SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 3253

GPU implementacija vremenski i memorijski učinkovitoga paralelnog algoritma za poravnanje slijedova

Marija Mikulić

Zagreb, lipanj 2013.

Umjesto ove stranice umetnite izvornik Vašeg rada.
Da bi ste uklonili ovu stranicu obrišite naredbu `\izvornik`.

Hvala Miletu što me motivirao da se posvetim računarstvu, na ukazanom povjerenju, na prenesenom znanju i ponajviše na strpljenju.

SADRŽAJ

POPIS TABLICA

POPIS SLIKA

POPIS ISPISA

1. Uvod

Od svojih začetaka u 1940-ima, računala su postupno postala dio naše svakodnevice. To je područje ljudskog djelovanja koje se najbrže mijenja i evoluira. Ta evolucija se događa velikim dijelom zbog toga što se računala sve više i na sve više različitih načina primjenjuju u raznim područjima ljudskog djelovanja.

Jedno od tih područja je i biologija. U biologiji su za neke probleme, poput poravnavanja proteina i DNK, računala ključni alat za dolaženje do novih spoznaja. Iz te neodvojive veze nastalo je cijelo područje istraživanja koje nazivamo bioinformatika.

Biolozi ulažu velike napore u proučavanje našeg građevnog materijala, proteina i gena koji nas tvore. Proučavanje ovih molekula pokazalo se kao vrlo bitno i potencijalno vrlo korisno.

Primjerice, ako bismo mogli manipulirati svojom DNK, mogli bismo ukloniti iz svojih gena mnoge bolesti i sindrome koji danas, nažalost, pogađaju mnoge ljude.

Naravno, to otvara vrata i mnogim drugim stvarima te je predmet brojnih etičkih i filozofskih rasprava, ali prvotni cilj znanosti jest nastojati otkriti kako funkcionira svijet oko nas. Stoga nam je vrlo zanimljivo proučavati kako, u svojoj srži, funkcioniramo mi sami.

Iz te znatiželje i iz činjenice da su u biološkim molekulama spremljeni gigabajti informacija, pojavila se potreba za razvijanjem algoritama kojima bi se moglo učinkovito obrađivati tolike količine informacija. Kao jedan od ključnih problema nametnuo se problem poravnavanja molekula proteina i DNK.

Konkretan problem kojim se bavi ovaj završni rad jest poravnavanje jednog proteina s listom proteina. Odnosno, zanima nas koliko je neki protein udaljen od svakog od n proizvoljnih proteina. Ovaj rad opisuje rješenje toga problema algoritmom koji koristi predmetačni račun i Needleman-Wunschov algoritam.

Poglavlje 2 opisuje općenito problem poravnavanja struktura. U poglavlju 3 izložen je Needleman-Wunschov algoritam, a u poglavlju 4 predmetačni račun. Poglavlje 5 opisuje tehnologiju CUDA, a poglavlje 6 specifičnosti implementacije, dok se u poglavlju 7 izlažu dobiveni rezultati. Poglavlje 8 posvećeno je zaključku rada.

2. Poravnavanje struktura

Problem kojim se bavi ovaj rad jest problem poravnavanja proteina za potrebe biologije. Svrha tog poravnavanja jest ustanoviti područja sličnosti između proteina, jer ona mogu indicirati funkcijsku, strukturalnu ili evolucijsku vezu između proteina.

Za potrebe informatike, ovaj problem se može svesti na problem poravnavanja nizova znakova. Za ilustraciju, zamislimo da se nizovi koje poravnavamo stave jedan ispod drugog. U tom slučaju, cilj je tako postaviti znakove nizova da oni izgledaju najsličnije moguće.

Razlikujemo dvije vrste poravnavanja: globalno i lokalno.

Globalno poravnavanje je postupak kojim se jedan niz preslikava na drugi od početka do kraja. Pri tome je dozvoljeno u bilo koji niz umetnuti prazninu na proizvoljno mjesto, osim na prvo i posljednje. Ukoliko je jedan niz znatno dulji od drugog, ovakvo poravnanje može za posljedicu imati umetanje velikog broja praznina u kraći niz.

Za razliku od globalnog, lokalno poravnanje ne vodi računa o tome da se iskoriste svi znakovi oba niza. Ovdje je cilj poravnati jedan niz na dio drugoga na način da taj niz i dio drugog na koji je poravnan budu najsličniji.

2.1. Ocjenjivanje procjepa

Praznine koje se pojavljuju u nizovima znakova zapravo predstavljaju procjepe

```
Globalno        FTFTALILLAVAV
                F--TAL-LLA-AV

Lokalno         FTFTALILL-AVAV
                --FTAL-LLAAV--
```

Slika 2.1: Primjer globalnog i lokalnog poravnanja dvaju nizova

3. Needleman-Wunschov algoritam

Poznat još i kao algoritam optimalnog poravnanja, Needleman-Wunschov algoritam koristi se za globalno poravnavanje slijedova. To je prvi algoritam koji je primijenio dinamičko programiranje na problem poravnanja slijedova u biologiji.

Za ilustraciju rada algoritma poslužit će primjer poravnanja DNK slijedova *AGACTAGTTAC* i *CGAGACGT*. Nad abecedom (u ovom slučaju {A, C, G, T}) definira se *matrica sličnosti* za svaki par elemenata abecede, **S(a, b)**, kojom se boduje poravnanje ta dva elementa. Dodatno, funkcijom *d* kažnjava se umetanje praznine u gornji ili donji slijed.

Neka je zadana sljedeća matrica sličnosti **S**:

	A	**G**	**C**	**T**
A	10	-1	-3	-4
G	-1	7	-5	-3
C	-3	-5	9	0
T	-4	-3	0	8

Usto, neka je vrijednost funkcije kazne $d = 5$. Tada bi poravnanje:

A	G	A	C	T	A	G	T	T	A	C
C	G	A	-	-	-	G	A	C	G	T

imalo broj bodova 1: S(A, C) + S(G, G) + S(A, A) + 3 · d + S(G, G) + S(T, A) + S(T, C) + S(A, G) + S(C, T) = -3 + 7 + 10 - (3 · 5) + 7 + -4 + 0 + -1 + 0 = 1

3.1. Algoritam

Cilj Needleman-Wunschovog algoritma jest pronaći optimalno poravnanje neka dva slijeda A i B, odnosno ono poravnanje koje u obzir uzima sve znakove oba niza i usto ima najviši broj bodova s obzirom na sva moguća poravnanja (dozvoljeno je postojanje više rješenja s istim brojem bodova).

Označimo s *n* duljinu slijeda A i s *m* duljinu slijeda B. Za potrebe pronalaska optimalnog globalnog rješenja potrebno je alocirati dvodimenzionalnu matricu dimenzija $n \times m$. Svaki redak matrice predstavlja znak u slijedu A, a svaki stupac predstavlja znak u slijedu B. Element matrice F_{ij}, gdje je *i* indeks retka, a *j* indeks stupca, predstavlja broj bodova za poravnanje prvih *i* znakova slijeda A i prvih *j* znakova slijeda B.

Pošto tražimo optimalno globalno poravnanje, želimo biti sigurni da će poravnanje krenuti od prvog znaka slijeda A i prvog znaka slijeda B. Zbog toga se postavljaju sljedeći početni uvjeti:

$$F_{0j} = d \times j, i = 0 \ldots n\text{-}1$$
$$F_{i0} = d \times i, i = 0 \ldots m\text{-}1$$

Svi ostali elementi matrice računaju se prema formuli:

$$F_{i,j} = max \begin{cases} F_{i-1,j-1} + S(A_i, B_j) \\ F_{i,j-1} + d \\ F_{i-1,j} + d \end{cases}$$

Nakon što su izračunati svi elementi matrice, element F_{nm} sadrži broj bodova optimalnog globalnog poravnanja.

Pseudokod algoritma dan je u ispisu:

Ispis 3.1: Needleman-Wunschov algoritam

```
1  za svaki i od 0 do duljina(A):
2    F[i, 0] = d * i
3  za svaki j od 0 do duljina(B):
4    F[0, j] = d * j
5  za svaki i od 1 do duljina(A):
6    za svaki j od 1 do duljina(B):
7      bez_praznine = F[i - 1, j - 1] + S[A[i], B[i]]
8      praznina_dolje = F[i - 1, j] + d
9      praznina_gore = F[i, j - 1] + d
10     F[i, j] = max(bez_praznine, praznina_dolje, praznina_gore)
```

3.2. Afina funkcija kazne

Opisani algoritam kao funkciju kazne koristi konstantnu vrijednost. Međutim, u primjeni ovog algoritma na poravnanje bioloških struktura, proteina i nizova DNK,

potrebno ga je modificirati. Naime, praznine koje se umeću predstavljaju mutacije, a veća je vjerojatnost da se dogodila jedna mutacija nad nekim podslijedom nego nekoliko mutacija duljine 1 zaredom. Stoga se definira afina funkcija kazne, koja za produljenje podslijeda praznina dodjeljuje manju kaznu nego za otvaranje novog podslijeda.

3.2.1. Formalna definicija Needleman-Wunschovog algoritma s afinom funkcijom kazne

Za verziju algoritma s afinom funkcijom kazne potrebno

4. Predmetačni račun

Problem predmetačnog računa definira se pomoću niza A = $\{a_0, a_1, a_2, ..., a_{n-1}\}$ i općenitog binarnog operatora $\oplus$. Cilj je generirati niz B = $\{0, 0 \oplus a_0, 0 \oplus a_0 \oplus a_1, ..., 0 \oplus a_0 \oplus ... \oplus a_{n-2}\}$.

Ovaj se problem pojavljuje često kao usko grlo u raznovrsnim problemima. Neki od njih su:

- leksička usporedba nizova znakova (npr. određivanje da niz "strategija" dolazi ispred niza "strateški" u rječniku)
- paralelno ostvarenje *radix-sort* i *quick-sort* algoritama
- određivanje vidljivosti točaka u 3D krajoliku – uz operator *max*
- zbrajanje s brojevima višestruke (proizvoljne) preciznosti
- alokacija procesora/memorije
- pretraživanje regularnih izraza (npr. u implementaciji *grep* naredbe u *UNIX*-u)
- izvedba nekih operacija nad stablima (npr. dubina svakog čvora u stablu)

4.1. Primjer

Za ilustraciju problema, pogledajmo slijedeći primjer:

Zadan je niz A:

indeks	0	1	2	3	4	5	6	7
vrijednost	5	3	-8	4	12	6	-9	1

Ako je binarni operator "+", tada je cilj dobiti niz B:

indeks	0	1	2	3	4	5	6	7
vrijednost	0	5	8	0	4	16	22	13

Gore navedeni primjer opisuje isključivi predmetačni račun. To znači da za element b_i ($i = 0..n$) niza B u obzir uzimamo elemente niza A čiji je indeks manji od i, a b_0 popunimo neutralnim elementom (primjerice, za zbrajanje je to 0).

Druga vrsta predmetačnog računa jest uključivi predmetačni račun, koji za izračun elementa b_i koristi sve elemente od a_0 do a_i, uključivo. Za gore navedeni primjer, niz B bi izgledao ovako:

indeks	0	1	2	3	4	5	6	7
vrijednost	5	8	0	4	16	22	13	14

Isključivi predmetačni račun se jednostavno generira iz uključivog tako da se svi elementi pomaknu za jedno mjesto udesno, a kao nulti element se doda neutralni element.

4.2. Algoritam

Predmetačni račun je jedan od algoritama koji se čine inherentno sekvencijalnima, ali za koje postoji učinkovita paralelna implementacija.

4.2.1. Sekvencijalni algoritam

Sekvencijalna implementacija ovog algoritma je trivijalna i svodi se na dinamičko programiranje. Verzija algoritma sa binarnim operatorom "+" dana je u ispisu:

Ispis 4.1: Predmetačni zbroj

```
1 b[0] := 0
2 za svaki k od 1 do n-1
3   b[k] := a[k-1] + b[k-1]
```

4.2.2. Paralelni algoritam

Učinkovita paralelna implementacija predmetačnog računa temelji se na radu Blellocha i koristi balansirano stablo. Ideja je izgraditi balansirano binarno stablo nad ulaznim podacima te njegovim obilaskom do listova do korijena pa od korijena do listova napraviti račun.

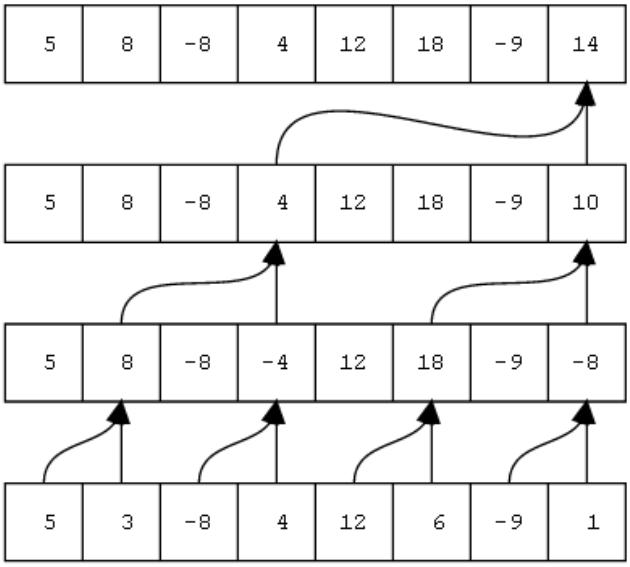
Binarno stablo s n listova ima dubinu $d = log_2 n$, a svaka dubina ima 2^d čvorova. Ako izvodimo jedan izračun po čvoru, u jednom obilasku stabla izvodimo $O(n)$ izračuna. Ovo binarno stablo nije fizički implementirano, nego je koncept koji koristimo

kako bismo odredili što je zadatak svake dretve u obilasku stabla, a svi izračuni izvode se nad ulaznim nizom podataka.

Izračun se izvodi u dvije faze: reduciranje i predtraženje.

U fazi reduciranja, stablo se obilazi od listova do korijena i izvode se djelomični izračuni, te na kraju ove faze korijen stabla sadrži izračun za sve čvorove u stablu.

Slika i ispis predstavljaju fazu reduciranja na gore navedenom primjeru.



Slika 4.1: Faza redukcije

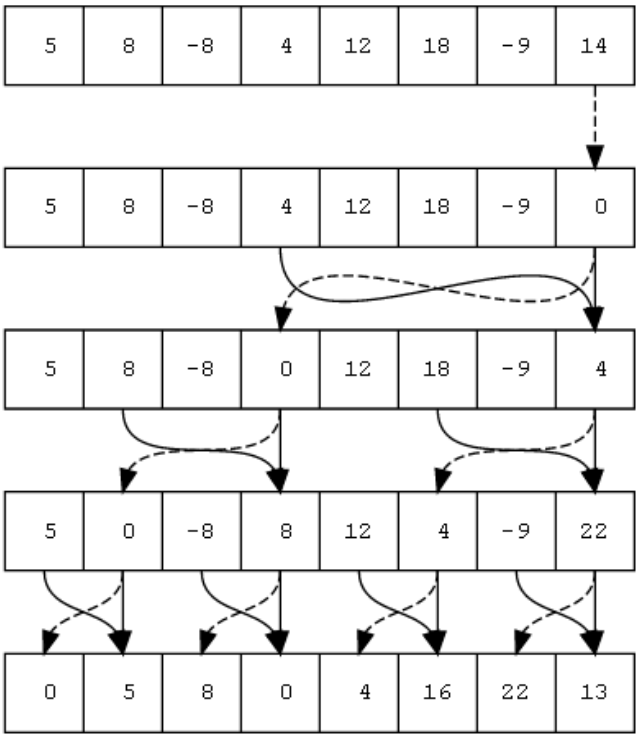
Ispis 4.2: Faza reduciranja

```
1 za svaki d od 0 do log2n - 1:
2   za svaki k od 0 do n - 1 s korakom 2^(d+1) u paraleli:
3     a[k + 2^(d+1) - 1] = a[k + 2^d - 1] + a[k + 2^(d+1) - 1]
```

U fazi predtraženja stablo se obilazi od korijena do listova i koriste se djelomični izračuni iz faze reduciranja kako bismo izveli predmetačni račun za svaki element.

U korijen stabla se umetne neutralni element. Zatim, u svakom koraku čvor trenutne razine svom lijevom djetetu preda svoju vrijednost, a zbroj svoje vrijednosti i prethodne vrijednosti lijevog djeteta preda desnom djetetu.

Slika i ispis predstavljaju fazu predtraženja na gore navedenom primjeru.



Slika 4.2: Faza predtraženja

Ispis 4.3: Faza predtraženja

```
1 a[n-1] = 0
2 za svaki d od log2n - 1 do 0:
3   za svaki k od 0 do n - 1 s korakom 2^d + 1:
4     temp = a[k + 2^d - 1]
5     a[k + 2^d - 1] = a[k + 2^(d+1) - 1]
6     a[k + 2^(d+1) - 1] = temp + a[k + 2^(d+1) - 1]
```

5. CUDA tehnologija

Grafičke kartice su prvotno dizajnirane kao grafički ubrzivači i podržavale su cjevovode točno određenih funkcija. No, u 1990-im godinama doživjele su dramatičan razvoj i postajale sve više programabilne. NVIDIA, tvrtka koja je i skovala termin GPU (*graphics processing unit - grafička procesna jedinica*), svoj je prvi GPU proizvela 1999. Od tad su moć te tehnologije, osim proizvođača računalnih igara i umjetnika, prepoznali i mnogi istraživači te ju počeli koristiti za svoje potrebe.

Međutim, u to doba je bilo potrebno "prevariti" grafičku karticu koju se htjelo iskoristiti, to jest, trebalo podatke pretočiti u probleme koji se mogu prikazati trokutima i poligonima. Kako bi se to izbjeglo, razvio se GPGPU (*general purpose graphics processing unit - grafička procesna jedinica opće namjene*).

Uskoro su se pojavile i prilagodbe općih programskih jezika za novonastalu tehnologiju.

CUDA tehnologija (punog naziva *Compute Unified Device Architecture - računski ujedinjena arhitektura uređaja*) je programski model i platforma za paralelno programiranje koju je razvila tvrtka NVIDIA i implementirala na svojim grafičkim procesnim jedinicama.

Programe za grafičke procesne jedinice sa CUDA arhitekturom jednostavno je napisati pomoću proširenja jezika C (*CUDA C*). *CUDA C* proširuje C na način da dozvoljava korisniku da definira funkcije, nazvane *kernel*, koje izvodi *N CUDA dretvi* paralelno.

6. Implementacija

7. Rezultati

8. Zaključak

Zaključak.

LITERATURA

GPU implementacija vremenski i memorijski učinkovitoga paralelnog algoritma za poravnanje slijedova

Sažetak

Poravnavanje slijedova proteina bitan je dio istraživanja moderne biologije. Kako se radi o velikoj količini podataka, njihova računalna obrada je ključna za učinkovitost istraživanja.

Ovaj rad bavi se pronalaženjem optimalnog poravnanja jednog proteina i liste od N proteina. U informatici se proteini mogu prikazati kao nizovi znakova nad fiksnom abecedom te se njihovo poravnavanje svodi na poravnavanje tih nizova znakova.

Obrada podataka i nalaženje optimalnog poravnanja izvodi se pomoću modificiranog Needleman-Wunschovog algoritma s afinom funkcijom kazne i predmetačnim računom.

Ključne riječi: Needleman-Wunsch, CUDA, paralelizacija, poravnanje, protein, predmetačni račun

GPU implementation of a space and time optimal parallel sequence alignment algorithm

Abstract

Protein sequence alignment makes for a big portion of modern day research in biology. Since the amount of data that needs to be processed is vast, it is crutial to develop optimal software to facilitate this procedure.

This paper deals with finding an optimal alignment between a protein and a list of N proteins. For the purpose of informatics, protein sequences are represented as strings over a fixed alphabet and their alignment comes down to string alignment.

Score of the optimal alignment is found using a modified version of the Needleman-Wunsch algorithm that uses an affine gap penalty function and prefix computing.

Keywords: Needleman-Wunsch, CUDA, parallelization, sequence alignment, protein, prefix computing, scan