost $E = 1.0$ oziroma 100%, ponavadi je pa manjša od 100% in pada z večanjem števila procesorjev.

Pri popolnoma vzporednem problemu, kjer so podproblemi povsem neodvisni, kot je ponazorjeno na sliki 1.10 (str. 22), lahko računanje pohitrimo sorazmerno s številom procesorjev. Čas računanja vsakega izmed $|P|$ neodvisnih podproblemov je T_p,

$$T_p = \frac{T}{|P|}. \tag{1.7}$$

Ker računi potekajo vzporedno, je razvidno, da je pohitritev enaka številu procesorjev $|P|$, učinkovitost pa je enaka 1 oz. 100%.

Nekaterih delov problema ne moremo računati vzporedno. Čas T razdelimo na dva dela: $T_{\text{vzporedni}}$ oziroma čas računanja dela problema, ki ga lahko računamo tudi vzporedno; $T_{\text{zaporedni}}$ oziroma čas računanja dela problema, ki ga ne moremo računati vzporedno. Tedaj je čas računanja na vsakem procesorju enak

$$T_p = \frac{T_{\text{vzporedni}}}{|P|} + T_{\text{zaporedni}}, \tag{1.8}$$

pohitritev je

$$S = \frac{T}{\frac{T_{\text{vzporedni}}}{|P|} + T_{\text{zaporedni}}}.$$

Z večanjem števila procesorjev se stopnja rasti pohitritve vse bolj manšja; limita pohitritve je

$$T/T_{\text{zaporedni}}, \tag{1.9}$$

ki je znan kot Amdahlov zakon [42], ki določa največjo možno pohitritev problema, ki ga ne moremo v celoti računati vzporedno.

Kadar podproblemi niso neodvisni, npr. pri vzporedni simulaciji MD, morajo procesorji, ki vzporedno računajo podprobleme, izmenjevati podatke. Čas komunikacije T_c, torej čas, ki je potreben za izmenjavo podatkov, pa podaljša čas računanja vsakega izmed podproblemov,

$$T_p = \frac{T}{|P|} + T_c, \tag{1.10}$$

kar je tudi čas vzporednega računanja celotnega problema. Komunikacijski čas, ki se spreminja s številom procesorjev – ponavadi se veča z večanjem števila procesorjev – tudi zmanjšuje vzporedno učinkovitost:

$$E = \frac{T}{|P|T_p} = \frac{T}{|P|(T/|P| + T_c)} = \frac{T}{T + |P|T_c}. \tag{1.11}$$

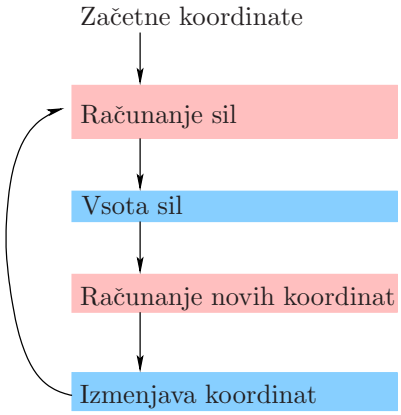
Razvidno je, da je zmanjševanje komunikacijskega časa en izmed glavnih ciljev pri načrtovanju vzporednih algoritmov.

1.4 Vzporedne metode za računanje simulacije MD

Pri vzporednem računanju simulacije MD procesorji vzporedno računajo dva dela vsakega koraka simulacije: vzporedno računajo sile in vzporedno računajo nove koordinate [71]. Ker sta ta dva koraka zaporedna (za računanje koordinat moramo poznati sile), koraki pa tudi zaporedni (za računanje sil moramo poznati koordinate s prejšnjega koraka), morajo procesorji dvakrat izmenjevati podatke v vsakem koraku simulacije. Postopek vzporedne simulacije MD je predstavljen na sliki 1.12.

1.4.1 Vrste vzporednih metod

Poznamo tri razrede vzporednih metod za računanje simulacije MD: replicirani podatki, prostorska delitev in delitev sil. Metode se razlikujejo po načinu, kako razdelijo računanja interakcij po procesorjih, kar obenem določa tudi prisotnost podatkov po



Slika 1.12: Glavna zanka vzporedna računanja simulacije MD. Rdeče označna dela koraka, t.j. Računanje sil in Računanje novih koordinat, računamo vzporedno. Modra označena dela koraka pa predstavlja komuniciranje, ki je potrebno za izmenjavo podatkov med procesorji, da ti lahko nadaljujejo računanje.

procesorjih.

Replicirani podatki

Najenostavnejša metoda paralelizacije je metoda repliciranih podatkov. Ime metode izvira iz značilnosti, da ima vsak procesor koordinate o vseh atomih, vsakemu izmed $|P|$ procesorjev pa dodelimo računanje $1/|P|$-tega dela interakcij (npr. $N^2/|P|$, če ne uporabimo razdalje *cut-off*) [16, 58].

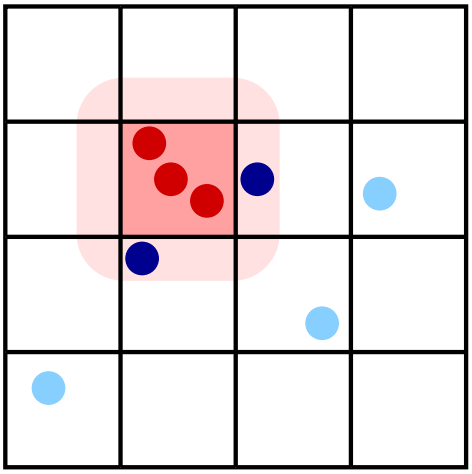
Po vzporednem računanju interakcij moramo združiti izračunane interakcije ter jih prenesti med procesorji tako, da procesorji poznajo sile za računaje novih koordinat. Po vzporednem računanju novih koordinat moramo koordinate prenesti na vse procesorje. Pri vseh prenosih prenašamo $\mathcal{O}(N)$ podatkov o vseh atomih. Podrobnejši opis in motivacija obeh prenosov sta zapisana v podpoglavju 1.4.3.

Edina posledica uvedbe razdalje *cut-off* pri metodi repliciranih podatkov je, da procesorjem dodelimo različno število interakcij, katere morajo računati. V tem primeru se pojavi neuravnoveteženost računanja, kar zmanjša vzporedno učinkovitost (več o uravnoteženosti računanja je zapisano v podpoglavju 3.1).

Prostorska delitev

Metoda prostorske delitve razdeli prostor simuliranega sistema na ločene dele, katerim dodelimo posamezne procesorje. Vsak procesor računa interakcije med atomi v njegovem delu prostora; zato ima vsak procesor podatke o atomih v njegovem delu.

Za računanje interakcij mora imeti vsak procesor koordinate svojih atomov



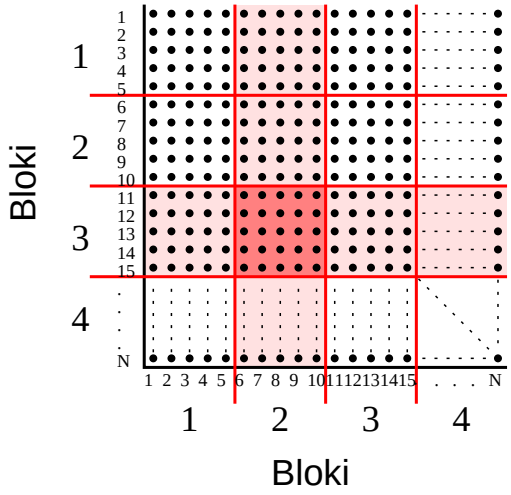
Slika 1.13: Prostorska delitev v 2 dimenzijah. Realni simulacijski prostor razdelimo na 16 delov. Vsakemu delu, kot npr. rdeče obarvanemu, priredimo procesor, ki ima koordinate vseh atomov v svojem delu (rdeči atomi). Obmejni pas sosednjih delov (svetlo rdeči) so široki toliko, kolikor je razdalja *cut-off* in vsebujejo atome (temno modri), ki jih mora »dobiti« procesor, da lahko izračuna interakcije vseh svojih atomov (treh rdečih).

in tudi tistih, ki so zunaj njegovega dela prostora in ki imajo interakcijo z atomi v njegovem delu prostora. Na sliki 1.13 je prikazana prostorska delitev 2-D prostora na 16 delov. Za vseh 8 sosedov so prikazani tudi pasovi, v katerih se nahajajo atomi, ki so znotraj razdalje *cut-off* od meje označenega dela. Atomi znotraj tega obmejnega pasu so dovolj blizu, da imajo lahko interakcijo z atomi znotraj označenega dela. Zato mora imeti procesor koordinate vseh takih bližnjih sosednjih atomov.

Pri metodi prostorske delitve lahko procesorji komunicirajo le po integraciji gibalnih enačb [5]. Vsak procesor pošlje vsakemu sosedu nove koordinate tistih atomov, ki so v njegovem obmejnem pasu. Če je razdalja *cut-off* dovolj majhna, izmenjujejo le $\mathcal{O}((N/|P|)^{2/3})$ podatkov [58]. Z večanjem razdalje *cut-off* se veča tudi obmejni pas, zato je za prostorsko delitev zaželeno, da je razdalja *cut-off* čim manjša. Če molekulski sistem nima enakomerne gostote, potem so število atomov v različnih delih prostora razlikuje, kar privede do neuravnoteženega računanja.

1.4.2 Delitev sil

Metoda delitev sil temelji na vzporednem računanju interakcij tako, da vsak procesor računa interakcije med dvema podmnožicama vseh atomov; disjunktno podmnožico atomov imenujemo blok, bloki pa tvorijo pokritje vseh atomov. Interakcije med N atomi lahko prikažemo v matriki sil. Če atome porazdelimo v $|B|$ blokov, potem



Slika 1.14: Delitev sil. Atome razdelimo v 4 bloke, vsakemu produktu blokov pa dodelimo procesor. Procesor, ki ga dodelimo rdeče obarvanem produktu blokov 2 in 3, računa interakcije le med atomi teh dveh blokov. Ta procesor komunicira le z ostalimi 6 procesorji (svetlo rdeči produkti), ki imajo skupen blok 2 ali 3.

matriko sil razdelimo na $|B|^2$ produktov blokov. Delitev vseh atomov v 4 bloke in nastalih 16 produktov med njimi prikazuje slika 1.14.

Produktom blokov dodelimo procesorje, ki računajo interakcije med atomi v dveh blokih. Procesorji komunicirajo le s tistimi procesorji, s katerimi imajo skupen blok in prenašajo $\mathcal{O}(N/\sqrt{|P|})$ podatkov. Komunikacija je neodvisna od razdalje *cut-off*, lahko pa pride do neenakomerne obremenitve procesorjev, podobno kot pri metodi repliciranih podatkov.

Več o delitvi sil je zapisano v podpoglavju 2.1. Razvili so tudi hibridne metode, ki združujejo lastnost prostorske delitve in delitev sil [13, 14, 63, 64].

1.4.3 Globalne operacije

Globalne operacije so komunikacijske operacije, v katerih sodelujejo vsi procesorji. Primer take operacije je *globalno oddajanje*: podatek, ki je na enem procesorju, moramo poslati vsem procesorjem.

Pri vzporedni simulaciji MD imamo dve glavni globalni operaciji: porazdeljeno globalno oddajanje in porazdeljeno globalno vsoto. Ponazoril ju bomo na primeru repliciranih podatkov. Sosledje korakov, kot je prikazano na sliki 1.12 (str. 26) narekuje podatke, ki si jih morajo izmenjati procesorji [16].

Pri vzporedni metodi repliciranih podatkov izhajamo iz omejitev, da morajo imeti vsi procesorji na vsakem koraku simulacije koordinate vseh atomov. Vsak procesor i izračuna nove koordinate c_i za svoj del $N/|P|$ vseh atomov; te atome

imenujemo *domače atome* procesorja i.

Porazdeljeno globalno oddajanje

Po izračunu novih koordinat mora vsak procesor p poslati koordinate c_i svojih domačih atomov vsem ostalim procesorjem. Po izvedeni globalni operaciji oddajanja ima vsak procesor celotno množico koordinat vseh atomov, $\{c_0, c_1, \ldots, c_P\}$.

Porazdeljena globalna vsota

Vsak izmed procesorjev računa poljubno interakcijo, saj ima koordinate vseh atomov. Tako dobimo izračunane sile na atome, vendar so porazdeljene po procesorjih: za vsak atom moramo sešteti delne sile, ki se nahajajo na vseh procesorjih, da dobimo celotno silo, ki deluje na atom. To vsoto delnih sil za nek atom moramo imeti na tistem procesorju, ki računa nove koordinate za ta atom.

Po izračunu sil ima vsak procesor i delne sile za vse atome, $\{\vec{f_0^i}, \vec{f_1^i}, \ldots, \vec{f_P^i}\}$, kjer je $\vec{f_p^i}$ delna sila, ki jo izračuna procesor i za domače atome procesorja p. Da lahko nek procesor p izračuna nove koordinate svojim domačim atomom, mora imeti njihovo celotno silo $\vec{F}_p = \vec{f_p^0} + \vec{f_p^1} + \cdots + \vec{f_p^P}$. Torej vsak procesor p mora dobiti vsoto vseh delnih sil $\vec{f_p^i}$, $\forall i \in P$. Za to oddajanje in vsoto delnih sil uporabimo globalno operacijo porazdeljene globalne vsote.

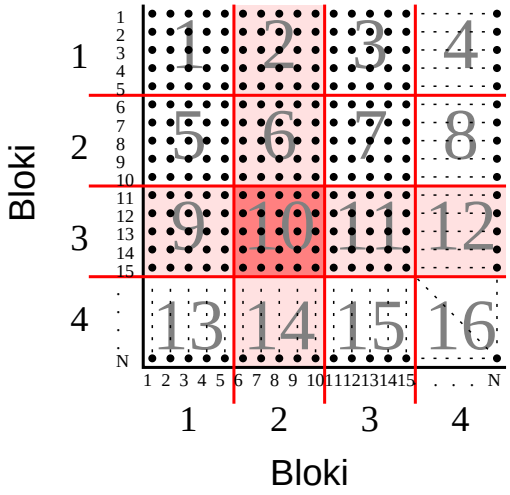
Poglavje 2

Nova vzporedna metoda delitev sil

V tem poglavju bom predstavil metodo delitev sil in njen izvor. Opisal bom objavljene različice in razložil novo metodo, ki sem jo razvil. Opisal bom rešitve, ki sem jih poiskal za težave, ki nastopijo, če želimo metodo delitev sil uporabiti za simulacijo realnih makromolekulskih sistemov. Sledi analiza komunikacijske zahtevnosti metode, ki kaže na njeno uporabnost pri večjem številu procesorjev. Ob tem bom opisal, kako sem implementiral usklajevanje prenosov podatkov med procesorji.

2.1 Delitev sil

Delitev sil je metoda za vzporedno računanje sil med delci, ki temelji na delitvi matrike sil. Vse interakcije med N delci tvorijo matriko sil velikosti $N \times N$. To matriko sil razdelimo na dele, s tem pa porazdelimo računanje interakcij na več procesorjev. Smisel metode delitev sil je v načinu delitve matrike, tako da procesorjem ni potrebno poznati koordinat vseh atomov. S tem zmanjšamo komunikacijski čas med procesorji in povečamo vzporedno učinkovitost. Atome razdelimo v $|B|$ blokov in vsakemu paru blokov dodelimo procesor, kot je nakazano na sliki 2.1. Procesor računa interakcije med delci svojega para blokov, kar ustreza vektorskemu produktu blokov v matriki sil. Skupaj tvorijo vsi produkti pokritje celotne matrike sil. Podatke o atomih v bloku izmenjuje le s procesorji, ki so dodeljeni istemu bloku. Metode za delitev sil se razlikujejo po načinu razporeditve delcev v bloke in načinu dodeljevanja procesorjev produktom blokov.



Slika 2.1: Osnovni princip metode delitev sil. Atome porazdelimo po blokih (na sliki v 4 bloke), pare blokov pa dodelimo procesorjem, ki računajo interakcije med njimi. Ponavadi uporabimo enako porazdelitev v obeh smerih. Na primer procesor 10 računa interakcije med atomi, ki so dodeljeni blokoma 2 in 3. Vektorski produkt teh dveh blokov je obarvan temno rdeče.

2.1.1 Izvor metode

Delitev

Elementi matrike sil so sile atoma j na atom i, $\vec{f}_{ij}$, za izračun katere moramo poznati koordinate obeh atomov, $\vec{r}_i$ ter $\vec{r}_j$. Slika 2.2 prikazuje matriko sil in ponazarja izvor enega elementa matrike iz interakcije med atomoma. Vsota vrste i v matriki,

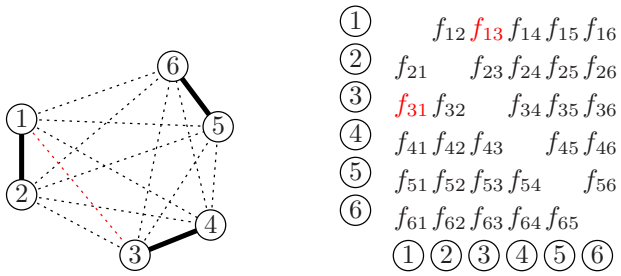
$$\vec{f}_i = \sum_j \vec{f}_{ij}, \tag{2.1}$$

je celotna sila, ki deluje na atom i, kot je prikazano na sliki 2.3. Vidimo, da moramo za izračun sile $\vec{f}_i$ poznati delne sile $\vec{f}_{ij}$ za vsak atom j. Če sile računamo vzporedno, potem mora imeti procesor, ki računa neko silo $\vec{f}_{ij}$, koordinate atomov i in j. Oziroma, procesor ne potrebuje koordinat atomov, s katerimi ne računa delnih sil.

Pri računanju interakcij upoštevamo 3. Newtonov zakon, po katerim sta sili $\vec{f}_{ij}$ in $\vec{f}_{ji}$ nasprotno enaki,

$$\vec{f}_{ij} = -\vec{f}_{ji}. \tag{2.2}$$

Izračunamo lahko le polovico sil, za manjkajoče pa upoštevamo zgornjo enakost. Tako prepolovimo računsko zahtevnost neveznih interakcij. Ker atomi nimajo lastnih interakcij s sabo, so sile na diagonali enaki 0: $\vec{f}_{ii} = 0$. Računanje vsote neke sile $\vec{f}_i = \vec{f}_{i1} + \vec{f}_{i2} + \cdots + f_{i,(i-1)} - f_{(i+1),i} - \cdots - \vec{f}_{N,i}$ je prikazano na sliki 2.4.



Slika 2.2: Izvor matrike sil. Na podsliki a so prikazane tri molekule kisika O_2 z oštevilčenimi atomi in vsemi interakcij med njimi. Na podsliki b) so interakcije predstavljene v obliki *matrike sil*. Sili med atomoma 1 in 3, ki je označena z rdečo barvo na podsliki a, ustrezata rdeče obarvana elementa $(1,3)$ in $(3,1)$ matrike sil na podsliki b. Vsota vseh sil v prvi vrsti oziroma sile ostalih 5 atomov na prvi atom je celotna sila, ki deluje na prvi atom.

$$
\begin{array}{cc}
f_1 & \textcircled{1} \\
f_2 & \textcircled{2} \\
f_3 & \textcircled{3} \\
f_4 & \textcircled{4} \\
f_5 & \textcircled{5} \\
f_6 & \textcircled{6}
\end{array}
\quad
\begin{array}{c}
f_{12}\, f_{13}\, f_{14}\, f_{15}\, f_{16} \\
f_{21}\quad f_{23}\, f_{24}\, f_{25}\, f_{26} \\
f_{31}\, f_{32}\quad f_{34}\, f_{35}\, f_{36} \\
f_{41}\, f_{42}\, f_{43}\quad f_{45}\, f_{46} \\
f_{51}\, f_{52}\, f_{53}\, f_{54}\quad f_{56} \\
f_{61}\, f_{62}\, f_{63}\, f_{64}\, f_{65} \\
\textcircled{1}\,\textcircled{2}\,\textcircled{3}\,\textcircled{4}\,\textcircled{5}\,\textcircled{6}
\end{array}
$$

Slika 2.3: Na vsak atom deluje sila $\vec{f}_i$, ki je vsota sil $\vec{f}_{i1} + \vec{f}_{i2} + \cdots + \vec{f}_{iN}; i \neq j$ i-te vrstice. Vsota rdeče označenih sil v prvi matriki sil je sila, ki deluje na atom 1.

$$
\begin{array}{cc}
f_1 & \textcircled{1} \\
f_2 & \textcircled{2} \\
f_3 & \textcircled{3} \\
f_4 & \textcircled{4} \\
f_5 & \textcircled{5} \\
f_6 & \textcircled{6}
\end{array}
\quad
\begin{array}{c}
\\
f_{21} \\
f_{31}\, f_{32} \\
f_{41}\, f_{42}\, f_{43} \\
f_{51}\, f_{52}\, f_{53}\, f_{54} \\
f_{61}\, f_{62}\, f_{63}\, f_{64}\, f_{65} \\
\textcircled{1}\,\textcircled{2}\,\textcircled{3}\,\textcircled{4}\,\textcircled{5}\,\textcircled{6}
\end{array}
$$

Slika 2.4: Sila $\vec{f}_3$, ki deluje na atom 3, je vsota rdeče označenih sil $\vec{f}_{31} + \vec{f}_{32} - \vec{f}_{43} - \vec{f}_{53} - \vec{f}_{63}$. Sile $\vec{f}_{43}$, $\vec{f}_{53}$ in $\vec{f}_{63}$ imajo negativen predznak, saj so nasprotno enake silam $\vec{f}_{34}$, $\vec{f}_{35}$ in $\vec{f}_{36}$.

Atome iz množice atomov A disjunktno razdelimo na bloke $b_1, b_2, \ldots, b_{|B|}$ iz množice blokov B s surjektivno preslikavo

$$\mathcal{B}\colon A \to B, \tag{2.3}$$

tako, da velja

$$b_i = \{a\colon \mathcal{B}(a) = b_i\}. \tag{2.4}$$

Vsakemu paru blokov $\{b_i, b_j\} \in B \times B$ priredimo procesor p iz množice procesorjev P, $p \in P,\ |P| = |B|^2$, z bijektivno preslikavo

$$\hat{\mathcal{B}}\colon P \to B \times B \tag{2.5}$$

in zapišemo $\hat{B}_p = \hat{\mathcal{B}}(p)$. Procesor p računa interakcije iz produkta blokov $b_i \times b_j$. Lahko definiramo tudi množico procesorjev, ki so dodeljeni vsakemu izmed blokov:

$$\hat{P}_b = \{p\colon b \in \hat{B}_p\}; \tag{2.6}$$

za te procesorje pravimo, da pripadajo bloku b. Na sliki 2.1 na primer velja $\hat{P}_2 = 2, 5, 6, 7, 8, 10, 14$ oziroma procesorji $5, 6, 7, 8$ in $2, 10, 14$ pripadajo bloku 2.

Uvedimo surjektivno preslikavo

$$\mathcal{P}\colon A \to P, \tag{2.7}$$

ki vsakemu atomu $a \in A$ priredi t.i. *domači procesor* $p = \mathcal{P}(a),\ p \in P$. Za preslikavo $\mathcal{P}$ naj velja, da je vsak atom prirejen enemu izmed procesorjev, ki je dodeljen bloku kateremu pripada atom:

$$\mathcal{P}(a) \in \hat{P}_b,\ b = \mathcal{B}(a). \tag{2.8}$$

Vsakemu procesorju definirajmo tudi množico *domačih atomov*,

$$A_p = \{a\colon p = \mathcal{P}(a)\}. \tag{2.9}$$

Procesor p računa nove koordinate svojim domačim atomom A_p.

Ob predpostavki, da so atomi enakomerno porazdeljeni v bloke (torej velja $|b_i| = |b_j|,\ b_i, b_j \in B$), so tudi vsi produkti $b_i \times b_j$ blokov enako veliki, $|b_i \times b_i| = |b_i| \cdot |b_i|$, torej imajo vsi enako število interakcij. Posledično to pomeni, da vsi procesorji računajo enako število interakcij, torej so enakomerno obremenjeni.

Prisotnost podatkov

Na procesorju morajo biti prisotni tisti podatki, ki jih procesor potrebuje za svoj del računov, torej za računanje sil in za računanje novih koordinat.

Vsak procesor računa sile interakcij med atomi dveh blokov, katerima je dodeljen. Če ima procesor p dodeljena bloka b_i in b_j, torej $(b_i, b_j) = \hat{\mathcal{B}}(p)$, potem računa delne sile $\vec{f}_{m,n},\ a_m \in b_i,\ a_n \in b_j$; za te izračune mora imeti vse koordinate $\{\vec{r}_k \colon a_k \in b_i \cup b_j\}$ atomov iz blokov b_i in b_j.

Za izračun novih koordinat mora imeti vsak procesor p sile $\vec{f}_i,\ i \in A_p$ svojih domačih atomov A_p.

Ker za vsak procesor p velja $A_p \subset b_i \cup b_j,\ (b_i, b_j) = \hat{\mathcal{B}}(p)$, lahko procesor omeji podatke le na atome svojih dveh blokov; ostalih ne potrebuje. Ni potrebno, da ima koordinate in izračunane sile vseh N atomov molekulskega sistema, ampak mora imeti podatke le o $2N/|B|$ atomih, po $N/|B|$ za vsakega izmed svojih dveh dodeljih blokov b_i in b_j.

Prenos podatkov

Pri metodi delitev sil poteka prenos podatkov po blokih: le procesorji, ki pripadajo istemu bloku, izmenjujejo podatke.

Oddajanje koordinat

Trditev. Če si vsi procesorji, ki imajo skupen blok, izmenjajo koordinate po njihovem izračunu, potem ima vsak procesor vse potrebne podatke za izračun sil.

Dokaz. Procesor p ima izračunane koordinate svojih domačih atomov A_p. Procesor pripada množici procesorjev $\hat{P}_{b_i}$ in $\hat{P}_{b_j}$. Procesor p pošlje vsem procesorjem v $\hat{P}_{b_i}$ koordinate $\vec{r}_k$ atomov $a_k \in A_p \cap b_i$ in vsem procesorjem v $\hat{P}_{b_j}$ koordinate $\vec{r}_l$ atomov $a_l \in A_p \cap b_j$. Ker je vsak blok ustrezno razdeljen (enačba 2.8), imajo sedaj vsi procesorji $\hat{P}_b$ vsakega bloka b vse koordinate, ki jih nato potrebujejo za računanje interakcij. □

Vsota sil

Trditev. Če vsi procesorji, ki imajo skupen blok, izvedejo vsoto izračunanih delnih sil, tako da je za vsak atom ustrezna vsota prisotna na njegovem domačem

procesorju, potem ima vsak procesor vse potrebne podatke za izračun novih koordinat.

Dokaz. Procesor p, ki je dodeljen blokoma b_i ter b_j ($p \in \hat{P}_{b_i} \cup \hat{P}_{b_j}$), ima izračunane delne vsote $\vec{f}_{m,n}$, $a_m \in b_i,\ a_n \in b_j$. Za vsak blok $b \in B$ velja, da če vsak procesor iz $\hat{P}_b$ pošlje vse delne sile $\vec{f}_{k,n}$, $a_k \in A_k$ procesorju k, $p_k \in \hat{P}_b$, potem ima vsak procesor p_k vsote sil $\vec{f}_l$, $l \in A_k$. Tako lahko izračuna nove koordinate svojim domačim atomom. □

Torej vsak procesor p z dodeljenima blokoma b_i, b_j komunicira le s procesorji iz množic $\hat{P}_{b_i}$ in $\hat{P}_{b_j}$, torej s procesorji, ki so dodeljeni blokoma b_i in b_j.

2.2 Obstoječe metode

Vzporedna metoda delitev sil za računanje simulacije MD je bila prvič predstavljena nekaj let po tem, ko so za simulacije pričeli uporabljati vzporedne računalnike [15, 56–58]. Kmalu po prvi taki metodi so predstavili še nekaj različic, potem pa teh metod niso več razvijali. Med razlogi lahko naštejemo tudi takrat vse večjo uporabo metode Ewald [21, 69] za računanje kristalnih struktur in drugih periodičnih molekulskih sistemov v povezavi z razdaljo *cut-off*. Obstoječe metode za delitev sil niso prilagojene uporabi razdalje *cut-off*.

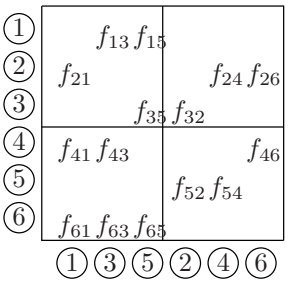
2.2.1 Metoda po Plimptonu in Hendricksonu

Prvo objavljeno metodo za delitev sil sta objavila Plimpton in Hendrickson za računanje Lennard-Jonesove tekočine [56].

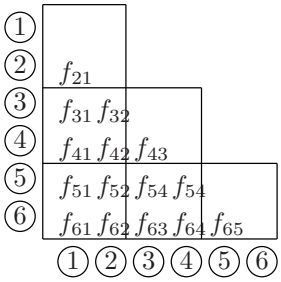
Metoda ima dve različni delitvi atomov na $|B|$ blokov, tako da je matrika sil permutacija take matrike sil, kot smo jo opisovali doslej. Uporablja dve različni preslikavi; $\mathcal{B}$, ki slika v bloke B, in $\mathcal{B}'$, ki slika v bloke B'. Množici blokov tvorita produkte blokov $b_i \times b'_j,\ b_i \in B,\ b_j \in B'$. Primer majhne matrike sil je prikazana na sliki 2.5. Upoštevamo vseh B^2 produktov blokov, $b_i \times b'_j$, za kar potrebujemo tudi B^2 procesorjev. Dvojnemu računanje sil $\vec{f}_{ij}$ in $-\vec{f}_{ji}$ se izognemo tako, da nad pri lihi vsoti $i + j$ računamo le silo nad diagonalo, pri sodi vsoti $i + j$ pa le silo pod diagonalo.

2.2.2 Metoda po Shuju

Shu in sodelevci [62] v svoji metodi uporabijo táko matriko sil, kot smo jo opisovali do sedaj. V matriki sil dodelijo vsakemu produktu blokov, ki vsebuje interakcije svoj procesor, kot je prikazano na sliki 2.6. Kot je razvidno iz slike 2.6 se število



Slika 2.5: Matrika sil, kot se uporablja v metodi po Plimptonu in Hendricksonu pri delitvi atomov na dva bloka. Delitvi v vodoravni in navpični smeri se razlikujeta.



Slika 2.6: Matrika sil, kot se uporablja v metodi po Shuju pri delitvi atomov na 3 bloke.

interakcij, ki pripadajo procesorjem, razlikuje med procesorji na diagonali in ostalimi procesorji.

2.3 Nova vzporedna metoda delitev sil s porazdelitvijo diagonale

Predstavim novo metodo delitev sil, metoda *delitev sil s porazdelitvijo diagonale* (DDFD, Distributed Diagonal Force Decomposition). Pri tej metodi imajo vsi procesorji enake vrste računov, poleg tega je možno računanje prestavljati med procesorji in tako omogočiti uravnoteženost računanja, posledično pa tudi povečanje vzporedne učinkovitosti. Uporabna je tudi ob upoštevanju razdalje *cut-off*. V primerjavi z drugimi metodami delitev sil uporablja metoda DDFD pri enakemu številu procesorjev večje število manjših blokov, kar zmanjša komunikacijsko zahtevnost metode.

2.3.1 Porazdelitev diagonale

Razlog za porazdelitev

Pri novi metodi imamo le eno porazdelitev atomov na bloke, kot je prikazano na sliki 2.7a Tako dobimo produkte blokov, prikazanih na sliki 2.7b, ki so podobni

kot pri metodi po Shuju [62]. Ustrezna matrika sil je simetrična, zato računanje zgornjega trikotnika (sivi produkti na sliki 2.7b) ni potrebno. Diagonalni produkti (obarvani rdeče na sliki 2.7b) so produkti istih blokov, $b_i \times b_i$, zato so drugačni od ostalih produktov, ki so produkti dveh različnih blokov, $b_i \times b_j,\ i \neq j$; ti vsebujejo $|b_i| \cdot |b_j|$ različnih interakcij. Ker so v diagonalnih produktih interakcije le med eno množico atomov, tudi tu velja, da zgornjega trikotnika matrike ni potrebno računati. Izmed $|b_i| \cdot |b_i|$ interakcij jih moramo računati le polovico, $|b_i| \cdot |b_i|/2$, kot je prikazano na sliki 2.7c.

Če bi vsakemu produktu na diagonali ali spodnjem trikotniku priredili procesor, tako kot naredijo Shu idr., potem bi imeli procesorji, prirejeni diagonalnim produktom, manjšo računsko obremenitev od ostalih. Polovična obremenjenost nekaterih procesorjev v primerjavi z drugimi poslabša vzporedno učinkovitost.

Porazdelitev diagonale

Ob upoštevanju, da so produkti na diagonali produkti enega bloka s sabo, lahko vse interakcije iz diagonalnih produktov porazdelimo po ostalih procesorjih. Računanje interakcij iz diagonale produkta $b_i \times b_i$ nekega bloka b_i lahko porazdelimo po procesorjih $\hat{P}_{b_i}$, torej po vseh procesorjih, ki računajo interakcije med blokom b_i in enem izmed ostalih blokov. Na sliki 2.8a so z modro barvo prikazani vsi taki procesorji $\hat{P}_{b_3}$ za primer bloka b_3, porazdelitev interakcij pa je prikazana na sliki 2.8b. Vsak procesor $p \in \hat{P}_{b_i}$ že ima vse podatke, ki so potrebni za izračun poljubne podmnožice teh interakcij $b_i \times b_i$: ima vse podatke o atomih b_i, saj jih potrebuj za izračun interakcij $b_i \times b_j$ z drugim blokom $b_j, j \neq i$. Dopolnimo tudi definicijo preslikave $\hat{\mathcal{B}}^{-1}$, tako da dodamo preslikave s parov atomov na procesorje $\hat{\mathcal{B}}\colon P \rightarrow A \times A$ in definiramo:

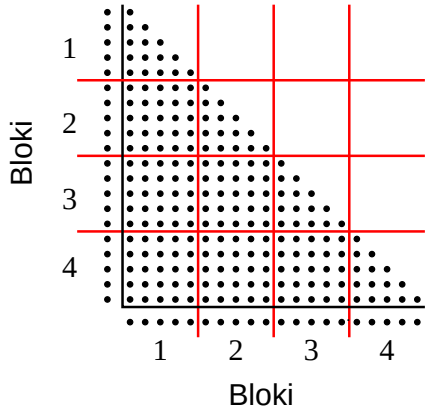
$$p = \hat{\mathcal{B}}^{-1}((a_i, a_j)), \tag{2.10}$$

preslikavo $\hat{\mathcal{B}}$ pa ustrezno dopolnimo, da ohranimo bijektivnost. Dopolnjeno preslikavo $\hat{\mathcal{B}}^{-1}$ uporabimo za določanje procesorja, ki bo računal interakcijo med dvema atomoma. Računanje sil

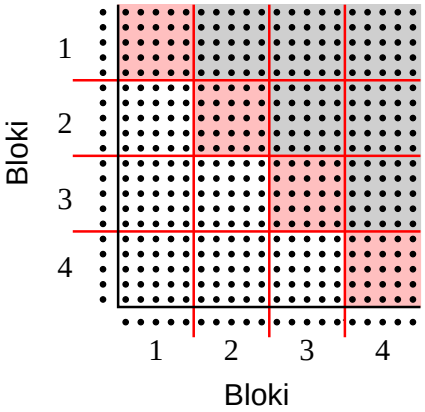
$$\vec{f}_m = \sum_{\{n:a_n \in b_j\}} \vec{f}_{mn}$$

se spremeni tako, da niso vse delne sile f_{mn}, $a_m, a_n \in b_i$ dodeljene istemu procesorju, temveč so porazdeljene med procesorji iz množice $\hat{P}_{b_i}$; končna vsota, sila f_m, ostane nespremenjena. Na sliki 2.8c je prikazana porazdelitev celotne diagonale.

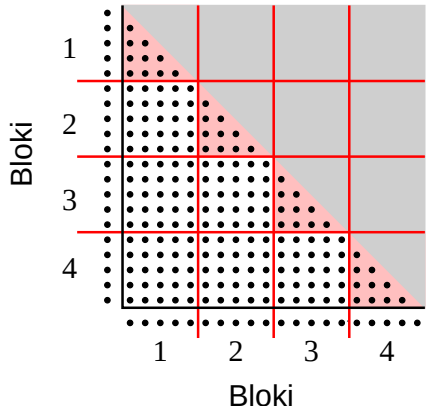
Pri novi metodi DDFD porazdelimo vse interakcije iz diagonalnih produktov blokov; ti produkti ostanejo prazni, zato jim ni potrebno dodeliti procesorja. Procesorje dodelimo le produktom v spodnjemu trikotniku matrike sil. Končna delitev



(a) Porazdelitev atomov na bloke in produkti blokov. Obe porazdelitvi, navpična in vodoravna, sta enaki.

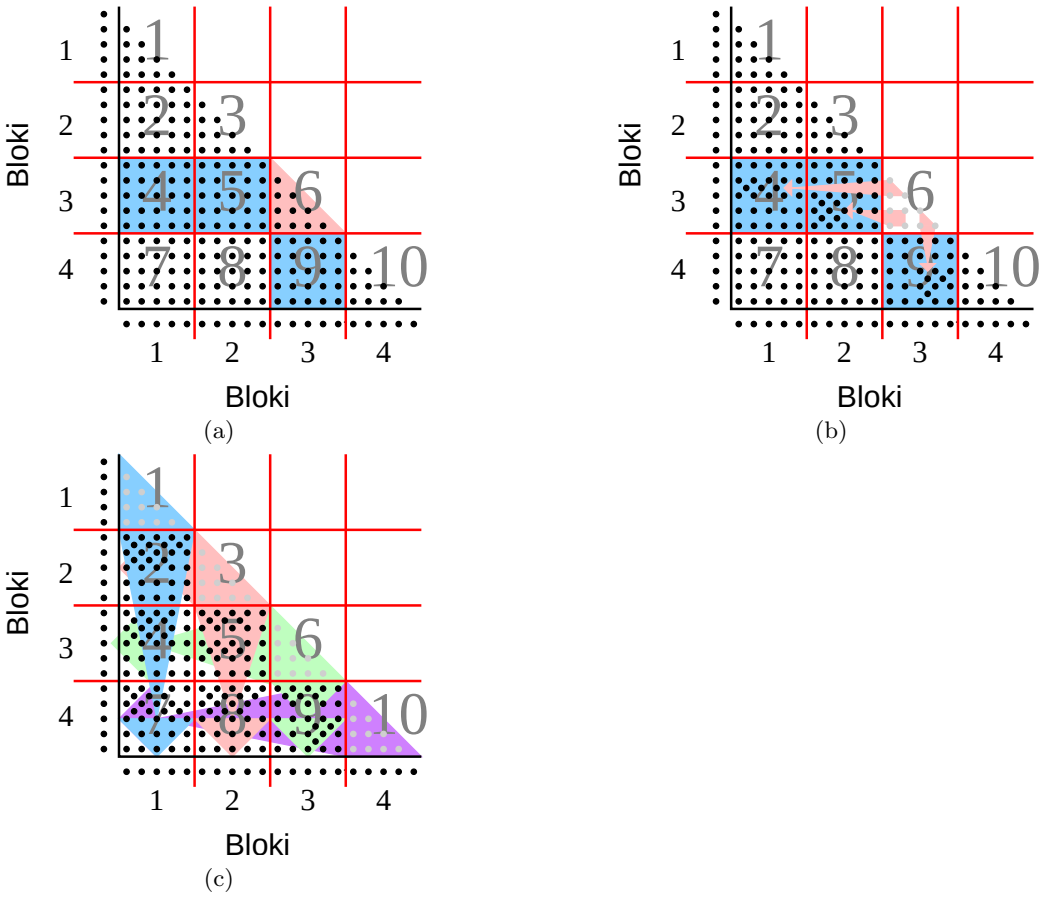


(b) Produkti blokov. Računanje sivih produktov je nepotrebno: vsak izmed njih ima zrcalno sliko med neobarvanimi produkti pod diagonalo. Rdeči produkti na diagonali (t.j. na diagonalnih produktih) so produkti med pari istih blokov.

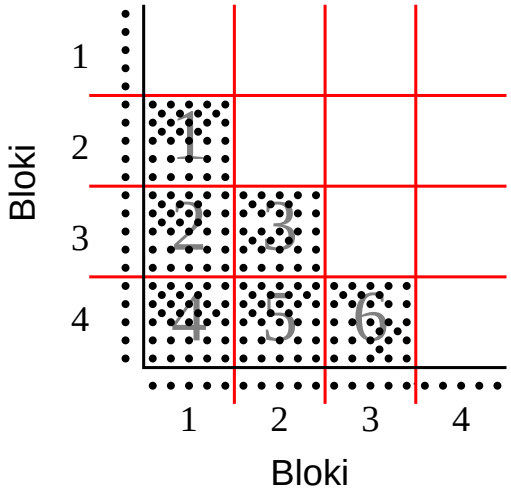


(c) Produkti blokov na diagonali. Ker so produkti na diagonali produkti enega bloka s samim sabo, je polovica interakcij v njem zrcalnih; interakcij nad diagonalo ne računamo. Odstranjene so tudi interakcije iz produktov blokov nad diagonalo in interakcije atomov s samim sabo.

Slika 2.7: Različnost produktov blokov nad diagonalo, pod diagonalo in na diagonali pri metodi DDFD.



Slika 2.8: Množica procesorjev, katerim lahko dodelimo računanje interakcij z diagonalnega produkta. V primeru bloka 3 (podslika a) je to množica $\hat{P}_{b_3} = \{p_4, p_5, p_9\}$ (brez p_6): interakcije enakomerno dodelimo vsem trem procesorjem (podslika b). Na zadnji sliki (podslika c) je prikazana porazdelitev interakcij diagonalnih produktov vseh blokov po procesorjih. Procesorji na diagonali p_1, p_3, p_6 in p_{10} ne računajo nobenih interakcij.



Slika 2.9: Dodelitev procesorjev za računanje interakcij pri novi metodi DDFD. Procesorje dodelimo le produktom blokov pod diagonalo.

računanja interakcij pri novi metodi je prikazana na sliki 2.9.

Število procesorjev, ki jih metoda DDFD lahko uporabi, je odvisno od števila blokov:

$$|P| = \frac{|B|(|B|-1)}{2}, \tag{2.11}$$

na primer 3, 6, 10, 15, ...

2.3.2 Vezne interakcije

Vzporedna metoda mora upoštevati tudi vezne interakcije. Med vsemi interakcijami med N atomi jih je le nekaj, $\mathcal{O}(N)$, veznih; v praksi je veznih interakcij med N in $2N$ [10, 24, 47, 64]. Vezne interakcije ne nastopajo le med pari atomov (vezi), temveč tudi med tremi (koti), ponavadi pa do štirimi atomi (dihedralni koti). Ker je metoda za delitev sil prirejena računanju parnih interakcij, kakršne so nevezne in vezi, moramo metodo prilagoditi, da pravilno obravnava tudi večatomske interakcije.

V zasnovi nove metode delitev sil ima vsak procesor podatke o dveh blokih, torej dveh podmnožicah atomov. Le med temi lahko izračuna interakcijo. Če vsi atomi ene interakcije pripadajo le dvema blokoma, potem ni potrebna dodatna obravnava te interakcije. Pri večatomskih interakcijah pa so lahko vsi trije atomi v različnih blokih, kot prikazuje slika 2.10a. V tem primeru noben procesor nima podatkov o vseh treh atomih, zato noben ne more izračunati take interakcije. Težavo premostimo tako, da vsaki interakciji priredimo domači procesor in vse podatke prenašamo na ta procesor in z njega.

Določimo, da je domači procesor p neke večatomske interakcije atomov a_i, a_j, a_k, ...

enak domačemu procesorju p dveh atomov $a_i \in b_i,\ a_j \in b_j$, torej $p = \hat{\mathcal{B}}^{-1}((a_i, a_j))$, ostale atome večatomske interakcije pa proglasimo za *sirote*. Podatke vseh sirot prenašamo na domači procesor, kar poteka po blokih (izmenjava le znotraj neke množice $\hat{P}_b$): v bloku b_i ima nek procesor q podatke o vsaki siroti, torej za vsako siroto $a_k \in b_k$ velja, da $\exists q \in \hat{P}_{b_k} \cap \hat{P}_{b_i}$. Procesor q pošlje koordinate sirote a_k procesorju p (saj oba pripadata bloku $\hat{P}_{b_i}$) po porazdeljenem globalnem oddajanju, tako da ima procesor p podatke vseh sirot in lahko izračuna interakcije; ta prenos imenujemo *oddajanje sirot* in je ponazorjen na sliki 2.10c. Pred porazdeljeno globalno vsoto pa procesor p pošlje izračunane sile interakcij procesorju q, tako da se izračunane interakcije pravilno upoštevajo tudi za sirote pri računanju novih koordinat; prenos imenujemo *zbiranje sirot* in je ponazorjen na sliki 2.10c. Oddajanje in zbiranje vseh sirot izvedemo na vsakem koraku simulacije.

Časovna zahteva prenosov sirot

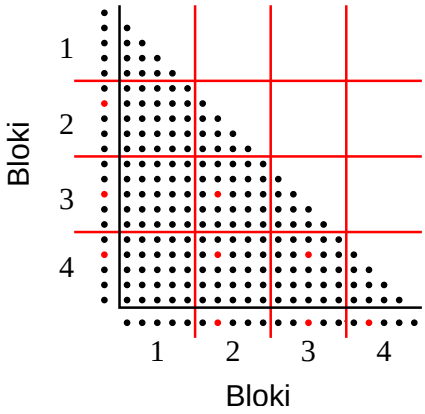
Oddajanje in zbiranje sirot zahteva tudi določen čas za prenos podatkov. Pokažemo lahko, da je količina prenešenih podatkov pri prenosu sirot majhna v primerjavi z ostalimi prenešenimi podatki. V tabeli 2.1 je prikazano število atomov vseh interakcij za dva molekulska sistema, na katerih smo izvajali simulacije MD. Za kotne, dihedralne in neprave dihedralne interakcije je prikazano tudi število sirot. Razvidno je, da je število sirot majhno v primerjavi s številom atomov, zato tudi njihov prenos zahteva manj časa. Glede na porazdelitev atomov po blokih se spreminja število sirot. V tabeli so prikazana dejanska števila za izvedene simulacije.

Tabela 2.1: Število sirot pri računanju na 6 procesorjih (stolpec sirote/6) in na 10 procesorjih (stolpec sirote/10) na dveh molekulskih sistemih s 54212 in 14026 atomi. Prikazano je tudi število vezi, kotov, dihedralnih (dih. koti) in nepravih dihedralnih (nepr. dih. koti) kotov za molekulska sistema.

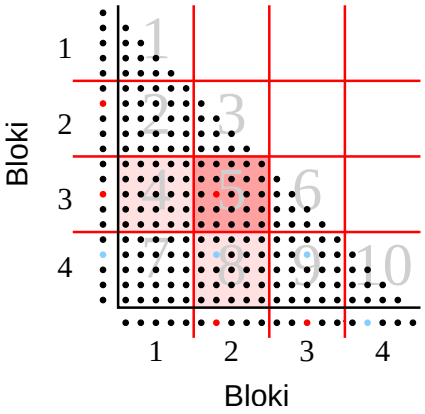
atomi	vezi	koti	dih. koti	nepr. dih. koti	sirote/6	sirote/10
54212	54235	21604	6293	587	1444	1627
14026	14062	8506	1200	635	2482	2897

Število sirot je majhno v primerjavi z velikostjo blokov, torej predstavlja njihov prenos le majhen delež celotnega prenosa podatkov. Njihov prenos zahteva kvečjemu eno izmenjavo med procesorji istega bloka, kar predstavlja dodaten korak pri globalnih prenosih.

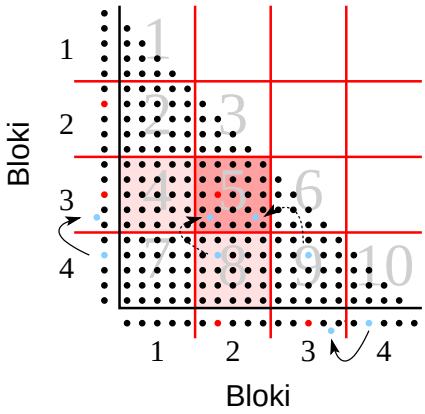
Če bi izbrali drugačno porazdelitev atomov v bloke $\mathcal{B}$, bi se sirote spremenile, saj bi se spremenilo razbitje večatomskih interakcij na bloke. Z ustrezno porazdelitvijo bi lahko načrtno zmanjšali njihovo število. Npr. za kotne interakcije bi



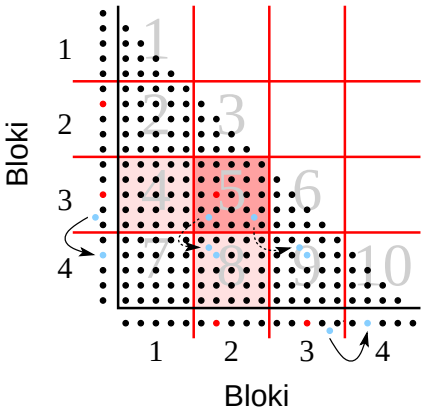
(a) Pri interakcijah treh atomov se lahko zgodi, da vsak atom pripada drugemu bloku. Prikazani rdeči atomi pripadajo trem različnim blokom, $2, 3, 4$. Noben procesor nima podatkov o vseh treh atomih, zato ne more noben izračunati interakcije med njimi. Tri rdeče označene interakcije predstavljajo tri deleže celotne interakcije; le če so na istem procesorju lahko izračunamo celotno interakcijo.



(b) Interakciji treh atomov dodelimo domači procesor, v tem primeru je to domači procesor za interakcije med atomi 2. in 3. bloka, procesor 5. Zadnji atom (moder), ki pripada bloku 4, postane t.i. sirota, saj procesor 5 nima njegovih koordinat.



(c) Kopiranje koordinat modro označenega atoma iz bloka 4 v blok 3. Sedaj ima procesor 5, ki računa interakcijo med vsemi tremi atomi, koordinate vseh atomov in lahko izračuna celotno tri-atomsko interakcijo.



(d) Kopiranje izračunanih interakcij modro označenega atoma iz bloka 3 nazaj v svoj domači blok 4. Domači procesor tega atoma ima sedaj celotno silo, ki je potrebna za izračun novih koordinat tega atoma.

Slika 2.10: Upoštevanje sirot pri računanju sil in novih koordinat v interakcjah treh atomov iz treh različnih blokov.

tretji atom vedno dodelili istemu bloku, kot je prvi ali drugi atom. V tem primeru ne bi bilo sirot, saj vsi trije atomi pripadajo le dvema blokoma. Težava nastopi, ko se zaradi spremenjene porazdelitve atomov v bloke spremenijo produkti blokov. Kot je kasneje opisano v podpoglavju 3.1, se take spremembe produktov blokov odražajo v spremembi obremenjenosti procesorjev, kar lahko občutno zmanjša vzporedno učinkovitost. Pri odpravljanju neenakomerne obremenjenosti pa se lahko kot cilj upošteva tudi zmanjšanje števila sirot.

2.4 Globalne operacije pri metodi DDFD

Kot pri vseh vzporednih metodah za računanje simulacij MD morajo procesorji izmenjavati podatke tudi pri metodah delitev sil. Globalne operacije prikrojimo komunikacijskim vzorcem [8], ki so značilni za metodo deljenja sil.

2.4.1 Neodvisne operacije po blokih

Globalni operaciji porazdeljenega globalnega oddajanja in porazdeljene globalne vsote, predstavljeni v podpoglavju 1.4.3, se izvajata na vsakem koraku simulacije; ker sta edini operaciji, ki se redno izvajata, se osredotočimo nanju. Za zmanjšanje obsega in časa komunikacij lahko uporabimo dejstvo, da vsi procesorji nimajo podatkov o vseh N atomih. Vsak procesor mora imeti podatke le o atomih v svojih dveh blokih. Podatke je potrebno prenašati le znotraj blokov, kar pomeni, da procesorji komunicirajo le z $\mathcal{O}(\sqrt{P})$ procesorjev s skupnima blokoma.

Znotraj vsakega bloka pa morajo vsi procesorji izmenjati podatke o vseh $N/|B|$ atomih bloka. Če se omejimo na posamezen blok b, torej na atome tega bloka in procesorje $\hat{P}_b$, je izmenjava podatkov bloka enaka operacijama porazdeljenega oddajanja in vsote: vsi ti procesorji iz $\hat{P}_b$ si morajo izmenjati vse podatke o atomih v bloku b; tako dobimo lokalno operacijo. Vsak blok ima torej lokalni operaciji porazdeljenega oddajanja in vsote.

Zaradi omejenega prenašanja podatkov po blokih lahko uvedemo tri različice globalnega oddajanja in globalne vsote:

polni način, ki je ekvivalenten globalnemu oddajanju oziroma vsoti;

notranji način, kjer podatke izmenjujejo le procesorji istega bloka (z lokalno operacijo za vsak blok);

zunanji način, ki dopolnjuje notranji način v polno operacijo globalnega oddajanja oziroma vsote.

Notranji način je najbolj uporabljena različica, saj se uporablja med simulacijo MD; ostale se uporabijo le na začetku ali na koncu simulacije, npr. za prenos začetnih koordinat z glavnega procesorja na ostale. Izvedba zunanjega načina za notranjim načinom je ekvivalentna izvedbi ene operaciji v polnemu načinu.

Komunikacijska zahtevnost

Globalni operaciji razbijemo na $|B|$ neodvisnih lokalnih operacij, po eno za vsak blok. Vsaka lokalna operacija bloka b poteka na vseh $|B| - 1 = |\hat{P}_b|$ procesorjih bloka in izmenjuje $|b| = \mathcal{O}(N/|B|) = \mathcal{O}(N/\sqrt{|P|})$ podatkov (število atomov v bloku). Za izvedbo prenosov lokalne operacije se zgledujemo po izvedbi globalnih komunikacij v hiperkocki [8, 73]. Pri taki izvedbi je komunikacijska zahtevnost vsake lokalne operacije vsota zakasnitve $l \log_2(|B-1|)$ in časa prenosa $s2N/|B|$, kjer je l zakasnitev, s pa hitrost prenosa v danem omrežju. Kadar je število procesorjev táko, da $|B-1| \neq 2^x, x \in \{1, 2, \ldots\}$ (kadar $|P| \notin \{10, 36, \ldots\}$) potem se zakasnitev poveča za $2l$, čas prenosa pa za največ $s(N/|B| + N/(|B| \cdot |B-1|))$.

Lokalne operacije so neodvisne, vendar ne morejo potekati povsem sočasno, saj vsak procesor sodeluje v dveh lokalnih operacijah, po eno za vsak svoj blok. Ob predpostavki, da procesorji ne morejo hkrati pošiljati in prejemati podatke za obe lokalni operaciji, je komunikacijska zahtevnost globalne operacije dvakratnik zahtevnosti ene lokalne operacije. Skupna zahtevnost je torej zakasnitev $2l \log_2(|B|-1) = l\mathcal{O}(\log_2 \sqrt{|P|})$ in čas prenosa $s4N/|B| = s\mathcal{O}(N/\sqrt{|P|})$.

Za primerjavo, globalna operacija za vzporedno metodo repliciranih podatkov poteka na vseh $|P|$ procesorjih in izmenjuje N podatkov. Njena komunikacijska zahtevnost je zakasnitev $l \log_2 |P|$ in čas prenosa $s2N$ (neodvisno od števila procesorjev), kar je skupno $l \log_2 |P| + s2N$. Število korakov – in zakasnitev – je enakega reda pri obeh metodah. Vendar je čas prenosa manjši pri globalnih operacijah metode DDFD že pri $|B| > 3$ oziroma pri 6 ali več procesorjih.

2.4.2 Urnik prenosov

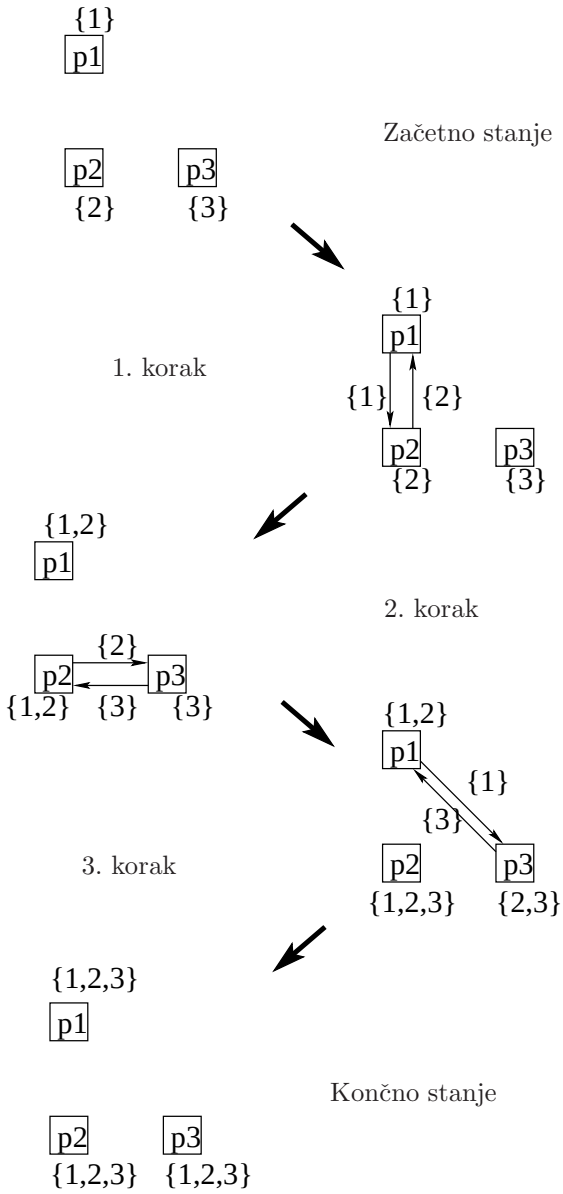
Za izvedbo operacij porazdeljenega globalnega oddajanja in porazdeljene globalne vsote je potrebno izvesti več enostavnih prenosov med dvema procesorjema. Pri vsakem takem prenosu prenesemo nekaj podatkov (npr. koordinat ali sil) z enega na drug procesor. Prenose lokalnih operacij izvedemo kot komunikacijo v hiperkocki [73]. Ker procesorji sodelujejo v dveh lokalnih operacijah, enostavni prenosi in njihovo zaporedje niso enostavno določljivi med samim izvajanjem globalnih operacij, zato jih vnaprej izračunamo. Tako določimo *urnik* enostavnih prenosov, katerega upoštevamo za izvedbo globalne operacije.

Urnik sestavljajo delno urejeni enostavni prenosi. Četverica (korak, izvor, ponor, podatki) določi vsak tak prenos. Več neodvisnih prenosov lahko poteka sočasno, zato so delno urejeni po koraku; vsi prenosi z enakim korakom se izvajajo sočasno. Sestava urnika zavisi od povezanosti procesorjev in drugih omejitev strojne opreme, tako da niso zasičene le nekatere povezave. Na primer pri enoprocesorskih računalnikih povezanih z mrežnim stikalom moramo upoštevati, da en procesor ne sme hkrati sodelovati pri več kot enem enostavnem prenosu.

Primer urnika za porazdeljeno globalno oddajanje na treh procesorjih je prikazan na sliki 2.11a. Na sliki 2.11b je prikazan potek operacije po opisanem urniku. Na vsakem koraku se izvedejo enostavni prenosi, predpisani za trenutni korak.

Korak	Izvor	Ponor	Podatek
1	1	2	1
1	2	1	2
2	2	3	2
2	3	2	3
3	1	3	1
3	3	1	3

(a) Urnik prenosa globalnega porazdeljenega oddajanja na 3 procesorjih.



(b) Prisotnost podatkov na procesorjih pri izvedbi porazdeljene globalne operacije oddajanja koordinat. Procesorji so predstavljeni s kvadrati, nad ali pod njimi pa je zapisana množica podatkov, ki je prisotna na vsakem; podatki za množico atomov A_1 so predstavljeni s številko 1. Zgornja slika kaže stanje pred začetkom globalne operacije, nato pa sledijo izmenjave podatkov na vsakem izmed izvršenih korakov po urniku iz podslike a. Spodnja slika vsebuje končno stanje, ko ima vsak procesor vse potrebne podatke – koordinate vseh atomov svojih dveh blokov $\hat{\mathcal{B}}(p)$.

Slika 2.11: Primer urnika in izvedbe urnika na 3 procesorjih.

Poglavje 3

Uravnoteženje računanja pri metodi delitev sil s porazdelitvijo diagonale

Pri metodi delitev sil atome porazdelimo v bloke. Interakcije v produktu dveh blokov ustrezajo interakcijam med pari atomov, katere moramo izračunati. Če so vsi bloki enako veliki, potem so enaki tudi produkti blokov; če moramo izračunati vse interakcije, potem je računska obremenitev računanja interakcij enaka za vse produkte blokov. Ker je vsak produkt dodeljen enemu procesorju, je računska obremenitev procesorjev enaka in računske neuravnoteženosti ni.

Sprememba kake izmed naštetih predpostavk povzroči, da računske obremenitve računanja interakcij različnih produktov blokov niso enake. Če je v enem produktu potrebno izračunati več interakcij kot v ostalih produktih, potem je računanje neuravnoteženo. Procesor, ki je dodeljen produktu z večimi interakcijami, računa dlje od ostalih. Zato morajo ostali procesorji čakati nanj pri naslednji globalni komunikaciji. Takšno čakanje povzroči, da se celotni izvajalni čas simulacije podaljša, vzporedna učinkovitost pa se zmanjša.

Za preprečevanje neuravnoteženosti računanja moramo uporabiti metode, ki uravnavajo računsko obremenitev procesorjev. Za novo metodo DDFD sem razvil metodo, ki omogoča dinamično uravnoteženje računanja med izvajanjem vzporedne simulacije MD.

3.1 Vzroki neuravnoteženega računanja pri metodi delitev sil

Metoda DDFD porazdeli računanje interakcij po vseh procesorjih. Porazdeljeno je računanje vseh neveznih in veznih interakcij.

3.1.1 Nevezne interakcije

Interakcijo med atomoma a_i in a_j, ki pripadata blokoma $b_i = \mathcal{B}(a_i)$ in $b_j = \mathcal{B}(a_j)$ po porazdelitvi 2.3, računa procesor $p = \hat{\mathcal{B}}^{-1}((b_i, b_j))$, kot je definirano s preslikavo 2.5. Celotna metoda temelji na porazdelitvi računanja neveznih interakcij. Začetno porazdelitev atomov v bloke $\mathcal{B}$ naredimo enakomerno, tako da je število neveznih interakcij enakomerno porazdeljeno po procesorjih [64]. Zaradi enakih velikosti blokov (t.j. bloki imajo enako število atomov), $|b_i| = |b_j|,\ b_i, b_j \in B$, so tudi njihovi produkti enako veliki, $|b_i \times b_j| = |b_k \times b_l|,\ b_i, b_j, b_k, b_l \in B$ oziroma, produkti imajo enako število interakcij. Če računamo vse interakcije, potem imajo vsi procesorji enako računsko obremenitev računanja neveznih interakcij.

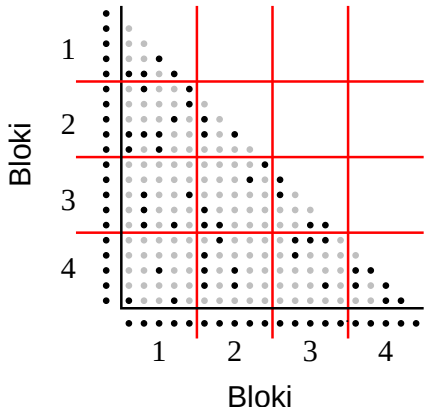
Za pohitritev računanja simulacij mnogokrat uporabimo metodo *cut-off*, pri kateri privzamemo, da interakcije med oddaljenimi atomi ni oziroma, da je enaka 0 (slika 1.7; podrobnejši opis je v podpoglavju 1.2.2). Ob taki predpostavki zato ni potrebno računati oddaljenih interakcij.

Na sliki 3.1 je prikazana matrika sil po uvedbi razdalje *cut-off*. Interakcije med atomi, oddaljenimi več kot je razdalja *cut-off*, ne računamo. (V dovolj velikih sistemih, kjer je premer sistema mnogo večji od razdalje *cut-off*, večine interakcij ni potrebno računati.) Računska zahtevnost je torej manjša, kot če bi morali računati vseh N^2 interakcij. Vendar, kot je razvidno iz slike 3.1, preveč oddaljene interakcije, ki jih ne računamo, niso enakomerno porazdeljene po produktih blokov. Na primer, v produktu blokov $b_1 \times b_4$ so 3 interackij preveč oddaljene, v produktu blokov $b_1 \times b_2$ pa jih je 9. Posledično sta tudi procesorja $p = \hat{\mathcal{B}}^{-1}((b_1, b_4))$ in $q = \hat{\mathcal{B}}^{-1}((b_1, b_2))$ neenakomerno obremenjena, saj mora procesor p računati 3 interakcije, procesor q pa 9.

Neenakomerno porazdeljene nevezne interakcije, ki jih moramo računati na vsakem koraku simulacije, vodijo v neuravnoveženo računanje.

3.1.2 Vezne interakcije

Kot je opisano v podpoglavju 2.3.2, temelji porazdelitev veznih interakcij na porazdelitvi neveznih interakcij. Vezne interakcije med dvema atomoma a_i, a_j dodelimo procesorju $p = \hat{\mathcal{B}}^{-1}((a_i, a_j))$, enako kot bi pri nevezni interakciji med



Slika 3.1: Matrika sil po uvedbi razdalje *cut-off* pred porazdelitvijo diagonale. Interakcije med atomi, ki so oddaljeni več kot je razdalja *cut-off*, so obarvane sivo. Takih interakcij ni potrebno računati. Razvidno je, da imajo produkti blokov sedaj različno število črno obarvanih interakcij, katere morajo izračunati. Procesorji imajo zato različne računske obremenitve, kar privede do neuravnoteženega računanja.

tema atomoma. Vezne interakcije treh ali več atomov $a_i, a_j, a_k, \ldots$ dodelimo enemu izmed procesorjev $\hat{\mathcal{B}}^{-1}((a_i, a_j)), \hat{\mathcal{B}}^{-1}((a_i, a_k)), \ldots, \hat{\mathcal{B}}^{-1}((a_j, a_k)), \ldots$. Lahko definiramo, da vedno vzamemo prvo možnost, torej računanje interakcije dodelimo procesorju $\hat{\mathcal{B}}^{-1}((a_i, a_j))$.

Značilnost veznih interakcij je, da jih je relativno malo, le $\mathcal{O}(N)$ v nasprotju z neveznimi, ki jih je $\mathcal{O}(N^2)$. Ob enakomerni porazdelitvi $\mathcal{B}$ je tudi porazdelitev veznih interakcij enakomerna a ne nujno uravnotežena, vendar razlike niso opazne.

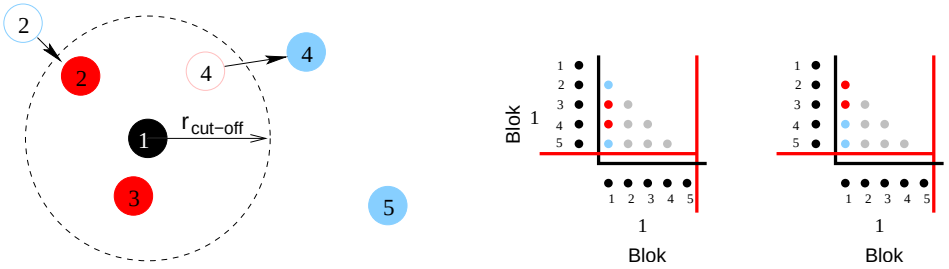
3.1.3 Dinamika neuravnoteženosti

Med simulacijo MD se atomi gibljejo po prostoru, zaradi česar se spreminjajo razdalje med atomi. Pri uvedbi razdalje *cut-off* se zato med simulacijo spreminjata tudi komplementarni množici interakcij: tiste, ki jih računamo in tiste, ki jih ne računamo. Na sliki 3.2 je prikazan vpliv gibanja atomov na interakcije v matriki sil.

Spremembe razdalj med atomi povzročijo spremembe v računski obremenjenosti procesorjev med simulacijo, zaradi česar se spreminja tudi neuravnoteženost računanja.

3.2 Merjenje neuravnoteženosti

Za oceno neuravnoteženosti moramo definirati meritev neuravnoteženosti, torej moramo izmeriti ali oceniti obremenitev procesorjev. Neuravnoteženost računanja potem zmanjšamo tako, da izenačimo računsko obremenitev med procesorji s presta-



Slika 3.2: Vpliv gibanja atomov na matriko sil pri razdalji *cut-off* z vidika atoma 1 (črn). Interakcije in atomi v dosegu *cut-off* so obarvane rdeče, preveč oddaljeni atomi in njihove ustrezne interakcije pa modro. Interakcije med ostalim atomi so obarvane sivo. V prejšnjem koraku simulacije je bil atom 2 (sedaj rdeč) oddaljen več kot je razdalja *cut-off*, zato se interakcija med atomoma 1 in 2 ni računala; na podsliki b (matrika sil v prejšnjem koraku) je ustrezna interakcija označena z modrim krogom. Sedaj se je atom 2 približal atomu 1 in je v dosegu razdalje *cut-off*, zato se interakcija med atomoma 1 in 2 računa. V matriki sil na podsliki c je zato ustrezna interakcija označena rdeče. Nasprotno se je atom 4 (sedaj moder) oddaljil. Prej se je interakcija računala, sedaj je pa preveč oddaljen in se interakcija ne računa več. Atom 3 (obakrat rdeč) je v obeh korakih dovolj blizu, da se njegova interakcija z atomom 1 računa, medtem kot je atom 5 (obakrat moder) obakrat predaleč. Dotični del matrike sil se ne spremeni za ta dva atoma. V prikazanem primeru ne pride do spremembe števila interakcij, ki jih moramo računati, je pa prikazana dinamika tovrstnih interakcij.

vljanjem računanja med njimi. Za to moramo poznati trenutno računsko obremenitev procesorjev ter računsko zahtevnost računov, ki jih prerazporejamo.

3.2.1 Teoretična računska obremenitev

Računsko obremenitev merimo in tudi modeliramo. Z meritvijo časa računanja ugotavljamo neuravnoteženost in nastavimo parametre modela neuravnoteženosti. Model pa nam omogoča, da ocenimo neuravnoteženost, predvsem pa nam pomaga pri selitvi računanja interakcij z enega procesorja na drug procesor.

Značilnost neposredne meritve časa je, da izmerimo dejansko računsko obremenitev. V tako izmerjenem času so upoštevani tako čas, ki je potreben za izračun interakcij, kot tudi drugi dejavniki, ki jih težje modeliramo, na primer različno obnašanje predpomnilnikov na različnih procesorjih zaradi podatkov, ki se med njimi razlikujejo.

Čas celotnega koraka vzporedne simulacije MD, t_{korak}, na nekem procesorju zapišemo z vsoto

$$\begin{aligned} t_{\text{korak}} = t_{\text{račun sil}} + t_{\text{čakanje sil}} + t_{\text{porazdeljena globalna vsota sil}} \\ + t_{\text{računanje premika}} + t_{\text{čakanje koordinat}} + t_{\text{porazdeljeno oddajanje koordinat}}, \end{aligned} \tag{3.1}$$

kjer je $t_{\text{račun sil}}$ čas računanja vseh interakcij, $t_{\text{porazdeljena globalna vsota sil}}$ je čas izvajanja komunikacije in računanja porazdeljene globalne vsote sil, $t_{\text{računanje premika}}$ je čas računanja novih koordinat, $t_{\text{porazdeljeno oddajanje koordinat}}$ je čas izvajanja komunikacije pri porazdeljenem globalnem oddajanju, časa $t_{\text{čakanje sil}}$ in $t_{\text{čakanje koordinat}}$ pa sta časa, kolikor mora procesor čakati druge, da dokončajo računanje (računanje interakcij oziroma novih koordinat), predno si izmenjajo izračunane podatke. Komunikacijski časi $t_{\text{porazdeljena globalno vsota sil}}$ so nujno enaki na vseh procesorjih, prav tako so časi $t_{\text{porazdeljeno oddajanje koordinat}}$ nujno enaki na vseh procesorjih.

Zaradi enakomerne porazdelitve na domače procesorje je čas $t_{\text{računanje premika}}$ enak na vseh procesorjih, saj vsi računajo nove koordinate enakemu številu atomov, $|A_p|,\ p \in P$. Računanje novih koordinat je torej uravnoteženo in časi $t_{\text{čakanje koordinat}}$ so enaki 0 na vseh procesorjih.

Pri neuravnoteženem računanju se razlika med procesorji pojavi le pri času $t_{\text{račun sil}}$. Vsota $t_{\text{račun sil}} + t_{\text{čakanje sil}}$ mora biti enaka na vseh procesorjih, ta pa je enaka času $\max_{p \in P}(t_{\text{račun sil}})$ na najbolj obremenjenem procesorju (na katerem je $t_{\text{čakanje sil}} = 0$). Na ostalih procesorjih je čas $t_{\text{čakanje sil}}$ enak razliki med časom računanja sil najbolj obremenjenega procesorja in lastnim časom računanja sil, $t_{\text{čakanje sil}} = \max_{p \in P}(t_{\text{račun sil}}) - t_{\text{račun sil}}$.

Model računske obremenitve

Računsko obremenitev modeliramo z modelom računske zahtevnosti, kateremu nastavimo parametre na podlagi izmerjenih obremenitev. Model je podan z enačbo, ki pove, koliko časa poteka računanje interakcij v posameznih delih enega koraka simulacije, $t_{\text{račun sil}}$,

$$t_{\text{račun sil}} = t_{\text{nb}} n_{\text{nb}} + t_{\text{b}} n_{\text{b}} + t_{\text{ostalo}}, \tag{3.2}$$

kjer je t_{nb} čas, potreben za izračun ene nevezne interakcije, n_{nb} je število neveznih interakcij, ki jih dani procesor računa, t_{b} je čas, potreben za izračun ene vezne interakcije, n_{b} je število veznih interakcij, ki jih dani procesor računa, v času t_{ostalo} pa so zajeti vsi ostali računi in komunikacijski čas. V modelu predpostavimo, da je čas t_{ostalo} enak na vseh procesorjih ne glede na računsko obremenitev interakcij. Tudi predpostavimo, da je čas računanja ene nevezne interakcije, t_{nb} enak na vseh procesorjih in prav tako, da je čas računanja ene vezne interakcije, t_{b} enak na vseh procesorjih

Časa t_{nb} in t_{b} določimo z meritvijo časa računanja večjega števila interakcij obeh vrst. V tabeli 3.1 so zbrane meritve časa računanja neveznih in veznih interakcij pri simulaciji realnega molekulskega sistema. Iz teh meritev dobimo ustrezne vrednosti za časa t_{nb} in t_{b}. Razvidno je, da so časi računanja veznih interakcij $t_{\text{b}} n_{\text{b}}$ bistveno manjši od časa računanja neveznih interakcij $t_{\text{nb}} n_{\text{nb}}$. Poleg tega se časi računanja neveznih interakcij ne razlikujejo med procesorji. Zato bomo tudi za čas računanja neveznih interakcij predpostavili, da je konstanten za vse procesorje. Tako nam ostane linearen model

$$t_{\text{račun}} = \underbrace{t_{\text{nb}}}_{\text{konstanta}} n_{\text{nb}} + \underbrace{(t_{\text{b}} n_{\text{b}} + t_{\text{ostalo}})}_{\text{konstanta}}, \tag{3.3}$$

kjer se čas t_{korak} spreminja le s spremembo števila neveznih interakcij n_{nb}.

Tabela 3.1: Meritve časa računanja in določanje časa računanja veznih in neveznih interakcij na procesorju AMD Opteron 242 molekulskega sistema 14026 atomov in razdalji *cut-off* 14 Å. Število neveznih interakcij je povprečno število, saj se točno število spreminja med simulacijo.

vrsta interakcij	število interakcij	čas računanja vseh interakcij	hitrost računanja	čas računanja ene interakcije
vezne	24403	5.8×10^{-3} s	$4.2 \times 10^{6}\ s^{-1}$	2.4×10^{-7} s
nevezne	6710237 ± 4431	5.334×10^{-1} s	$1.26 \times 10^{7}\ s^{-1}$	7.95×10^{-8} s

Pri uravnoteženju računanja poskušamo zagotoviti, da so časi t_{korak} enaki na vseh procesorjih. Ker sta časa t_{b} in t_{ostalo} konstantna, lahko upoštevamo cilj, da

naj bo čas, ki je potreben za računanje neveznih interakcij ($t_{\mathrm{nb}} n_{\mathrm{nb}}$) enak na vseh procesorjih. Torej ni potrebno meriti čas računanja, ampak lahko namesto časa upoštevamo število neveznih interakcij, ki jih računamo.

Računanje uravnotežimo s prerazporejanjem računanja neveznih interakcij; tako si zadamo cilj, da vsak procesor računa enako število neveznih interakcij. Pri tem ni potrebno poznati časa t_{nb}, saj je konstanten. Zato lahko računske obremenitve računanja neveznih interakcij, kar na nekem procesorju traja $t_{\mathrm{nb}} n_{\mathrm{nb}}$, neposredno primerjamo kar s številom neveznih interakcij n_{nb}. To število opredelimo po procesorjih $p \in P$ z oznako nb_p.

Neuravnoteženost

Na podlagi modela računske obremenitve, opisanega z enačbo 3.3 in časa izvajanja celotnega koraka simulacije, opisanega z enačbo 3.1 v primeru neuravnoteženega računanja, lahko izrazimo neuravnoteženost računanja neveznih interakcij s faktorjem

$$nu = \frac{\max\limits_{p \in P}(nb_p)}{\mathrm{avg}_{p \in P}(nb_p)}. \tag{3.4}$$

kjer je $\max_{p \in P}(nb_p)$ računska obremenitev najbolj obremenjenega procesorja, $\mathrm{avg}_{p \in P}(nb_p)$ pa je povprečna obremenjenost procesorjev, ki je enaka času računanja vseh interakcij deljeno s številom procesorjev. Uravnotežen račun ima faktor $nu = 1$, faktorji neuravnoteženosti nekaterih realnih računov so prikazani v tabeli 3.2.

Tabela 3.2: Faktorji neuravnoteženosti nu pri simulacijah različno velikih molekulskih sistemov z različnim številom procesorjev.

število procesorjev	velikost sistema	povprečni nu.
3	14026	1.03
3	54212	1.04
6	14026	1.06
6	54212	1.04
10	14026	1.05
10	54212	1.01
36	14026	1.53
45	14026	1.44

Pri danem faktorju neuravnoteženja nu lahko izračunamo, za koliko se podaljša čas računanja enega koraka simulacije. Če je faktor $nu = 1$, potem vsi procesorji interakcije računajo $\mathrm{avg}_{p \in P}(nb_p)$ časa. Čim faktor $nu \neq 1$, morajo vsi procesorji čakati najbolj zasedenega; ta računa interakcije nu-krat dlje, kot bi,

če bi bilo računanje uravnoteženo. Posledično je čas računanja interakcij omejen z najbolj obremenjenim procesorjem. Računanje interakcij se podaljša za razliko časov $\max_{p\in P}(nb_p) - \text{avg}_{p\in P}(nb_p)$, prav toliko se podaljša čas računanja enega koraka simulacije.

Faktor neuravnoteženosti se ne spreminja veliko med simulacijo, zato lahko oceno podaljšanja časa računanja enega koraka simulacije uporabimo kot oceno za časovno podaljšanje t_{dod} neuravnoteženega računanja celotne simulacije z n_k koraki,

$$t_{\text{dod}} = n_k \left(\max_{p\in P}(nb_p) - \text{avg}_{p\in P}(nb_p) \right). \tag{3.5}$$

3.2.2 Cena uravnoteženja

Meritve neuravnoteženosti in prerazporejanje računanja tudi porabijo določen čas. Če je ta čas velik oziroma večji, kot je čas, ki ga pridobimo z uravnoteženjem, potem je uravnoteženje nekoristno. Enačbi 3.1 dodajmo člen $t'_{\text{uravnoteženje}}$, ki predstavlja čas uravnoteženja:

$$\begin{aligned} t'_{\text{korak}} = {} & t'_{\text{uravnoteženje}} + t'_{\text{račun sil}} + t'_{\text{čakanje sil}} + t'_{\text{porazdeljena globalna vsota sil}} \\ & + t'_{\text{računanje premika}} + t'_{\text{čakanje koordinat}} + t'_{\text{porazdeljeno oddajanje koordinat}}, \end{aligned} \tag{3.6}$$

kar je čas računanja enega koraka simulacije pri dinamičnem uravnoteženju. Členi $t'_{\text{porazdeljena globalna vsota sil}}$, $t'_{\text{računanje premika}}$, $t'_{\text{čakanje koordinat}}$, $t'_{\text{porazdeljeno oddajanje koordinat}}$ ostanejo enaki tistim iz enačbe 3.1 (npr. $t'_{\text{računanje premika}} = t_{\text{računanje premika}}$). Zaradi dinamičnega uravnoteženja velja, da je čas $t'_{\text{čakanje sil}} = 0$ in

$$t'_{\text{račun sil}} + t'_{\text{čakanje sil}} = t'_{\text{račun sil}} = \text{avg}_{p\in P}(t_{\text{račun sil}}) \tag{3.7}$$

za vse procesorje. Uravnoteženje je smotrno le, če pohitri računanje, torej če velja neenačba

$$t'_{\text{korak}} < t_{\text{korak}} \tag{3.8}$$

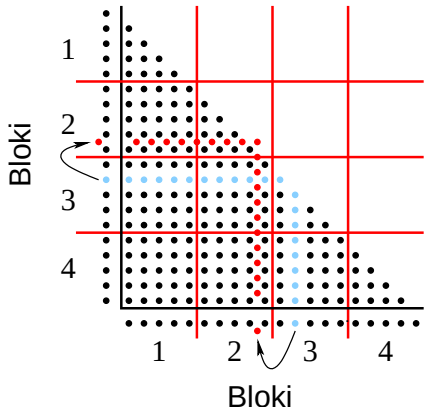
oziroma

$$t'_{\text{uravnoteženje}} + t'_{\text{račun sil}} < t_{\text{račun sil}} + t_{\text{čakanje sil}} \tag{3.9}$$

za vse procesorje. Najbolj obremenjen procesor ima najdaljše računanje sil, čaka pa ne, saj ostali čakajo nanj. Neenačbo 3.9 lahko torej prepišemo drugače,

$$t'_{\text{uravnoteženje}} + t'_{\text{račun sil}} < \max_{p\in P} t_{\text{račun sil}} \tag{3.10}$$

$$t'_{\text{uravnoteženje}} < \max_{p\in P} t_{\text{račun sil}} - t'_{\text{račun sil}} \tag{3.11}$$



Slika 3.3: Prikaz selitve modro obarvanega atoma iz bloka 3 v blok 2 (sedaj rdeče obarvan) in posledična selitev računanja interakcij med tem in ostalimi atomi, kar je tudi ponazorjeno s spremembo barve iz modre v rdečo.

Čas $t'_{\text{uravnoteženje}}$, ki je potreben za uravnoteženje računanja, mora biti torej krajši od časa $max_{p\in P}(t_{\text{račun sil}}) - t'_{\text{račun sil}}$, t.j. čas, za kolikor pohitrimo računanje najbolj obremenjenega procesorja.

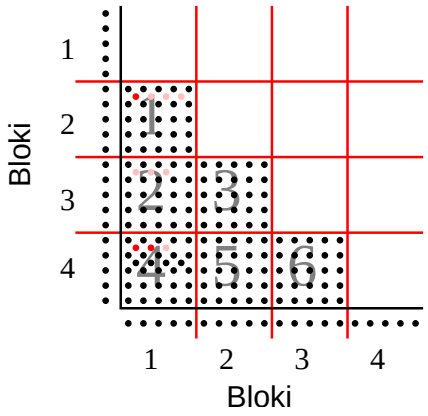
3.3 Uravnoteženje računanja pri metodi DDFD

3.3.1 Neuravnoteženost pri novi metodi DDFD

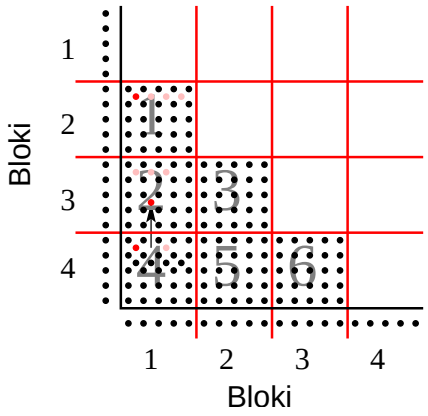
Pri novi metodi DDFD ima vsak procesor dodeljeni dve vrsti interakcij: interakcije v produktu dveh različnih blokov ter del interakcij iz porazdeljene diagonale. Pri uvedbi razdalje *cut-off* obe vrsti interakcij prispevata k neuravnoteženosti računanja.

Interakcije v produktu dveh različnih blokov so vedno dodeljene istemu procesorju. Za atoma a_i in a_j, ki pripadata različnima blokoma $a_i \in b_i = \mathcal{B}(a_i)$, $a_j \in b_j = \mathcal{B}(a_j)$, $b_i \neq b_j$, vedno velja, da je njuna interakcija dodeljena procesorju $p = \hat{\mathcal{B}}^{-1}((b_i, b_j))$. Primer neuravnoteženja zaradi takih interakcij je prikazan na sliki 3.1 (str. 51). Računanje interakcij iz različnih blokov bi lahko preselili na drug procesor le, če bi spremenili porazdelitev atomov v bloke, torej če bi spremenili preslikavo $\mathcal{B}$ in bi ustrezno spremenili tudi vse podatkovne strukture programa. Taka selitev atoma je prikazana na sliki 3.3. Izvedba uravnoteženja računanja s spremembo porazdelitve atomov v bloke je torej relativno zamudna operacija.

Interakcije z diagonale so take, kjer oba atoma a_i in a_j interakcije pripadata istemu bloku b_i. Tako interakcijo dodelimo procesorju $p = \hat{\mathcal{B}}^{-1}((a_i, a_j))$ iz množice $\hat{P}_{b_i}$. Na sliki 3.4a je prikazan primer neuravnoteženosti računanja zaradi interakcij s porazdeljene diagonale. Ker imajo vsi procesorji iz množice $\hat{P}_{b_i}$ podatke o vseh atomih



(a) Neuravnoteženost računanja zaradi interakcij s porazdeljene diagonale. Prikazane so le porazdeljene interakcije za blok 1. Temno rdeče interakcije moramo izračunati, svetlo rdeče interakcije pa zaradi *cut-off* ne računamo. Procesor 4 mora računati največ interakcij, 2 več od procesorja 2 in eno več od procesorja 1.



(b) Selitev računanja ene interakcije z diagonale s procesorja 4 na 2. Po selitvi vsi trije procesorji bloka 1 računajo enako število interakcij.

Slika 3.4: Neuravnoteženost računanja zaradi interakcij s porazdeljene diagonale in selitev računanja med procesorji.

bloka b_i, selitev takih interakcij ni računsko zahtevna, saj moramo le spremeniti podatek o dodelitvi interakcije novemu procesorju na vseh procesorjih iz množice P_{b_i}. Slika 3.4b prikazuje selitev takih interakcij s procesorja 4 na 2. Računanje lahko torej relativno enostavno uravnotežimo s prerazporejanjem interakcij diagonale.

3.3.2 Prerazporejanje diagonale

V novi metodi za uravnoteženje računanja spreminjamo razporeditev diagonale, da izenačimo računsko obremenitev vseh vzporednih procesorjev. Diagonalo prerazporejamo na podlagi sledečih ugotovitev:

1. glavnina neuravnoteženosti izhaja iz neuravnoteženosti računanja neveznih interakcij, torej želimo izenačiti računski čas neveznih interakcij na vseh procesorjih;

2. zaradi gibanja atomov se neuravnoteženost spreminja med izvajanjem simulacije;

3. imamo model neuravnoteženosti, ki tudi omogoča določanje števila neveznih interakcij, ki jih moramo prerazporediti, da bi bil računski čas računanja neveznih interakcij enak na vseh procesorjih;

4. nevezne interakcije z diagonale lahko zelo hitro prerazporedimo.

Nova metoda za uravnoteženje računanja dinamično prerazporeja nevezne interakcije z diagonale na podlagi modela neuravnoteženosti in računske obremenitve s ciljem minimizirati neuravnoteženost računanja.

Računsko obremenitev vseh nb neveznih interakcij razdelimo na število interakcij atomov iz različnih blokov, nb^{neq}, in število interakcij atomov z diagonale, nb^{eq}, kjer atomi pripadajo istemu bloku:

$$nb = nb^{neq} + nb^{eq} \tag{3.12}$$

Porazdelitev atomov v bloke nam določa tudi porazdelitev računanja neveznih interakcij atomov iz različnih blokov; število nb^{neq} lahko torej zapišemo z vsoto

$$nb^{neq} = \sum_{p \in P} nb_p^{neq}, \tag{3.13}$$

kjer je nb_p^{neq} število neveznih interakcij atomov različnih blokov, ki jih računa procesor p. Ta porazdelitev je ob nespremenljivi porazdelitvi atomov v bloke tudi nespremenljiva.

Število vseh neveznih interakcij, ki jih računa procesor $p \in P$, pa razdelimo na vsoto števila interakcij atomov iz različnih blokov nb_p^{neq} in števila interakcij atomov z diagonale nb_p^{eq}, kjer atomi pripadajo istemu bloku

$$nb_p = nb_p^{neq} + nb_p^{eq}. \tag{3.14}$$

Porazdelitev interakcij z diagonale matrike sil določi število nb_p^{eq} za vsak procesor. Njihova vsota je vedno enaka nb^{eq}, členi nb_p^{eq} vsote

$$nb^{eq} = \sum_{p \in P} nb_p^{eq} \tag{3.15}$$

pa se lahko spreminjajo.

S prerazporejanjem vsote 3.15 lahko dosežemo, da imajo vsi procesorji enako računsko breme.

Računanje je uravnoteženo, kadar vsak procesor računa $nb/|P|$ interakcij. Vendar pa imajo procesorji pri neuravnoteženem računanju odstopanja Δ_p:

$$\Delta_p = nb_p - \frac{nb}{|P|} = nb_p^{neq} + nb_p^{eq} - \frac{nb}{|P|}. \tag{3.16}$$

Ker lahko porazdelitev diagonale enostavno spreminjamo s spreminjanjem preslikave $\hat{\mathcal{B}}$ (enačba 2.10, str. 2.10), s prerazporeditvijo diagonale dosežemo zmanjšanje razlik Δ_p. Vsakemu procesorju dodelimo računanje toliko interakcij z diagonale, da bo njegova računska obremenjenost nb_p enaka ciljni obremenjenosti $nb/|P|$, torej interakcije z diagonale porazdelimo tako, da velja

$$nb_p^{eq} = \frac{nb}{|P|} - nb_p^{neq}. \tag{3.17}$$

Tako porazdelitev lahko izvedemo le ob pogoju, da je število interakcij atomov med različnimi bloki večje od ciljne obremenjenosti:

$$nb_p^{neq} < \frac{nb}{|P|}. \tag{3.18}$$

V takem primeru bi morali za uravnoteženje izvesti tudi prerazporeditev atomov v bloke, kot je prikazano na sliki 3.3 (str. 57).

Poglavje 4

Načrtovanje nove gruče za novo razvito metodo delitev sil

V tem poglovju bom predstavil zasnovo strojne opreme po značilnostih programske opreme in naštel prednosti, ki jih prinaša tako načrtovanje. Predstavil bom topologije dosedanjih gruč, ki sem jih načrtoval po zahtevah vzporednih programov. Nova metoda DDFD ima značilno delitev podatkov po procesorjih vzporednega računalnika; od tod izvirajo njeni vzorci komuniciranja med procesorji. Na podlagi vzorcev komuniciranja sem zasnoval gručo osebnih računalnikov, ki je prilagojena računanju simulacije MD. Načrtovana gruča ni omejena zgolj na to metodo, saj je uporabna za splošne vzporedne račune. V tem primeru se ne izkoristijo vse njene zmogljivosti. Predstavil bom tudi uporabnost načrtovane gruče v primeru okvare ali druge izločitve sestavnih računalnikov. Predstavil bom uporabo namenskih procesorjev MDGRAPE-II, ki omogočajo hitrejše računanje interakcij z metodo delitev sil.

4.1 Načrtovanje strojne opreme po programski opremi

Prilagodljivost in raznolikost strojne opreme je smotrno izkoristiti za povečanje njene računske zmogljivosti. Z načrtovanjem strojne opreme po značilnostih programske opreme, kateri je strojna oprema namenjena, lahko dosežemo pohitritev izvajanja programske opreme [8, 9, 11, 23].

4.1.1 Izbira topologije glede na komunikacijske vzorce

Pri vzporednih računalnikih s porazdeljenim pomnilnikom je potrebno posebno pozornost nameniti povezavam med procesorji. Mnogo vzporednih programov ima

majhno množico vzorcev komuniciranja. Taki vzorci karakterizirajo izmenjavo podatkov med procesorji: vzorci zajemajo sodelujoče procesorje, velikosti sporočil ter sosledje izmenjav sporočil. Z načrtovanjem povezovalne topologije na podlagi takih vzorcev komuniciranja se izognemo nepotrebnim povezavam in tako bolje izkoristimo uporabljene povezave.

Če ima nek procesor omejeno število povezav z drugimi procesorji, lahko uvedemo točkovne povezave. Točkovna povezava neposredno povezuje dva procesorja. Prednost točkovnih povezav je večja prepustnost v primerjavi z običajnim mrežnim stikalom [9]. Vendar pa stikalo nudi splošno povezljivost, torej med vsemi povezanimi računalniki. Osebni računalniki imajo le omejeno razširljivost in posledično je omejena povezljivost s točkovnimi povezavami; v običajnih osebnih računalnikih lahko implementiramo le nekaj, npr. 6, mrežnih povezav vrste Ethernet [9]. Če moramo procesorje povezati z večimi drugimi procesorji, moramo uporabiti mrežna stikala.

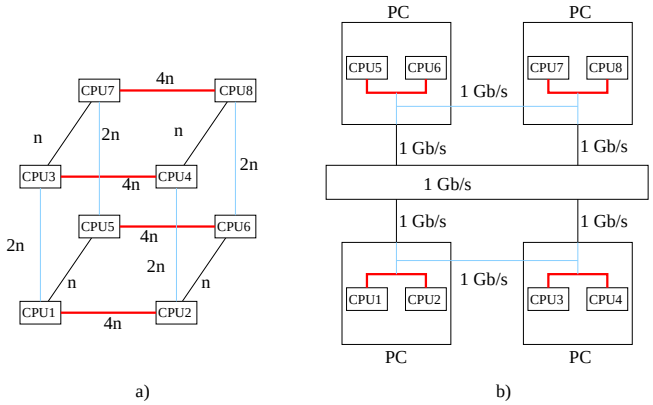
Hierarhična hiperkocka

Načrtovali smo topologijo, ki reši težavo omejene razširljivosti osebnih računalnikov tako, da uporabimo točkovne povezave za najbolj obremenjene povezave, splošno povezljivost pa dosežemo z uporabo mrežnega stikala. Tako dosežemo, da so povezave, kjer procesorji izmenjujejo več podatkov, hitrejše od povezav, po katerih procesorji izmenjujejo manj podatkov.

Implementirana topologija hierarhične hiperkocke za gruče osebnih računalnikov ima topologijo hiperkocke, v kateri se hitrosti in lastnosti povezav razlikujejo po njenih dimenzijah. Uporabna je za vzporedno simulacijo MD po metodi repliciranih podatkov, za katero je značilno, da globalni operaciji porazdeljene globalne vsote in porazdeljenega globalnega oddajanja potekata po vzorcu hiperkocke, zato lahko izkoristimo za njuno izvedbo topologijo hiperkocke [73]. Količina podatkov, ki se prenašajo po različnih dimenzijah hiperkocke, se razlikuje. Največji prenos je po prvi dimenziji, zato tem procesorskim parom dodelimo najhitrejše povezave. Najmanjši prenos je po zadnji dimenziji, zato tem procesorskim parom dodelimo najpočasnejše povezave. Shema topologije hierarhične hiperkocke je prikazana na sliki 4.1.

4.1.2 Gruče VRANA

Razvil in implementiral sem več gruč osebnih računalnikov VRANA (vzporedni računalnik za akceleracijo numeričnih algoritmov) z različnimi topologijami.



Slika 4.1: Princip topologije hierarhične hiperkocke, ponazorjene na 3-dimenzionalni kocki. Po različnih dimenzijah hiperkocke prenašamo različno količino podatkov (podslika a): po vsaki naslednji dimenziji prenašamo le polovico podatkov, kot v prejšnji. Po prvi dimenziji (rdeče povezave) prenašamo najdaljša sporočila, dolžine $4n$, po drugi dimenziji (modre povezave) so sporočila dolga $2n$, po zadnji dimenziji (črne povezave) pa so sporočila dolga le še n. Za najmanjšo dolžino sporočil n velja $n = 3N/|P|$, saj prenašamo podatke o koordinatah oz. silah po vseh treh koordinatnih oseh za $N/|P|$ atomov. Povezave nižjih dimenzij implementiramo s hitrejšimi povezavami kot povezave višjih dimenzij (podslika b). Štiri povezave prve dimenzije implementiramo z vodilom dvoprocesorskega računalnika, štiri povezave druge dimenzije implementiramo s točkovnimi povezavami, za prestale štiri povezave, tretje dimenzije, pa uporabimo skupno stikalo.

VRANA1

Gruča VRANA1 je sestavljena iz 4 dvoprocesorskih računalnikov s procesorji Intel Pentium II s hitrostjo 400 MHz. Računalniki so povezani s topologijo polnega grafa s točkovnimi povezavami Ethernet s hitrostrjo 100 Mb/s.

VRANA2 in VRANA3

Gruči VRANA2 in VRANA3 sta povezani s topologijo torusa. Uporabili smo točkovne povezave Ethernet s hitrostjo 100 Mb/s. Gruča VRANA2 je sestavljena iz 16 računalnikov s procesorji Intel Pentium II s hitrostjo 450 MHz, gruča VRANA3 pa iz 32 računalnikov s procesorji Intel Celeron s hitrostjo 466 MHz.

VRANA4

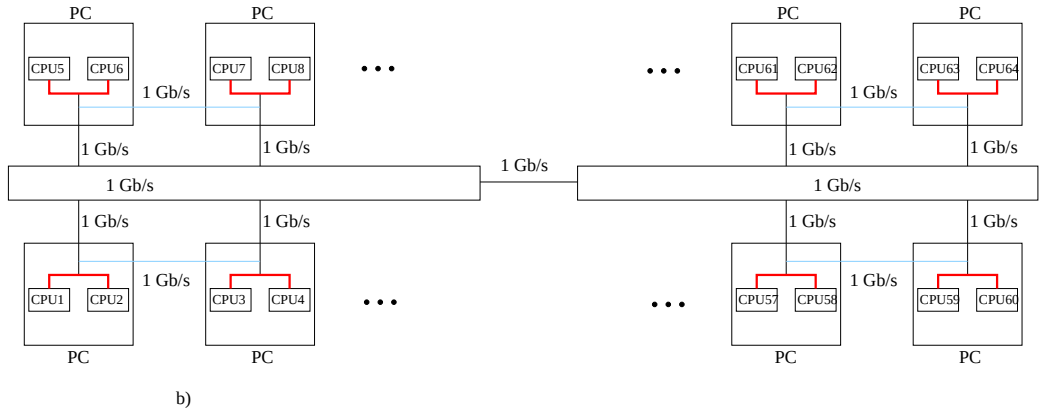
Gruča VRANA4 je sestavljena iz 64 računalnikov s procesorji AMD Athlon s hitrostjo 700 MHz. Povezani so s topologijo hiperkocke s točkovnimi povezavami Ethernet s hitrostjo 100 Mb/s.

VRANA5 in VRANA8

Gruča VRANA5 je sestavljena iz 16 dvoprocesorskih računalnikov s procesorji AMD Athlon MP-1600+. Gruča VRANA8 je sestavljena iz 32 dvoprocesorskih računalnikov s procesorji AMD Athlon MP-2200. Obe gruči sta povezani s topologijo hierarhične hiperkocke:

Hierarhična hiperkocka

Topologijo hierarhične hiperkocke sem implementiral na gruči VRANA8 z 32 dvoprocesorskimi osebnimi računalniki s skupno 64 procesorji. Povezavi prvih dveh dimenzij sta točkovni, ostale 4 dimenzije so implementirane s stikali. Povezave prve dimenzije so najhitrejše, saj zanje uporabimo vodilo vsakega dvoprocesorskega računalnika. Povezave druge dimenzije implementiramo s točkovnimi povezavami med pari računalnikov. Za povezave 3., 4. in 5. dimenzije uporabimo dve stikali, po eno za 16 računalnikov. Za povezave 6. dimenzije uporabimo obe stikali, vendar te povezave premostijo obe stikali, zato so najpočasnejše. Topologija implementirane gruče je prikazana na sliki 4.2. Topologija gruče VRANA5 je podobna opisani hierarhična hiperkocki, le da ima 5 dimenzij in eno stikalo Ethernet s hitrostjo 100 Mb/s.



Slika 4.2: Topologija hierarhične hiperkocke v gruči osebnih računalnikov VRANA8. Za prvo in drugo dimenzijo hiperkocke, po katerih prenašamo največ podatkov, uporabimo najhitrejši povezavi: vodilo in točkovno povezavo med parom računalnikov. Za ostale 3 uporabimo skupno stikalo, za 6. dimenzijo pa povezavo med dvema stikaloma.

VRANA6

Gruča VRANA6 je sestavljena iz 8 dvoprocesorskih računalnikov s procesorji AMD Athlon MP-1600+. Povezani so s stikalom Ethernet s hitrostjo 1 Gb/s.

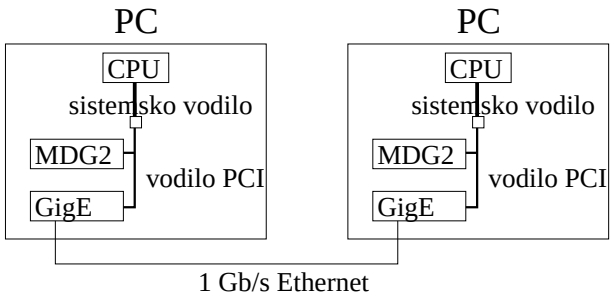
VRANA9

Gruča VRANA9 je sestavljena iz 64 dvoprocesorskih računalnikov s procesorji AMD Opteron 242. Povezani so s 3 stikali Ethernet s hitrostjo 1 Gb/s.

4.2 Namenska strojna oprema

Namenska strojna oprema je taka oprema, ki je povsem prilagojena izvajanju le nekaj določenih vrst računov. Primer namenske strojne opreme so procesorji grafičnih kartic, ki so namenjeni pohitritvi računanja in prikaza tridimenzionalne grafike. Specializirani so za izvajanje operacij linearne algebre in drugih operacij, ki se uporabljajo v računalniški grafiki, njihovi vhodni in izhodni podatki pa so strukturirani v obliki vektorjev in matrik [45, 51].

Namenska strojna oprema pohitri izvajanje računov, za katere je načrtovana in izdelana, v primerjavi s splošnimi procesorji.



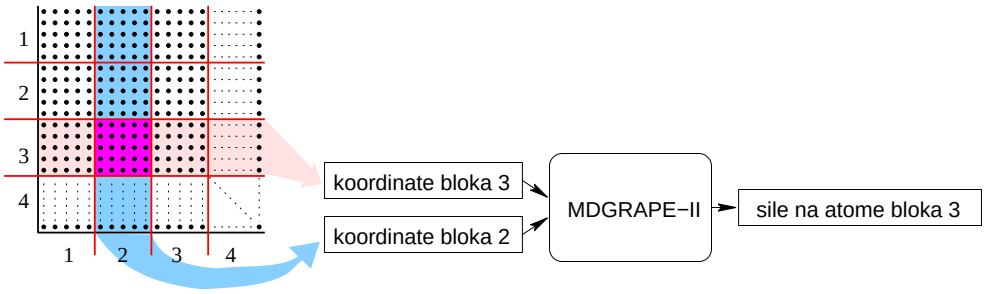
Slika 4.3: Shema namestitve dveh procesorjev MDGRAPE-II (označena MDG) v dveh ločenih osebnih računalnikih (označena PC). Procesorja uporabimo le za računanje neveznih interakcij; vse ostale računa procesor gostitelj (označen CPU). Procesor MDGRAPE-II in mrežni vmesnik Ethernet (označen GigE) sta povezana s procesorjem preko vodila PCI in sistemskega vodila. Komunikacija med računalnikoma poteka po neposredni točkovni povezavi preko mreže Ethernet s hitrostjo 1 Gb/s.

4.2.1 MDGRAPE-II

Procesor MDGRAPE-II(ang. MD Gravity Pipeline) je bil narejen za hitrejše računanje simulacije MD [52–54, 68]. Namenjen je hitremu računanju neveznih interakcij; ostale dele simulacije mora računati procesor gostitelj. V gruči VRANA7 smo procesorje uporabili vzporedno. Gruča z dvema procesorjema MDGRAPE-II je prikazana na sliki 4.3.

Vhodni podatki procesorju MDGRAPE-II so lastnosti atomov (vrsta vsakega atoma, ki določa npr. njegov naboj) in funkcija interakcije med njima, npr. sila van der Waalsove interakcije ali sila elektrostatske interakcije. Funkcija interakcije je podana tabelarično kot vrednost interakcije v odvisnosti od razdalje med atomoma. Na vsakem koraku simulacije procesorju znova prenesemo nove koordinate atomov. Procesor na podlagi shranjenih atomskih podatkov in podanih koordinat izračuna interakcije med njimi (npr. sile), kar je tudi izhod. Postopek računanja pri uporabi procesorja MDGRAPE-II je prikazan na sliki 4.4.

Procesor loči dve množici vhodnih atomov: ena je množica atomov, ki »povzročajo« interakcije (ang. force-asserting), ena pa je množica atomov, na katere »delujejo« interakcije (ang. force-receiving). Kot je razvidno iz slike 4.4, lahko MDGRAPE-II sklopimo z vzporedno metodo delitev sil: procesor računa interakcije med dvema blokoma [11]. Za upoštevanje tretjega Newtonovega zakona moramo računati tudi nasprotne sile, kar pomeni, da moramo računati celotno matriko sil,



Slika 4.4: Podatkovni tok pri uporabi namenskega procesorja MDGRAPE-II. Pripravimo koordinate dveh množic atomov (recimo bloka 2 in 3), med katerimi naj procesor računa interakcije. Procesor izračuna sile na prvi blok (na atome bloka 3), ki delujejo zaradi interakcij z drugim blokom (z blokom 2).

ne le polovično.

4.3 Komunikacijske zahteve nove metode DDFD

4.3.1 Prisotnost podatkov po procesorjih

Na vsakem procesorju morajo biti prisotni podatki, da lahko procesor izračuna potrebne interakcije, npr. sile. S porazdelitvijo atomov v bloke in na domače procesorje metoda delitev sil določi, kateri podatki morajo biti prisotni na katerih procesorjih.

Metoda delitev sil porazdeli atome v bloke s porazdelitvijo $\mathcal{B}$ (preslikava 2.3), vsakemu procesorju iz množice procesorjev P pa s preslikavo $\hat{\mathcal{B}}$ (preslikava 2.5) dodelimo en produkt dveh blokov. Delitev N atomov v bloke je enakomerna, torej vsak blok vsebuje $N/|B|$ atomov. Vsak izmed procesorjev $p \in P$ računa interakcije med atomi dveh blokov b_i in b_j, $(b_i, b_j) = \hat{\mathcal{B}}(p)$. Takih interakcij je največ $(N/|B|)^2$. Ob uporabi razdalje *cut-off* (opisano v podpoglavju 1.2.2) ni potrebno računati vseh interakcij. Procesor p mora torej imeti koordinate vseh atomov, ki pripadajo blokoma b_i in b_j, kar je skupaj $2N/|B|$ atomov. Poleg tega procesor p računa tudi nove koordinate za svoje domače atome A_p na podlagi izračunanih interakcij. Procesor ima le $N/|P|$ domačih atomov. Vsi atomi iz množice A_p so tudi v eni izmed množic b_i ali b_j, $(b_i, b_j) = \hat{\mathcal{B}}(p)$, zato so vsi podatki o atomih iz A_p že prisotni na procesorju p.

Vsak procesor mora dodatno imeti podatke tudi o tistih atomih, ki nastopajo v tri- ali več-atomskih veznih interakcijah. Pri takih interakcijah je lahko vsak atom iz različnega bloka. V tem primeru izberemo en procesor, ki računa tako interakcijo in določimo, da mora ta procesor dobiti podatke tudi o ostalih atomih – sirotah, ki so izven njegovih dveh blokov.

Glavni izsledki analize prisotnosti podatkov po procesorjih so naslednji:

1. vsak procesor ima podatke o dveh blokih s po $N/|B|$ atomi, skupaj $2N/|B|$ atomi;

2. procesor mora imeti zgoraj omenjene podatke za računanje interakcij med svojima dvema blokoma;

3. za nekatere izmed atomov, katerih podatke ima, procesor računa tudi spremembo koordinat na podlagi izračunanih in izmenjanih interakcij.

4.3.2 Izmenjava podatkov

Pri metodi delitev sil se vsa komunikacija vrši po blokih. Procesorji ne izmenjujejo podatkov s procesorji, s katerimi nimajo skupnega bloka. Nek procesor p, ki pripada blokoma b_i in b_j ($p \in \hat{P}_{b_i} \cap \hat{P}_{b_j}$), komunicira le s procesorji iz $\hat{P}_{b_i}$ in $\hat{P}_{b_j}$. S procesorji iz množice $\hat{P}_{b_i}$ izmenjuje podatke o atomih bloka b_i, s procesorji $\hat{P}_{b_j}$ pa izmenjuje podatke o atomih bloka b_j. Velikost vsake izmed množic $\hat{P}_{b_i}$ in $\hat{P}_{b_j}$ je $|B| - 1$, zato vsak procesor komunicira le z $2(|B| - 2)$ procesorji.

Izmenjava podatkov sirot tudi poteka le po blokih. Sirote procesorja p so tisti atomi, ki ne pripadajo dvema blokoma $(b_i, b_j) = \hat{\mathcal{B}}(p)$ nekega procesorja p, a jih mora procesor p imeti za pravilno računanje veznih interakcij treh in več atomov. Za vsak blok b_s, $b_s \neq b_i$, $b_s \neq b_j$, obstaja tak procesor $q \in P$, ki ima nek skupen blok s procesorjem $p \in P$, saj vedno velja

$$b_s = \hat{\mathcal{B}}(p) \cap \hat{\mathcal{B}}(q) \neq \emptyset \tag{4.1}$$

Torej vedno obstaja tak procesor q, ki za poljubno siroto $a_s \in b_s$ lahko pošlje podatek o siroti procesorju p.

Drugih vrst komunikacije pri simulaciji MD ni.

Iz analize izmenjave podatkov ugotovimo naslednji poglavitni komunikacijski značilnosti:

1. vsak procesor komunicira le z dvema podmnožicama po $|B| - 2$ procesorjev, torej z $2(|B| - 2)$ procesorji;

2. velikost bloka je $N/|B|$, torej procesor izmenjuje podatke o $2N/|B|$ atomih.

4.4 Računalnik za računanje po metodi DDFD

Na podlagi analize delitve podatkov, komunikacijskih vzorcev metode DDFD in dosegljive strojne opreme sem zasnoval namensko računalniško gručo za izvajanje simulacije MD z novo metodo delitev sil.

4.4.1 Zahteve računalnika

Opisal bom značilnosti, ki naj jih ima računalnik za računanje simulacije MD z metodo DDFD. Poimenujmo tak računalnik FDM po kratici angleški besed Force Decomposition Machine.

Vzporedna arhitektura

Glavna značilnost vzorca komuniciranja pri metodi DDFD je omejena množica drugih procesorjev, s katero vsak procesor komunicira. Zagotoviti moramo, da so vsi procesorji, ki pripadajo enemu bloku, povezani. Za računanje simulacije MD so druge povezave nepotrebne, saj ne pohitrijo komunikacije, zato tudi ne pohitrijo simulacije.

Ugotovitev. Odločimo se za arhitekturo vzporednega računalnika s porazdeljenim pomnilnikom. Gruča osebnih računalnikov ustreza taki arhitekturi.

Ugotovitev. Zaželena je uporaba enoprocesorskih računalnikov.

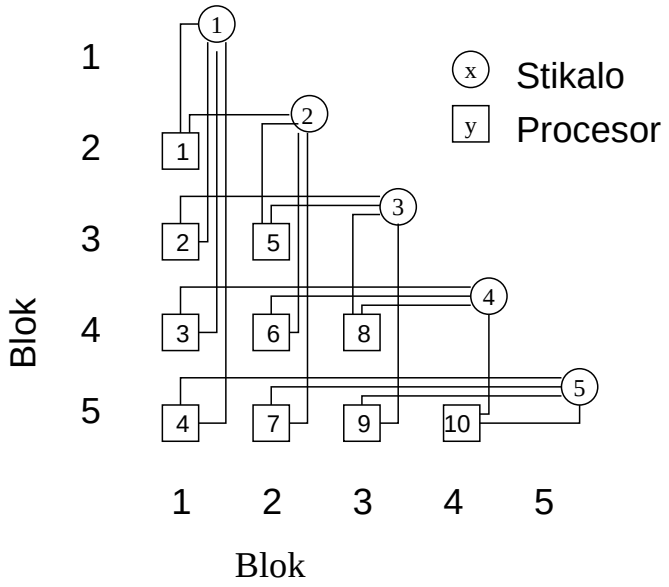
Prostorska zahtevnost

Vsak procesor mora imeti podatke o atomih dveh blokov, torej o $2N/|B|$ atomih. Pri vzporedni metodi repliciranih podatkov mora imeti vsak procesor podatke o vseh N atomih. Iz primerjave sledi, da če ima računalnik dovolj pomnilnika za izvajanje vzporednih simulacij z metodo repliciranih podatkov na nekem molekulskem sistemu, potem ima zagotovo dovolj pomnilnika za izvajanje vzporedne simulacije z metodo DDFD. Z večanjem števila procesorjev se prostorska zahtevnost nove metode celo manjša, zato lahko na enakem računalniku obravnavamo večje sisteme kot z metodo repliciranih podatkov.

Ugotovitev. Metoda delitev sil ne zahteva večjega pomnilnika od obstoječih metod.

4.4.2 Topologija gruče

Na podlagi komunikacijskih značilnosti, uvedenih v podpoglavju 4.3.2, predstavimo topologijo gruče osebnih računalnikov. Uporabimo enoprocesorske računalnike, ki so povezani s številnimi mrežnimi stikali. Za vsakega izmed $|B|$ blokov uporabimo svoje stikalo, na katerega priključimo vse procesorje, ki imajo podatke o atomih tega bloka. Torej s stikalom nekega bloka b povežemo vseh $|B| - 1$ računalnikov iz množice $\hat{P}_b$. Vsaka lokalna operacija (opisane v podpoglavju 2.4) nekega bloka poteka le po stikalu tega bloka. Topologija z lastnim mrežnim stikalom za vsak blok je prikazana na sliki 4.5. Za stikalo je mišljena tehnologija Ethernet, čeprav je možno uporabiti tudi druge, zmogljivejše, kot sta npr. Myrinet ali Infiniband.

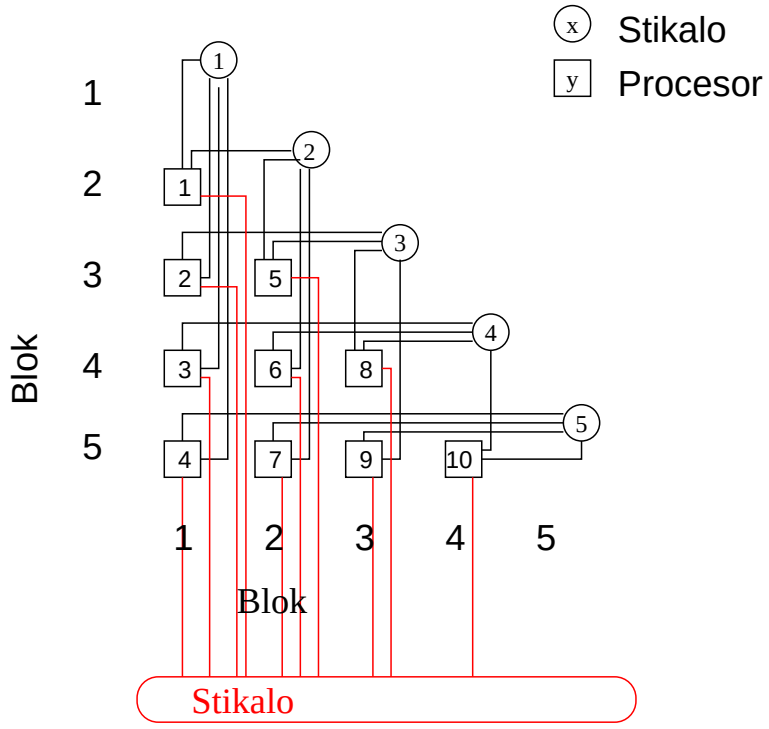


Slika 4.5: Topologije gruče, ki je prilagojena računanju vzporedne simulacije MD z metodo DDFD. Vsak blok ima svoje lastno mrežno stikalo, s katerim so povezani vsi procesorji bloka.

Vsakemu bloku priredimo svoje stikalo zato, ker je tako skupna pasovna širina (ang. bandwidth) stikal večja, kot je pasovna širina enega samega stikala. Namreč, stikalo ima neko največjo notranjo pasovno širino. Ta omejitev pomeni, da obstaja neko največje število parov računalnikov, ki lahko hkrati komunicirajo po nazivni pasovni širini. Če je n računalnikov povezanih s stikalom z neko hitrostjo s, npr. 1 Gb/s, potem lahko dva izmed n računalnikov komunicirata s hitrostjo s. Večina stikal z n priključki nima pasovne širine $ns/2$, zato $n/2$ računalnikov ne more komunicirati z ostalimi $n/2$ s hitrostjo s, temveč z manjšo hitrostjo [8, 34].

Pri porazdeljenih globalnih operacijah veliko parov računalnikov hkrati izmenjuje podatke. Uporaba le enega stikala bi omejevala pretok ter podaljšala čas izvedbe operacij.

Vsak računalnik ima po dva mrežna vmesnika, zato moramo na vsakem računalniku določiti vmesnik, preko katerega vrši oddajo sporočil drugim računalnikom. Usmerjevalne tabele v računalnikih moramo nastaviti tako, da računalnik p, $p \in \hat{P}_{b_i}, p \in \hat{P}_{b_j}$, posreduje sporočila, namenjena drugim računalnikom iz množice $\hat{P}_{b_i}$ preko vmesnika, s katerim je povezan s stikalom bloka b_i, sporočila, namenjena računalnikom iz $\hat{P}_{b_j}$, pa preko vmesnika, s katerim je računalnik povezan s stikalom bloka b_j. Če želimo le z opisanimi povezavami uvesti še splošnejše usmerjanje, tako da so vsi računalniki dosegljivi med sabo, potem moramo usmerjevalne tabele dopolniti. V tem primeru morajo biti računalniki sposobni tudi posredovati sporočila



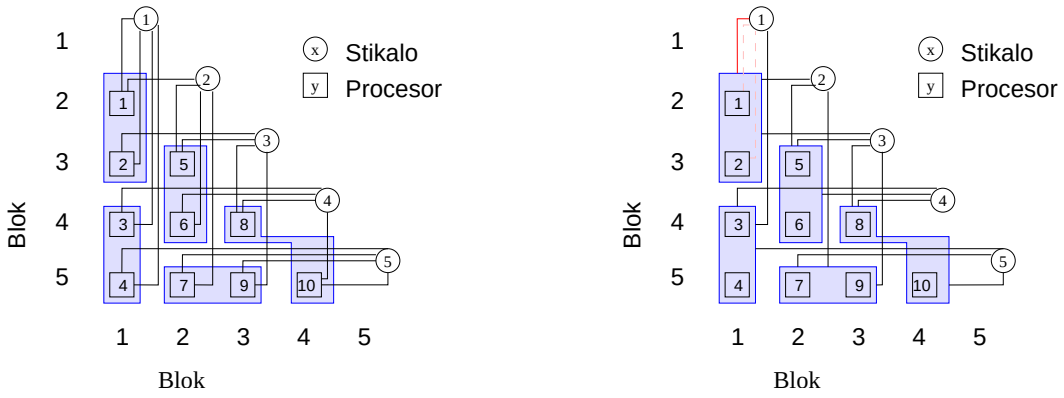
Slika 4.6: Uvedba splošnega stikala. Tako povečamo uporabnost gruče pri splošnih simulacijah. Novo stikalo mora imeti dovolj priključkov, da lahko nanj povežemo vse računalnike.

na poti od enega do drugega računalnika [70].

Za polno dosegljivosti med računalniki lahko uporabimo drugačno rešitev, ki hkrati veča uporabnost. Predlagam uporabo dodatnega, večjega mrežnega stikala, s katerim so povezani vsi računalniki. Stikalo mora imeti najmanj toliko priključkov, kolikor je računalnikov v gruči. Taka topologija z dodanim stikalom je prikazana na sliki 4.6. To stikalo se ne uporabi pri komunikaciji v globalnih operacijah.

Večprocesorski računalniki

Vsak procesor mora biti povezan z dvema drugima skupinama procesorjev. Če želimo uporabiti večprocesorski računalnik ugotovimo, da moramo število povezav v enem računalniku povečati z 2 na $2 - 1 + mp$, kjer je mp število procesorjev v enem računalniku. Primer takega združevanja pri dvoprocesorskem računalniku je prikazan na sliki 4.7a. Procesorje združujemo v računalnike tako, da pripadata skupnemu bloku, npr. procesorja p_1 in p_2 sta v istem računalniku, zato naj velja $p_1 \in \hat{P}_{b_1}$ in $p_2 \in \hat{P}_{b_1}$. Na sliki 4.7b so prikazane povezave med dvoprocesorskimi računalniki. V vseh računalnikih oba procesorja pripadata eni izmed množic $\hat{P}_b$,



Slika 4.7: Primer, kako dodelimo procesorje dvoprocesorskih računalnikov različnim produktom blokov (podslika a). Po dva procesorja sta združena v enem računalniku (predstavljeni z modrimi liki). Na podsliki b je je prikazano, kako si procesorja 1 in 2 delita isto povezavo k stikalu skupnega bloka 1: namesto dveh povezav (rdeči črtani črtai) uporabimo eno (povezana rdeča črta). Urnik prenosov moramo ustrezno spremeniti, da upoštevamo skupne povezave in kolokacijo procesorjev.

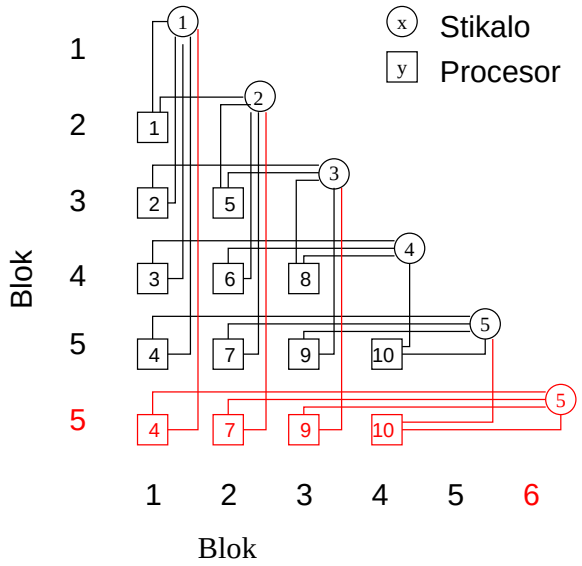
zato nadomestimo njuni dve povezavi z eno, kar zmanjša število vseh povezav. Urnik prenosov moramo ustrezno spremeniti, da upoštevamo spremenjeno topologijo, sicer lahko dva procesorja sočasno uporabljata eno povezavo, kar upočasni komunikacijo in poslabša vzporedno učinkovitost.

Poleg komunikacijskih ovir je pomembno tudi neskladje med številom procesorjev, ki jih potrebuje metoda, in številom vseh procesorjev v razpoložljivih računalnikih. Metoda lahko uporabi tudi liho število procesorjev (npr. 3, 15, ali 21); v tem primeru ostane en procesor neizkoriščen že pri dvoprocesorskih računalnikih. Pri računalnikih z večimi procesorji je neskladje še večje.

4.4.3 Možnosti razširitve

Ker ima vsak računalnik (v primeru enoprocesorskih računalnikov) v predlagani topologiji vedno le dve povezavi, lahko gručo razširimo z novim stikalom in novimi računalniki. Omejitev predstavljajo le mrežna stikala, ki imajo omejeno število priključkov. Če uporabimo stikalo z n priključnimi mesti, potem je lahko v enem bloku največ n računalnikov, iz česar sledi, da je število blokov omejeno na B_{max} blokov,

$$B_{max} = n + 1, \tag{4.2}$$



Slika 4.8: Razširitev gruče za en blok z dodatnimi 5 računalniki in enim stikalom. Dodani elementi so označeni rdeče.

kar omeji število računalnikov (oziroma procesorjev) v gruči na P_{max},

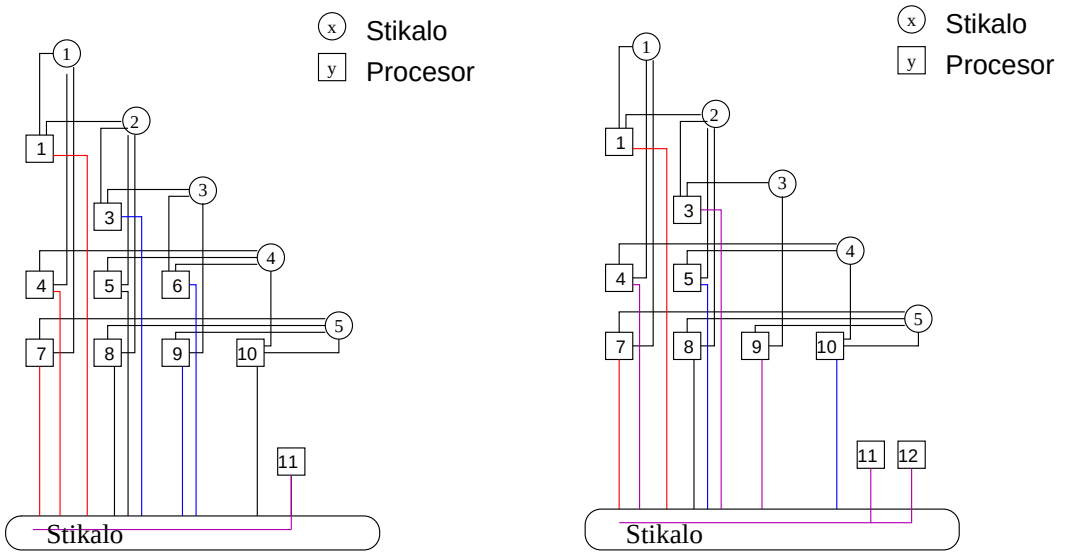
$$P_{max} = \frac{B_{max}(B_{max}-1)}{2} = \frac{n(n+1)}{2}. \tag{4.3}$$

Obstoječo gručo, ki temelji na $|B|$ blokih in ima $|B|(|B|-1)/2$ enoprocesorskih računalnikov in $|B|$ stikal, ki imajo najmanj $|B|$ priključkov za povezave z računalniki, lahko razširimo z dodatnim mrežnim stikalom in dodatnimi $|B|$ procesorji. Primer take razširitve gruče s 5 bloki in 10 računalniki na gručo s 6 bloki in 15 računalniki je prikazan na sliki 4.8. Dodano je eno stikalo in 5 računalnikov. Ob uporabi stikal z 8 priključki lahko gručo razširimo na največ 36 procesorjev.

4.4.4 Zanesljivost in redundančnost računalnikov

Pri načrtovanju gruče osebnih računalnikov moramo predvideti vpliv okvar in drugih vzrokov za izločitev računalnikov iz gruče. Neprekinjeno nadaljevanje začetih računov kljub okvari procesorjev je področje aktivnih raziskav, vendar take metode zahtevajo korenite posege v vzporedni program in uporabljene komunikacijske knjižnice [12, 26]. Pri načrtovanju gruče se zadovoljimo s ciljem, da po okvari sestavnega računalnika gručo na enostaven način usposobimo za ponovno uporabo.

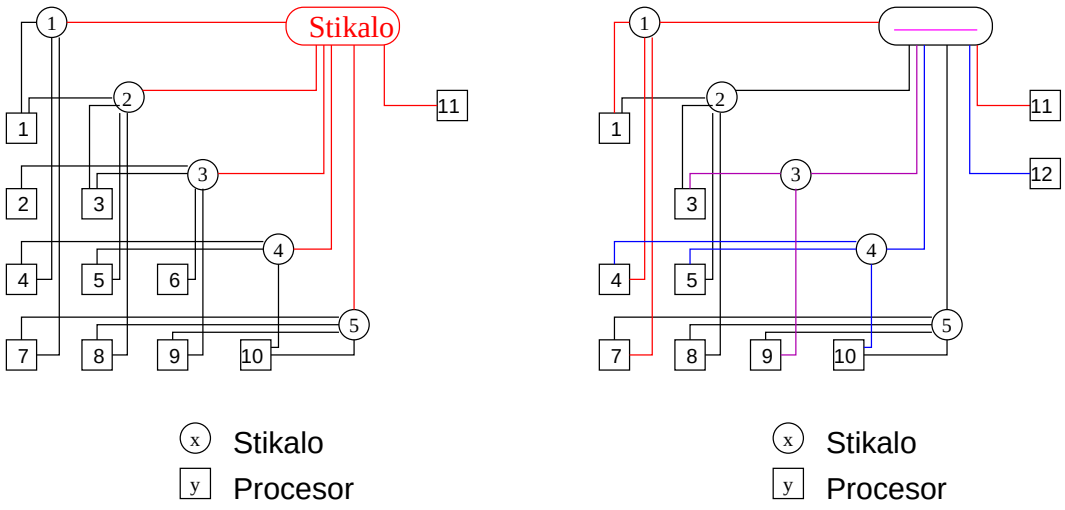
Ob okvari ali drugačni odstranitvi moramo nedelujoč računalnik nadomestiti. V ta namen moramo imeti pripravljen rezerven računalnik, ki je namenjen le zamenjavi nedelujočih računalnikov. Gručo in njeno topologijo načrtujemo tako, da je



Slika 4.9: Uporaba rezervnega računalnika 11 (zaradi uporabe enprocesorskih računalnikov enačimo računalnike in procesorje) na večjem stikalu ob okvari računalnika 2 je prikazana na podsliki a. Vsak računalnik ima dodatno povezavo na skupno stikalo. Ob izpadu računalnika 2 računalniki iz množice $\hat{P}_{b_1}$ bloka 1 $(1, 4, 7)$ uporabijo rdeče mrežne povezave, preostali računalniki iz množice $\hat{P}_{b_3}$ bloka 3 $(3, 6, 9)$ pa modre povezave. Na podsliki b je prikazana uporaba dveh rezervnih računalnikov, 11 in 12, ob izpadu računalnikov 2 in 6. Preostali računalniki $\hat{P}_{b_1} \cup \hat{P}_{b_3}$ blokov 1 in 3 namesto z računalnikom 2 komunicirajo z rezervnim računalnikom 11 po rdečih (ali vijoličnih) mrežnih povezavah, računalniki $\hat{P}_{b_3} \cup \hat{P}_{b_4}$ blokov 3 ter 4 pa namesto z računalnikom 6 komunicirajo z računalnikom 12 po modrih (ali viojoličnih) mrežnih povezavah. Povezave, kjer je promet za oba računalnika 11 in 12 skupen, so obarvane vijolično.

uporabna tudi do fizične zamenjave računalnika.

Z uporabo dodatnega stikala, s katerim so povezani vsi glavni in rezervni računalniki, lahko premostimo izpad enega ali večih računalnikov. V primeru okvare enega računalnika s procesorjem p računalnik nadomestimo z rezervnim računalnikom s procesorjem q, usmerjevalne tabele ostalih računalnikov s procesorji $\hat{P}_{b_i}$ in $\hat{P}_{b_j}$ iz dotičnih blokov $(b_i, b_j) = \hat{\mathcal{B}}(p)$ pa spremenimo, da namesto z izločenim računalnikov p komunicirajo z rezervnim računalnikom s procesorjem q preko skupnega mrežnega stikala. Primer premostitve okvare pri taki rešitvi je prikazan na sliki 4.9a. Računalnik 2 postane neuporaben, zato ga nadomestimo z računalnikom 11, kar vpliva tudi na ostale računalnike s procesorji iz množic $\hat{P}_{b_1}$ in $\hat{P}_{b_3}$ blokov b_1 in b_3. Na podoben način lahko povečamo redundančnost z večimi rezervnimi računalniki, kot je prikazano na sliki 4.9b.



Slika 4.10: Uvedba rezervnega računalnika (zaradi uporabe enprocesorskih računalnikov enačimo računalnike in procesorje) preko posrednega stikala je prikazana na podsliki a; novo stikalo, povezave in 2 rezervna računalnika so obarvani rdeče. Vsako stikalo ima novo povezavo do posrednega stikala, na katerega sta priključena rezervna računalnika. Uporaba rezervnih računalnikov 11 in 12 ob izpadu računalnikov 2 in 6 na podsliki b je prikazana na podsliki b. Računalniki $\hat{P}_{b_1} \cup \hat{P}_{b_3}$ blokov 1 in 3 komunicirajo z rezervnim računalnikom 11 po rdečih mrežnih povezavah, računalniki $\hat{P}_{b_1} \cup \hat{P}_{b_3}$ blokov 3 ter 4 pa komunicirajo z računalnikom 12 po modrih mrežnih povezavah. Povezave, po katerih poteka komunikacija z obema računalnikoma, so označene vijolično.

Redundančnost z uporabo večjega stikala sloni na predpostavki, da je tako stikalo enakovredno manjšim stikalom, ki jih uporabimo za povezovanje računalnikov po blokih. S tehnologijo Ethernet pri hitrostih 1 Gb/s je to možno do določene velikosti gruče (s stikalom z 48 priključki je največja velikost gruče 45 računalnikov s 3 rezervnimi), z drugimi pa ni nujno tako.

V takih primerih je možna rešitev povezava vseh manjših stikal, ki služijo prenosu enega bloka, z enim posrednim stikalom. S posrednim stikalom povežemo tudi rezervne računalnike. Taka izvedba je prikazana na sliki 4.10. Podobno kot pri prej opisanem pristopu moramo v primeru okvare spremeniti usmerjevalno tabelo. Slabost tega pristopa v primerjavi s prej opisanim pristopom je omejitev možne razširitve gruče, saj moramo za rezervne računalnike uporabiti po en priključek na vsakem izmed stikal. Pri hkratni uporabi več kot enega rezervnega računalnika postanejo mrežne povezave do rezervnega računalnika zasičene, če ustrezno ne spremenimo urnika prenosa za izvedbo globalnih operacij.

Poglavje 5

Uporaba računalniških simulacij

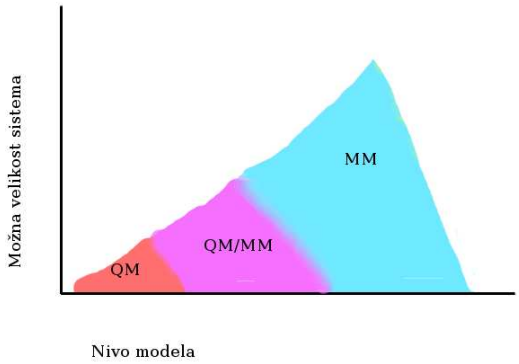
Predstavil bom študije nekaterih sistemov, ki smo jih izvajali s pomočjo modeliranja in simulacij MD. Pri vseh študijah smo uporabili vzporedne računalniške metode repliciranih podatkov in delitev sil, zaradi česar smo hitreje dobili končne rezultate simulacij, to so trajektorije gibanj, iz katerih z analizo dobimo lastnosti simuliranega sistema, npr. strukturo, spektre in aktivnost.

Za izvajanje računalniških simulacij sem uporabil gruče VRANA5, VRANA6, VRANA7 ter VRANA9. Za simulacije največjih sistemov z nekaj stotisoč atomov sem uporabil gručo VRANA7, saj je edino procesorj MDGRAPE-II omogočal simulacijo MD tako obsežnih sistemov. Za te simulacije sem uporabil metodo delitev sil. Ostale simulacije sem izvajal z vzporedno metodo repliciranih podatkov na 4–16 procesorjih, saj sem hkrati izvajal več simulacij različnih sistemov.

5.1 Molekulski modeli

Poznamo več vrst molekulskih modelov, ki jih uporabimo v simulacijah in ki se razlikujejo po natančnosti oz. nivoju opisa sistema:

Kvantna mehanika (QM) Molekulski sistem je opisan na kvantnem nivoju, kar je najbolj natančen opis atomov [44]. Kot celoto modeliramo jedro atomov in, ločeno od jeder, posamezne elektrone. Interakcije med elektroni opisujemo z osnovnimi kvantno-mehanskimi (QM, po angl. quantum mechanics) enačbami. Simulacije na kvantno-mehanskem nivoju so računsko zelo zahtevne in računska zahtevnost simulacije narašča eksponentno s številom atomov [44]. Simulacije so praktično izvedljive za sisteme z največ nekaj sto atomi [10]. Med QM štejemo tudi teorijo gostotnih funkcionalov osnovano na Kohn-Shamovi teoriji [43] (ang. density function theory, DFT), pri kateri namesto računanja



Slika 5.1: Omejitve velikosti sistema glede na nivo modela (QM: kvanta mehanika; QM/MM: kvantna mehanika/molekulska mehanika; MM: molekulska mehanika). Modeli višjega nivoja so manj natančni in imajo manjšo računsko zahtevnosti in zato omogočajo obravnavo večjih sistemov od modelov nižjega nivoja.

uporabimo empirične vrednosti določenih interakcij.

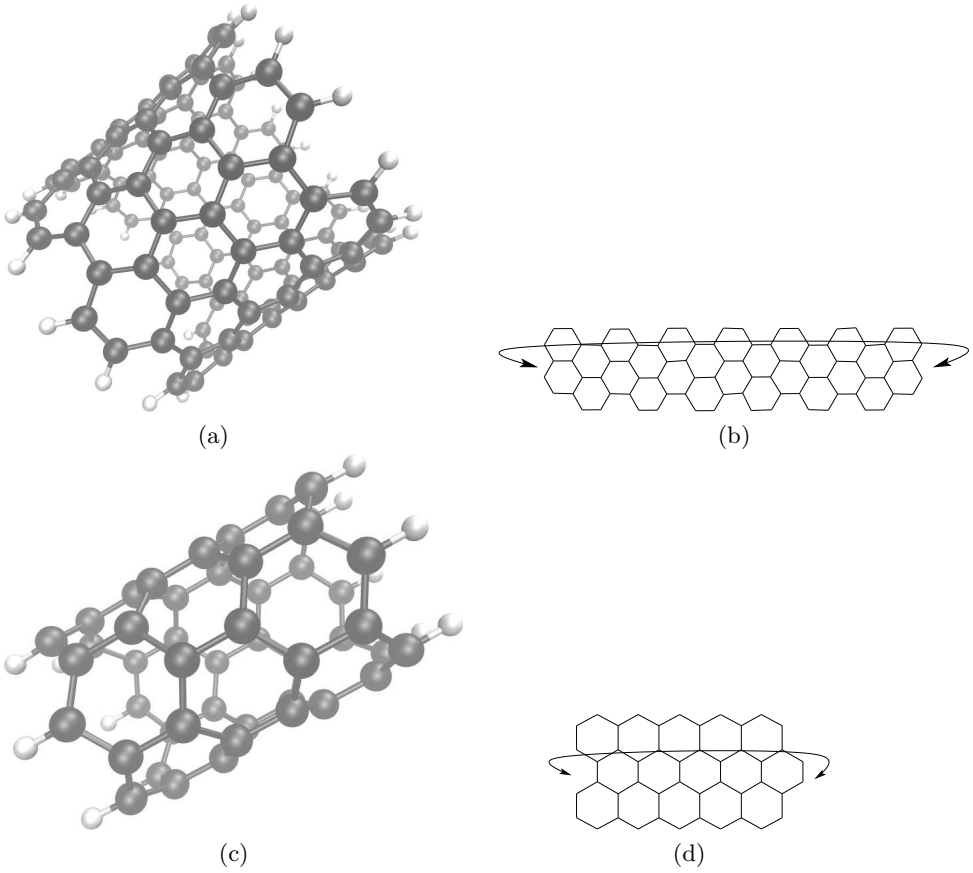
Klasična mehanika Molekulski sistem je opisan na nivoju klasične mehanike, kjer ločimo posamezne vrste atomov. Pri interakcijah med njimi nas zanima energija (molekulska mehanika, MM) ali sile (MD), kar računamo glede na različne vrste interakcij. Simulacije so računsko zahtevne, vendar je možno z MM simulirati tudi velike sisteme z več stotisočimi atomi [28].

Kvanta mehanika/molekulska mehanika QM/MM Kombinacija QM in MM, kjer je del molekulskega sistema opisan na kvantnem nivoju, preostali pa na klasičnem nivoju [25]. Model QM/MM omogoča simulacije večjih sistemov, pri katerih moramo uporabiti QM za obravnavo nekih atomov, njihovo okolico pa lahko obravnavamo klasično.

Na sliki 5.1 so prikazane relacije med največjimi sistemi, ki jih lahko obravnavamo z opisanimi modeli. Z manjšanjem nivoja modela – in večanjem natančnosti modela – se veča računska zahtevnost, zato je možno obravnavati manjše sisteme kot pri višjem nivoju.

5.2 Kvanta mehanika (QM)

Elektronska struktura in lastnosti ogljikovih nanocevk. Nanocevke so ena izmed treh vrst čistih oblik ogljika in so predmet aktivnih raziskov na področju nanotehnologije. Z računi na nivoju DFT smo želeli preučiti elektronsko strukturo



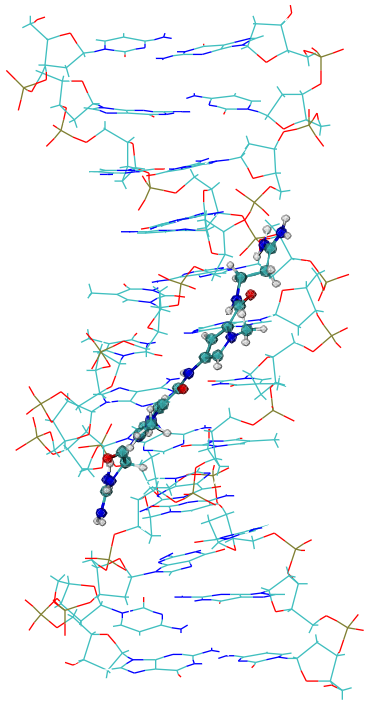
Slika 5.2: Dve vrsti obravnavanih nanocevk. Na podsliki a je predstavljena nanocevka vrste »stol« (ang. armchair) z dodanimi končnimi vodiki. Na podsliki b pa je prikazan izsek grafita, iz katerega je zvita prejšnja nanocevka, kot nakazuje puščica. Na podsliki c je prikazana »cik-cak« (ang. zig-zag) nanocevka z dodanimi končnimi vodiki; njeno zvitje iz izseka grafita pa je prikazan na podsliki d.

in lastnosti dveh vrst nanocevk različnih dolžin. Obe vrsti nanocevk, »cik-cak« in »stol«, sta prikazani na sliki 5.2.

Potrdili smo domneve, da imajo nanocevke vrste »stol« bolj kovinski značaj in so boljši prevodniki od nanocevk vrste »cik-cak«. Poleg tega smo ugotovili, da so »cik-cak« nanocevke polprevodniki ter da se prevodnost vseh nanocevk zmanjšuje z večanjem dolžine [10].

5.3 Kvanta mehanika/molekulska mehanika (QM/MM)

Pomembnost vodikovih vezi in van der Waalsovih interakcij za stabilnost vezave netropsina v ožji kanal DNK. Preučevali smo vezavo različnih konformacij liganda netropsina v ožji kanal deoksiribonukleinske kisline (DNK), kot je

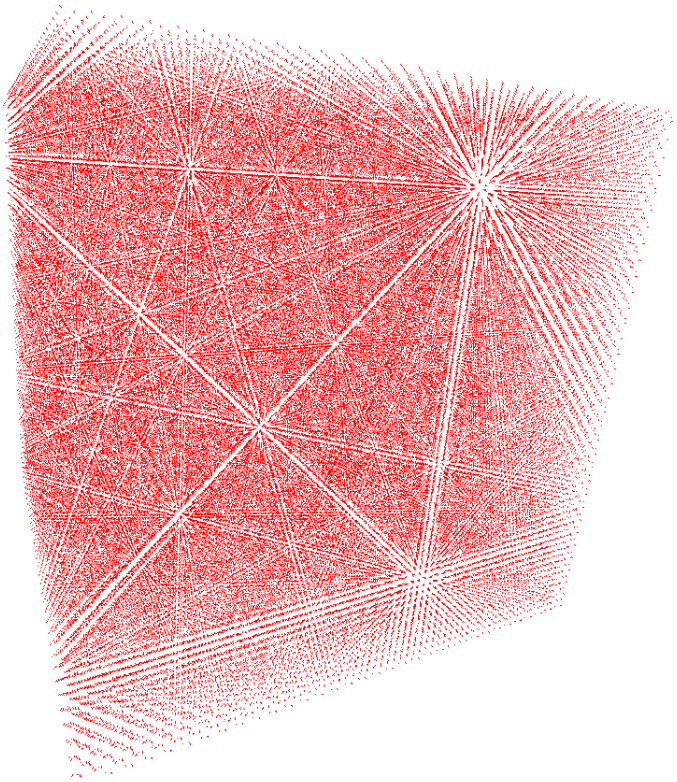


Slika 5.3: Vezava netropsina v ožji kanal DNK. Atomi in vezi DNK so predstavljeni s črtami, osnovna konformera netropsina pa z večjimi atomi in poudarjenimi vezmi.

prikazano na sliki 5.3. Nevezani netropsin je lahko v obliki ene izmed treh različnih konformer. Energija konformere vezane z DNK določa stabilnost vezave; nižja energija pomeni večjo stabilnost. Želeli smo ugotoviti stabilnost vezav različnih konformer v ožji kanal DNK.

Sistem smo računali na nivoju QM/MM: atome netropsina smo obravnavali kvantnomehansko, preostale atome (DNK in molekul vode) pa smo obravnavali klasično. Izvajali smo energijsko minimizacijo energije celotnega sistema.

Osnovna konformera je najbolj stabilna v vezavi z DNK, kar se ujema s poskusi. Ugotovili smo tudi, da je stabilnost vezave najbolj odvisna od van der Waalsovih interakcij, celo bolj kot od močnejših, a manj številnih, vodikovih vezi [24].



Slika 5.4: Sistem z 870.000 atomi vod.

5.4 Simulacija MD

Z atomističnimi simulacijami molekul vode na nivoju klasične mehanike smo proučevali lastnosti vode. Zanimali so nas vibracijski spekter in druge lastnosti vode. Novo razvite vzporedne metode in namenska strojna oprema za simulacije MD so nam omogočili obravnavo zelo velikih sistemov z 870.000 atomi. Simulirani sistem vod je prikazan na sliki 5.4 [9, 11].

Poglavje 6

Sklep

Računalniške simulacije so ključno orodje pri znanstvenih raziskavah. Simulacije molekulske dinamike (MD) nam omogočajo razumeti sisteme in nam omogočajo vpogled v sistem, ki je nedosegljiv z navadnimi poskusi na realnih sistemih. Računalniške simulacije so računsko zahtevne, zato iščemo načine, da bi pohitrili njihovo izvajanje.

Vzporedne računalniške simulacije porazdelijo računanje po večih procesorjih vzporednega računalnika. Tako nam omogočajo hitrejšo pot do želenih rezultatov, kar dodatno poveča uporabnost simulacij. Več različnih metod obstaja za vzporedno računanje simulacij MD.

Novo razvita metoda delitev sil s porazdelitvijo diagonale (DDFD) je metoda za vzporedno računanje simulacije MD. Metoda je splošno uporabna za študij najrazličnejših molekulskih sistemov. Omogoča izkoriščanje večjega števila procesorjev, saj se komunikacijska zahtevnost manjša z večanjem števila procesorjev. Z dinamičnim uravnoteženjem računskega bremena procesorjev zagotovimo veliko vzporedno učinkovitost, s tem pa izkoriščenost vzporednega računalnika. Zato je metoda uporabna tudi za pogoste simulacije, kjer uporabimo razdaljo *cut-off*, ki tudi zmanjša računsko zahtevnost simulacije.

Gruče osebnih računalnikov so primerni vzporedni računalniki za izvajanje računalniške simulacije. Z načrtovanjem topologije računalnika po komunikacijskih zahtevah programske opreme povečamo komunikacijsko hitrost med procesorji, kar tudi poveča učinkovitost vzporednega računanja.

Prispevki k znanosti

V okviru svojega doktorskega dela sem dosegel naslednje prispevke k znanosti:

- **Vzporedni računalniki (pregled področja).** Opisal sem obstoječe vzpore-

dne računalnike, ki jih uporabljamo za računalniške simulacije. Predstavil sem gruče osebnih računalnikov in načine, s katerimi jih povezujemo. Ker imajo take gruče porazdeljen pomnilnik, sem pokazal, kako dosežemo izmenjavo podatkov s prenašanjem sporočil med posamezni procesorji.

- **Simulacija MD (pregled področja).** Opisal sem ozadje in motivacijo za računanje simulacij MD. Predstavil sem metode za računanje tovrstnih simulacij in potek računanja.

- **Vzporedne simulacijske metode (pregled področja).** Predstavil sem vzporedno računanje s poudarkom na računalniških simulacijah. Predstavil sem, kako komunikacija omejuje učinkovitost vzporednega računanja in načine za merjenje učinkovitosti vzporednega računanja.

- **Vzporedne simulacije MD (pregled področja).** Opisal sem značilnosti obstoječih vzporednih metod in algoritmov, ki se uporabljajo za simulacijo MD. Opisal sem njihove razlike, predvsem časovne in prostorske zahtevnosti računanja in komunikacijske zahteve.

- **Razvoj nove metode DDFD za simulacijo MD.** Predstavil sem novo metodo za vzporedno računanje MD. Metoda, ki sem jo razvil, temelji na delitvi sil in ima nov pristop k deljenju računanja na procesorje vzporednega računalnika. Analiziral sem računsko zahtevnost metode in njene komunikacijske zahteve.

- **Razvoj algoritma za uravnoteženje računanja pri novi metodi DDFD.** Predstavil sem nov algoritem, ki dinamično uravnoteži računanje, s tem pa poveča vzporedno učinkovitost. Algoritem premešča posamezne manjše račune med procesorji med izvajanjem simulacije MD, tako da so procesorji čimbolj enakomerno obremenjeni. Pokazal sem, kdaj ima moj pristop prednost pred nespremenljivim obremenjevanjem procesorjev.

- **Primerjava nove vzporedne metode DDFD s standardnimi vzporednimi metodami za simulacije MD.** Primerjal sem novo razvito metodo z obstoječimi metodami za simulacijo MD.

- **Razvoj gruče osebnih računalnikov.** Analiziral sem zahteve vzporednega računalnika s porazdeljenim pomnilnikom za simulacijo MD. Zasnoval sem takšno topologijo povezav med procesorji, ki zmanjša čas, potreben za komunikacijo pri novi metodi delitev sil.

- **Uporaba namenske strojne opreme.** Predstavil sem namensko strojno opremo MDGRAPE-II za hitro računanje simulacije MD. Pokazal sem, kako sem izkoristil več procesorjev MDGRAPE-II skupaj z vzporedno metodo delitev sil za pohitritev vzporednih simulacij.

- **Uporaba vzporednih metod in gruč osebnih računalnikov za simulacijo veliki sistemov.** Novo razvite vzporedne metode na novi gruči sem uporabil za simulacije fizikalno, kemijsko in biološko zanimivih velikih sistemov, ki so preobsežni za obravnavo s standardnimi metodami za simulacijo MD in enoprocesorskimi računalniki.

Prihodnje delo

Opisano delo bom v prihodnost še razširil. Izpostavljam naslednja pričakovanja:

- Združiti metodo DDFD z metodami za obravnavo periodičnih sistemov. Pri simulacijah pogosto uporabimo periodičnost simulacijskega prostora, kjer sistem obravnavamo kot kristal. Za periodične pogoje obstajajo različni pristopi za računanje elektrostatskih interakcij. Te pristope bom združil z metodo delitev sil. S tem bo metoda postala uporabna za še več vrst simulacij.

- Povečati izkoriščanje večprocesorskih računalnikov. Trend razvoja osebnih računalnikov je uporaba večih procesorjev. Že današnji štiri-jedrni, dvoprocesorski računalnik ima drevesno hierarhijo procesorjev na treh nivojih. Take arhitekture bom izkoristil pri razporejanju procesorjev, tako da bo komunikacijski čas pri vzporedni simulaciji še manjši.

Literatura

[1] Almasi, G. S. in Gottlieb, A.: *Highly Parallel Computing*. Benjamin-Cummings Publishing Co., Inc., Redwood City, CA, 1989.

[2] Bagla, J.: TreePM: A code for cosmological N-body simulations. *J. Astrophys. Astron.*, 23:185–196, 2002.

[3] Barnes, J. in Hut, P.: A hierarchical O(N log N) force-calculation algorithm. *Nature*, 324(4):446–449, December 1986.

[4] Blackston, D. in Suel, T.: Highly portable and efficient implementations of parallel adaptive N-body methods. V *Proceedings of SuperComputing '97*, stran 4. ACM/IEEE SC'97 Conference, November 1997.

[5] Board, J. in Schulten, L.: The fast multipole algorithm. *Comput. Sci. Eng.*, 2(1):76–79, 2000.

[6] Board, J. A., Jr., Humphres, C. W., Lambert, C. G., Rankin, W. T. in Toukmaji, A. Y.: Ewald and multipole methods for periodic N-body problems. V Deuflhard, P. in dr., urednika, *Lecture Notes in Computational Science and Engineering*. Springer-Verlag, 1998.

[7] Boden, N. J., Cohen, D., Felderman, R. E., Kulawik, A. E., Seitz, C. L., Seizovic, J. N. in Su, W.-K.: Myrinet: A gigabit-per-second local area network. *IEEE Micro*, 15(1):29–36, 1995.

[8] Borštnik, U., Hodošček, M. in Janežič, D.: Improving the performance of molecular dynamics simulations on parallel clusters. *J. Chem. Inf. Comput. Sci.*, 44(2):359–364, 2004.

[9] Borštnik, U., Hodošček, M. in Janežič, D.: Fast parallel molecular simulations. *Croat. Chem. Acta*, 78(2):211–216, 2005.

[10] Borštnik, U., Hodošček, M., Janežič, D. in Lukovits, I.: Electronic structure of zig-zag and armchair nanotubes obtained by density functional calculations. *Chem. Phys. Lett.*, 411(4–6):384–388, 2005.

[11] Borštnik, U. in Janežič, D.: Symplectic molecular dynamics simulations on specially designed parallel computers. *J. Chem. Inf. Model.*, 45(6):1600–1604, 2005.

[12] Bosilca, G., Bouteiller, A., Cappello, F., Djilali, S., Fedak, G., Germain, C., Herault, T., Lemarinier, P., Lodygensky, O., Magniette, F. in dr.: MPICH-V: Toward a scalable fault tolerant MPI for volatile nodes. V *Proceedings of the Supercomputing 2002 Conference*, strani 29–29. ACM/IEEE, 2002.

[13] Bowers, K., Dror, R. in Shaw, D.: Overview of neutral territory methods for the parallel evaluation of pairwise particle interactions. *J. Phys. Conf. Ser.*, 16:300–304, 2005.

[14] Bowers, K., Dror, R. in Shaw, D.: The midpoint method for parallelization of particle simulations. *J. Chem. Phys.*, 124(18):184109–184109, 2006.

[15] Brooks, B. R., Bruccoleri, R. E., Olafson, B. D., States, D. J., Swaminathan, S. in Karplus, M.: CHARMM: A program for macromolecular energy, minimization, and dynamics calculations. *Journal of Computational Chemistry*, 4(2):187–217, 1983.

[16] Brooks, B. R. in Hodošček, M.: Parallelization of CHARMm for MIMD machines. *Chemical Design Automation News*, 7:16–22, 1992.

[17] Brooks, B. R., Janežič, D. in Karplus, M.: Harmonic analysis of large systems. I. Methodology. *J. Comp. Chem.*, 16(12):1522–1542, 1995.

[18] Burns, G., Daoud, R. in Vaigl, J.: LAM: An open cluster environment for MPI. V *Proceedings of Supercomputing Symposium*, volume 94, strani 379–386, 1994.

[19] A. Gara by, Blumrich, M. A., Chen, D., Chiu, G. L.-T., Coteus, P., Giampapa, M. E., Haring, R. A., Heidelberger, P., Hoenicke, D., Kopcsay, G. V., Liebsch, T. A., Ohmacht, M., Steinmacher-Burow, B. D., Takken, T. in Vranas, P.: Overview of the Blue Gene/L system architecture. *IBM J. Res. Devel.*, 49(2/3):195–212, 2005.

[20] Coteus, P., Bickford, H. R., Cipolla, T. M., Crumley, P. G., Gara, A., Hall, S. A., Kopcsay, G. V., Lanzetta, A. P., Mok, L. S., Rand, R., Swetz, R., Takken, T.,

Rocca, P. L., Marroquin, C., Germann, P. R. in Jeanson, M. J.: Packaging the Blue Gene/L supercomputer. *IBM J. Res. Devel.*, 49(2/3):265–276, 2005.

[21] Deserno, M. in Holm, C.: How to mesh up Ewald sums (I): A theoretical and numerical comparison of various particle mesh routines. *Arxiv preprint cond-mat/9807099*, 1998.

[22] Dietz, H. G. in T.I.Mattox: KLAT2's flat neighborhood network. V *Extreme Linux track of the 4th Annual Linux Showcase*, October 2000.

[23] Dobravec, T., Robič, B. in Vilfan, B.: Izmenjava podatkov v računalniških omrežjih. *Elektroteh. vest.*, 68:105–108, 2001.

[24] Dolenc, J., Borštnik, U., Hodošček, M., Koller, J. in Janežič, D.: An ab initio QM/MM study of the conformational stability of complexes formed by netropsin and DNA. The importance of van der waals interactions and hydrogen bonding. *J. Mol. Struct., Theochem*, 718:77–85, 2005.

[25] Eurenius, K. P., Chatfield, D. C., Brooks, B. R. in Hodošček., M.: Enzyme mechanisms with hybrid quantum and molecular mechanical potentials. I. Theoretical considerations. *Int. J. Quant. Chem.*, 60:1189–1200, 1996.

[26] Fagg, G. in Dongarra, J. : *FT-MPI: Fault tolerant MPI, supporting dynamic applications in a dynamic world*, volume 1908, strani 346–354. Springer-Verlag, Berlin, 2000.

[27] Feller, S., Pastor, R., Rojnuckarin, A., Bogusz, S. in Brooks, B.: Effect of electrostatic force truncation on interfacial and transport properties of water. *J. Phys. Chem.*, 100:17011–17020, 1996.

[28] Freddolino, P. L., Arkhipov, A. S., Larson, S. B., McPherson, A. in Schulten, K.: Molecular dynamics simulations of the complete satellite tobacco mosaic virus. *Structure*, 14:437–449, 2006.

[29] Frenkel, D. in Smit, B.: *Understandig Molecular Simulation: From Algorithms to Applications*. Academic Press, San Diego, 2002.

[30] Gropp, W., Lusk, E., Doss, N. in Skjellum, A.: A high-performance, portable implementation of the MPI message passing interface standard. *Parall. Comput.*, 22(6):789–828, 1996.

[31] Habata, S., Yokokawa, M. in Kitawaki, S.: The earth simulator system. *NEC Res. Devel.*, 44(1):21–26, 2003.

[32] Heermann, D. W. in Burkitt, A. N.: *Parallel Algorithms in Computational Science*. Springer-Verlag, Berlin, 1991.

[33] Intel. http://www.intel.com/.

[34] Jain, R.: *The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling*. John Wiley & Sons, New York, 1991.

[35] Janezic, D. in Orel, B.: Implicit runge-kutta method for molecular dynamics integration. *J. Chem. Inf. Comput. Sci.*, 33(2):252–257, 1993.

[36] Janezic, D. in Orel, B.: Improvement of methods for molecular dynamics integration. *Int. J. Quant. Chem.*, 51:407–415, 1994.

[37] Janežič, D. in Brooks, B. R.: Harmonic analysis of large systems. II. Comparison of different protein models. *J. Comp. Chem.*, 16(12):1543–1553, 1995.

[38] Janežič, D., Hodošček, M. in Ugi, I.: The simultaneous [alpha]-addition of a cation and an anion onto an isocyanide. *Internet Electron. J. Mol. Des.*, 1(6):293–299, 2002.

[39] Janežič, D., Praprotnik, M. in Merzel, F.: Molecular dynamics integration and molecular vibrational theory. I. New symplectic integrators. *J. Chem. Phys.*, 122:art. no. 174101, 2005.

[40] Janežič, D., Venable, R. M. in Brooks, B. R.: Harmonic analysis of large systems. III. Comparison with molecular dynamics. *J. Comp. Chem.*, 16(12):1554–1566, 1995.

[41] Jorgensen, W. L.: The many roles of computation in drug discovery. *Science*, 303(5665):1813–1818, 2004.

[42] Kodek, D.: *Arhitektura računalniških sistemov*. Bi-tim, Ljubljana, 1994.

[43] Kohn, W. in Sham, L. J.: Self-consistent equations including exchange and correlation effects. *Phys. Rev.*, 140:A1133, 1965.

[44] Koller, J.: *Struktura atomov in molekul: osnove kvantne mehanike, atomi*. Fakulteta za kemijo in kemijsko tehnologijo, Ljubljana, 1992.

[45] Krueger, J. in Westermann, R.: Linear algebra operators for GPU implementation of numerical algorithms. *ACM Trans. Graphics*, 22(3):908–916, 2003.

[46] Larson, S. M., Snow, C. D., Shirts, M. R. in Pande, V. S.: Folding@Home and Genome@Home: Using distributed computing to tackle previously intractable problems in computational biology. V Grant, R., urednik, *Comp. Genomics*. Horizon Press, 2002. To appear.

[47] Leach, A. R.: *Molecular Modeling: Principles and Applications*. Addison Wesley Longman Limited, Essex, 1996.

[48] Liu, J., Wu, J. in Panda, D. K.: High performance RDMA-based MPI implementation over InfiniBand. *Int. J. Parallel Programming*, 32(3):167–198, June 2004.

[49] Loncharich, R. in Brooks, B.: The effects of truncating long-range forces on protein dynamics. *Proteins: Struct. Funct. Genet*, 6:32–45, 1989.

[50] Mattson, T. G.: Parallel computing. V Mattson, T. G., urednik, *Parallel Computing in Computation Chemistry*, strani 1–15. American Chemical Society, 1995.

[51] Moreland, K. in Angel, E.: The FFT on a GPU. V *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*. ACM, 2003.

[52] Narumi, T.: *Special-purpose computer for molecular dynamics simulations*. Doctor's thesis, University of Tokyo, 1998.

[53] Narumi, T., Kawai, A. in Koishi, T.: An 8.61 Tflop/s molecular dynamics simulation for NaCl with a special-purpose computer: MDM. V *Proceedings of SuperComputing 2001*, Denver, 2001. ACM.

[54] Narumi, T., Susukita, R., Ebisuzaki, T., McNiven, G. in Elmegreen, B.: Molecular dynamics machine: Special-purpose computer for molecular dynamics simulations. *Mol. Sim.*, 21:401–415, 1999.

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

[56] Plimpton, S. J.: Fast parallel algorithms for short-range molecular dynamics. *J. Chem. Phys.*, 117(1):1–19, 1995.

[57] Plimpton, S. J. in Hendrickson, B. A.: A new parallel method for molecular-dynamics simulation of macromolecular systems. *J. Comp. Chem.*, 17:326–337, 1996.

[58] Plimpton, S. in Hendrickson, B.: Parallel molecular dynamics algorithms for simulation of molecular systems. V Mattson, T. G., urednik, *Parallel Computing in Computational Chemistry*, strani 114–132. American Chemical Society, 1995.

[59] Praprotnik, M. in Janežič, D.: Molecular dynamics integration and molecular vibrational theory. II. Simulation of non-linear molecules. *J. Chem. Phys.*, 122:art. no. 174102, 2005.

[60] Praprotnik, M. in Janežič, D.: Molecular dynamics integration and molecular vibrational theory. III. The IR spectrum of water. *J. Chem. Phys.*, 122:art. no. 174103, 2005.

[61] Reschke, C., Sterling, T., Ridge, D., Savarese, D., Becker, D. J. in Merkey, P.: A design study of alternative network topologies for the beowulf parallel workstation. V *Proceedings of the Fifth IEEE International Symposium on High Performance Distributed Computing*, strani 626–636. IEEE, 1996.

[62] Shu, J. W., Wang, B., Chen, M., Wang, J. Z. in Zheng, W. M.: Optimization techniques for parallel force-decomposition algorithm in molecular dynamic simulations. *Comput. Phys. Comm.*, 154(2):121–130, August 2003.

[63] Snir, M.: A note on n-body computations with cutoffs. *Theory Comput. Systems*, 37:295–318, 2004.

[64] Snir, M.: A note on N-body computation with cutoffs. Technical report, IBM T. J. Watson Research Center, 2001.

[65] Spector, D. H. M.: *Building Linux Clusters: Scaling Linux for Scientific and Enterprise Applications*. O'Reilly & Associates, Sebastopol, CA, 2000.

[66] Sterling, T., Becker, D. J. in Savarese, D.: Beowulf: A parallel workstation for scientific computation. V *Proceedings, 24th International Conference on Parallel Processing*, volume 1, strani 11–14, 1995.

[67] Sunderam, V. S.: PVM: A framework for parallel distributed computing. *Concurrency: practice and experience*, 2(4):315–339, 1990.

[68] Taiji, M., Narumi, T., Ohno, Y., Futatsugi, N., Suenaga, A., Takada, N. in Konagaya, A.: Protein explorer: A Petaflops special-purpose computer system for molecular dynamics simulations. V *Proceedings of SuperComputing 2003*, Phoenix, 2003. ACM.

[69] Toukmaji, A. Y. in Jr., J. A. B.: Ewald summation techniques in perspective: A survey. *Comput. Phys. Comm.*, 95:73–92, 1996.

[70] Trobec, R.: Two-dimensional regular d-meshes. *Parall. Comput.*, 26(13):1945–1953, December 2000.

[71] Trobec, R., Jerebic, I. in Janežič, D.: Parallel algorithm for molecular dynamics integration. *Parallel Computing*, 19(9):1029–1039, 1993.

[72] Trobec, R., Šterk, M., Praprotnik, M. in Janežič, D.: Implementation and evaluation of MPI-based parallel MD program. *Int. J. Quant. Chem.*, 84(1):23–31, 2001.

[73] Geijn, R. A. van de: LAPACK working note 29 on global combine operations. Technical Report CS-91-129, University of Tennessee, April 1991.

[74] Gunsteren, W. F. van in Berendsen, H. J. C.: Computer simulation of molecular dynamics: Methodology, applications, and perspectives in chemistry. *Angew. Chem. Int. Ed*, 29(9):992–1023, 1990.

[75] Windemuth, A.: Advanced algorithms for molecular dynamics simulation. V Mattson, T. G., urednik, *Parallel Computing in Computational Chemistry*, strani 151–169. American Chemical Society, 1995.

Tabela oznak

a_i	Atom i.
A_p	Množica atomov, za katere je procesor p domači procesor (enačba 2.9, str. 34).
B	Množica blokov.
$\|B\|$	Število blokov.
b_i	Blok i.
$b_i \times b_j$	Vektorski produkt dveh blokov; vsebuje interakcije med atomi dveh blokov.
$\mathcal{B}$	Preslikava atomov na bloke; porazdeli atome v bloke (enačba 2.3, str. 34).
$\hat{\mathcal{B}}$	Preslikava procesorjev na produkte blokov (enačba 2.5, str. 34).
$\hat{\mathcal{B}}^{-1}$	Preslikava produktov blokov na procesorje (enačba 2.5, str. 34); pri metodi DDFD je dopolnjena v preslikavo interakcije para atomov na procesor, ki to interakcije računa (enačba 2.10, str. 38).
Δ_p	Neuravnoteženost procesorja p; izraža razliko med računskim bremenom procesorja in povprečnim računskim bremenom (enačba 3.16, str. 59).
E	Vzporedna učinkovitost oz. učinkovitost vzporednega računanja (str. 24).
$\vec{f_i}$	Celotna sila, ki deluje na i-ti atom (enačba 2.1, str. 32).
$\vec{f_{ij}}$	Sila, ki nastane na i-tem atomov zaradi interakcij z j-tem atom; je delna sila celotne sile $\vec{f_i}$ (enačba 2.1, str. 32).
l	Zakasnitev prenosa enega sporočila.
N	Število atomov v molekulskem sistemu.
nb	Število vseh neveznih interakcij (str. 59).
nb_p	Število neveznih interakcij, ki jih računa procesor p (str. 55).

nb^{eq} Število neveznih interakcij v diagonalnih produktih blokov (enačba 3.12, str. 59).

nb^{neq} Število neveznih interakcij v produktih blokov pod diagonalo (enačba 3.12, str. 59).

nb_p^{eq} Število neveznih interakcij, ki so porazdeljene iz diagonale in jih računa procesor p (enačba 3.14, str. 59).

nb_p^{neq} Število neveznih interakcij, ki jih računa procesor p in niso z diagonale (enačba 3.13, str. 59).

nu faktor neuravnoteženosti (enačba 3.4, str. 55).

P Množica procesorjev.

$|P|$ Število procesorjev.

$\mathcal{P}$ Preslikava atomov na procesorje; atomu priredi domači procesor, ki zanj integrira gibalne enačbe (enačba 2.7, str. 34).

$\hat{P}_b$ Procesorji, ki imajo podatke atomov bloka b (enačba 2.6, str. 34).

$\vec{r}_i$ Koordinate i-tega atoma (enačba 1.1, str. 15).

r Razdalja med dvema atomoma.

r_{cutoff} Razdalje *cut-off* (str. 18).

s Hitrost prenosa sporočila.

S Pohitritev vzporednega računanja (enačba 1.5, str. 24).

T Čas računanja na enem procesorju (str. 24).

T_p Čas vzporednega računanja na večih procesorjih (str. 24).

t_{korak} Čas izračuna enega celotnega koraka simulacije molekulske dinamike brez uravnoteženja (enačba 3.1, str. 53).

t'_{korak} Čas izračuna enega celotnega koraka simulacije molekulske dinamike pri dinamičnem uravnoteženeju (enačba 3.6, str. 56).

t_{b} Čas računanja ene vezne interakcije (enačba 3.2, str. 54).

t_{nb} Čas računanja ene nevezne interakcije (enačba 3.2, str. 54).

$V(\vec{r}^N)$ Potencial celotnega sistema v odvisnosti od koordinat vseh atomov (enačba 1.2, str. 16).

Stvarno kazalo

Doktorska disertacija z naslovom *Vzporedne računalniške simulacije na gručah osebnih računalnikov* je v celoti rezultat mojega dela pod mentorstvom prof. dr. Boruta Robiča in univ. znan. svet. dr. Dušanke Janežič. Vsi dosežki, na katerih sem gradil, so dosledno označeni z navedbo virov literature.

Urban Borštnik

Objave iz doktorskega dela

Problematika celotne disertacije je v sledečih objavljenih člankih, člankih, poslanih v objavo in člankih v pripravi:

1. Hodošček, M., Borštnik, U. in Janežič, D.: CROW for Large Scale Macromolecular Simulations. *Cell. Mol. Biol. Lett.*, 7(1):118–119, 2002.

2. Borštnik, U., Hodošček, M. in Janežič, D.: Specialized network topologies for efficient communication in computer clusters V *User opportunities/network challenges: TERENA networking conference and CARNet users' conference 2003*.

3. Borštnik, U., Hodošček, M. in Janežič, D.: Improving the performance of molecular dynamics simulations on parallel clusters. *J. Chem. Inf. Comput. Sci.*, 44(2):359–364, 2004.

4. Borštnik, U., Hodošček, M. in Janežič, D.: Fast parallel molecular simulations. *Croat. Chem. Acta*, 78(2):211–216, 2005.

5. Borštnik, U. in Janežič, D.: Symplectic molecular dynamics simulations on specially designed parallel computers. *J. Chem. Inf. Model.*, 45(6):1600–1604, 2005.

6. Janežič, D. in Borštnik, U. Large-scale molecular dynamics simulations on parallel clusters. V Vajterišic, M. in dr., ur., *Parallel numerics '05: Theory and applications*, str. 223–231, University of Salzburg, Salzburg, Jožef Stefan Institute, Ljubljana, 2005.

7. Borštnik, U., Hodošček, M., Janežič, D. in Lukovits, I.: Electronic structure of zig-zag and armchair nanotubes obtained by density functional calculations. *Chem. Phys. Lett.*, 411(4–6):384–388, 2005.

8. Dolenc, J., Borštnik, U., Hodošček, M., Koller, J. in Janežič, D.: An ab initio QM/MM study of the conformational stability of complexes formed by

netropsin and dna. the importance of van der waals interactions and hydrogen bonding. *J. Mol. Struct., Theochem*, 718:77–85, 2005.

9. Trobec R., Borštnik, U. in Janežič, D. Communication performance of d-meshes in molecular dynamics simulation. *J. Math. Chem.*, 2007, Poslano v objavo.

10. Kutnar, K., Borštnik, U., Marušič, D. in Janežič, D. Interconnection networks for parallel molecular dynamics simulation based on hamiltonian cubic symmetric topology. *J. Math. Chem.*, 2007, Poslano v objavo.

11. Borštnik, U., Brooks, B.R. in Janežič, D.: Distributed diagonal force decomposition method. I. Description of the method. Pripravljeno za objavo v *J. Comp. Chem.*

12. Borštnik, U., Miller, T., Brooks, B.R. in Janežič, D.: Distributed diagonal force decomposition method. II. Force decomposition machine. Pripravljeno za objavo v *J. Comp. Chem.*