Predavanje 1

1) Sta su asimptotske notacije

Asimptotske notacije sluze za opis vremena izvrsenja algoritma T(n)

2) Nabroj asimptotske notacije

Teta,O,o,Omega,omega

3) Objasni Teta notaciju

Teta notacija sluzi za odredjivanje vremena izvrsenja algoritma u najgorem slucaju. Zasniva se na odbacivanju clanova nizeg reda i zanemarivanju koeficijenta uz vodeci clan.

4) Objasni O notaciju

Veliko O notacija sluzi za definisanje asimptotski gornje granice zadate funkcije

Malo o predstavlja granicu koju daje O notacija. Moze i ne mora biti asimptotski uska.

5) Objasni Omega notaciju

Omega notacija sluzi za definisanje asimptotski donje granice zadate funckije.

Omega(g(n)) daje vreme izvrsavanja algoritma u najboljem slucaju.

Malo omega oznacava donju granicu koja nije asimptotski uska.

6) Objasni teoremu o asimp. notacijama

Za bilo koje dve funkcije f(n) i g(n) vazi da je :

f(n) = Teta(g(n)) AKKO

f(n) = O(g(n)) && f(n) = Omega(g(n))

7) Master Metoda

Master metoda resava rekurentne jednacine oblika : T(n) = aT(n/b) + f(n)

-Ova jednacina opisuje vreme izvrsenja algoritma koji deli problem velicine n na a podproblema

- svaki velicine n/b se resava rekurzivno

- vreme resavanja podproblema je T(n/b)

-f(n) pokriva cenu deljenja problema na podprobleme

8) Tri slucaja Master teoreme

Ako je f(n) = O(n^log.b a-epsilon)

=> T(n) = Teta(n^log.b a)

Ako je f(n) = Teta(n^log.b a)

=> T(n) = Teta(n^log.b a * lg n)

Ako je f(n) = Omega(n^log.b a+epsilon)

=> T(n) = Teta(f(n))

/*

a = 1

b = 3/2

fn = 1

T(n) = Teta(n^log.3/2 1 * lg n)

T(n) = Teta(1 * lg n)

T(n) = Teta(lg n)

n^log(3/2) 1

Pa ako je fn polinominalno manja od n^log(b)a onda je 1.

Ako je fn polinomijalno jednaka n^log(b)a onda je 2.

A ako je veci onda je 3. S tim da moras proveriti da livazi neki uslov regularnosti

Af(n/b) <= cf(n) za c<1

*/

Predavanje 2

1) Sta je insertion sort

Insertion sort pripada klasi inkrementalnih algoritama. Efikasan je za sortiranje malog broja elemenata. Predstavlja simulaciju nacina na koji covek sortira karte u levoj ruci, gde se karta u desnoj poredi sa kartama u levoj.

2) Invarijanta petlje sta je i njene osobine

Invarijanta petlje se koristi kao dokaz da je algoritam korektan.Ima 3 osobine :

1.Inicijalizacija - Istinita je pre prve iteracije

2.Odrzavanje - Ako je istinita pre iteracije, ostaje istinita i posle iteracije

3.Zavrsetak - Kada se petlja zavrsi, invarijanta dokazuje da je algoritam korektan

3) Invarijanta petlje za Insertion-sort

1.Inicijalizacija - Pre prve iteracije j=2, podniz je A[1] tj. jedan element i on je sortiran

2.Odrzavanje - Povecanje indeksa j za sledecu iteraciju ne utice na prethodno sortiran niz

3.Zavrsetak - Kada je j > A.length, izlazi iz for petlje

4) Objasni vremena kod analize Insertion-sorta

Vreme izvrsenja algoritma zavisi od velicine ulaza. Vreme izvrsenja = broj operacija. Sto je ulazni niz bolje sortiran, brze ce se izvrsiti algoritam. Najgori slucaj je ukoliko je niz obrnuto sortiran. Najcesce nas zanima samo najgore vreme iz 3 razloga :

1. Koji god niz da uzmemo,vreme izvrsenja ne moze biti vece

2. Najgori slucaj se desava veoma cesto

3. Random slucaj je skoro los kao najgori slucaj

5) Objasni pristup Podeli i Zavladaj opste

Podeli i zavladaj koriste rekurzivni algoritmi.Na svakom nivou rekurzije postoje 3 koraka :

1. Podeli - deli problem na n podproblema

2. Zavladaj - resava podprobleme rekurzivno. Ukoliko su dovoljno mali, resava ih direktno

3. Kombinuj - Resenje podproblema u ukupno resenje originalnog problema

6) Objasni pristup Podeli i Zavladaj kod Insertion Sorta

1.Podeli - Deli niz od n elemenata u 2 podniza od n/2 elemenata

2.Zavladaj - Sortira dva podniza rekurzivno

3.Kombinuj - Spaja dva sortirana podniza u jedan sortiran niz

Rekurzijom se spustamo do dna, do niza duzine 1,a niz sa 1 elem je vec sortiran

7) Sta je procedura Merge

Potrebno je Teta(n) vremena gde je n = r - p + 1 = broj elemenata koje treba spojiti.

1.Osnovni korak - Izbor manje karte sa vrha dva spila

2.Uklanjanje karte sa vrha i smestanje dole

Osnovni korak se ponavlja dok se jedna gomila ne isprazni, onda ostatak druge gomile stavimo licem na dole. Korak uzima Teta(1) jer se sve vreme porede samo 2 karte => Teta(n) ukupno

8) Opsta rekurentna jednacina

$$T(n) = \begin{cases} \text{Teta}(1), & n <= c \\ aT(n/b) + D(n) + C(n), & \text{else} \end{cases}$$

9) Objasni pristup Podeli i Zavladaj kod Merge Sorta

1.Podeli- Racuna sredinu podniza D(n) = Teta(1)

2.Zavladaj - Dva podproblema velicine n/2

3.Kombinuj - Merge nad nizom od n elemenata uzima Teta(n) vremena, pa je C(n) = Teta(n)

Rekurentna za Merge sort :

$$T(n) = \begin{cases} \text{Teta}(1), & n = 1 \\ 2T(n/2) + \text{Teta}(n), & n > 1 \end{cases}$$

Master metoda : T(n) = Teta(n lg n)

Predavanje 3

1) Objasni platformu za dinamicku paralelizaciju

Platforma za dinamicku paralelizaciju omogucava specificiranje paralelizma u apk. Konkurentna platforma sadrzi rasporedjivac i podrzava 2 apstrakcije :

1. Ugnjezdeni paralelizam
2. Paralelne petlje

2) 4 prednosti modela dinamicke paralelizacije

1. Pseudo kod sa 3 kljucne reci : parallel,spawn i sync
2. Model obezbedjuje kvantifikaciju paralelizma zasnovanu na pojmovima RAD i RASPON
3. Mnogi || algoritmi proisticu iz prakse podeli i zavladaj
4. Moredne platforme podrzavaju neke od varijanti dinamicke || npr : Cilk, Cilk++, TBB...

3) Paralelizuj sl kod :

Fib(n)

1.if n <=1

2. return n

3.else x = Fib(n - 1)

4. y = Fib(n - 2)

5. return x + y

Resenje:

Fib(n)

1.if n <= 1

2. return n

3.else x = spawn Fib(n - 1)

4. y = Fib(n - 2)

5. sync

6. return x + y

4) Objasni model paralelnog izvrsavanja

Imamo graf racunanja G = (V,E)

Cvorovi V su intrukcije

Grane E su zavisnost izmedju instrukcija

Ako u G postoji usmerena putanja od linije 'u' do linije 'v', dve linije su logicki u rednoj vezi. U suprotnom su u paralelnoj.

5) Koje su 4 vrste grana grafa racunanja

Grana nastavka(u,u') - crta se udesno, povezuje liniju 'u' sa 'u prim'

Grana mrescenja(u,v) - crta se nadole i pokazuje da je linija izvrsenja u izmrestila liniju v

Grana poziva - crta se nadole i predstavlja obican poziv procedure

Grana povratka(u,x) - crta se na gore i pokazuje da se linija izvrsenja u vraca njenoj pozivajucoj proceduri x

6) Koje su mere performanse paralelnog algoritma

Mere performanse paralelnog algoritma su :

1. Rad - ukupno vreme racunanja na jednom procesoru. Ako je vreme za svaki cvor 1, rad = br. cvorova

2. Raspon - najvece vreme potrebno da se izvrse linije duz bilo kog puta. Ako je vreme za svaki cvor 1, raspon = br. cvorova duz najduze putanje

7) Sta je kriticna putanja

Kriticna putanja grafa je najduza putanja u grafu

8) Koja vremena sve postoje i kratko ih opisati

Postoji Tp,T1,T(inf)

1. Tp - Vreme na P procesora

2. T1 - Vreme na 1 procesoru

3. T(inf) - Vreme na neogranicenom broju procesora, u teoriji svaka linija bi imala svoj procesor.

Rad i raspon obezbedjuju donje granice za vreme izvrsavanja Tp - zakon rada i raspona

9) Formulisi zakon Rada

Zakon RADA : Tp >= T1/P

10) Formulisi zakon Raspona.

Zakon RASPONA : Tp >= T(inf)

11) Objasni zakon ubrzanja

Ubrzanje na P procesora je odnos T1/Tp i govori koliko je puta brze izvresenje na P procesora.

Ako T1/Tp = Teta(P) => Linearno ubrzanje

Ako T1/Tp = P => Savrseno linearno ubrzanje

12) Sta je paralelizam i koje sve perspektive postoje

Paralelizam je odnos T1/T(inf)

Tri perspektive :

1. Kao odnos - srednja kolicina rada

2. Kao gornja granica - max ubrzanje

3. Ogranicenje savrsenog linearnog ubrzanja - max br procesora za savrseno linearno ubrzanje

13) Sta je labavost paralelizma

Labavost paralelizma predstavlja odnos (T1/T(inf))/P

Faktor sa kojim paralelizam algoritma prevazilazi broj procesora. Ako je labavost manja od 1, ne moze da se ostvari savrseno linearno ubrzanje. Sto se vise odaljava od 1, ubrzane sve blize savrsenom.

14) Sta je pohlepni rasporedjivac i koje sve korake ima

Pohlepni rasporedjivac - dodeljuje sto je moguce vise linija svakom koraku. Koraci j:

1.Potpun

2.Nepotpun

15) Objasni i formulisi Teoremu o gornjoj granici Tp

Teorema o gornjoj granici Tp :

Na P procesora, pohlepni rasporedjivac izvrsava paralelni algoritam sa radom T1 i rasponom T(inf) u vremenu: Tp <= T1/P + T(inf)

16) Objasni trku do podataka

Trka do podataka - desava se izmedju dve logicki paralelne instrukcije koje pristupaju istoj memorijskoj lokaciji i bar 1 od tih instrukcija upusije u tu lokaciju

Rad Raspon i Paralelizam P-Fib(pitanje sa ispita)

Procitati pricu o razvoju na 512 procesora

Predavanje 3

1) Rad Raspon i Paralelizam direktong algoritma

Rad : Kod sa tri ugnjezdene for petlje sa po n iteracija => T1(n) = Teta(n^3)

Raspon : T(inf)(n) = Teta(n) jer je raspon za parallel_for Teta(lg n), a za obicnu Teta(n); a gleda se gore vreme

Paralelizam : Teta(n^3)/Teta(n) = Teta(n^2)

2) Strasenov Metod za mnozenje matrica - objasni

Kljuc je da se rekurzivno stablo ucini manje razgranatim. Umesto 8 mnozenja matrica n/2 puta n/2 on obavlja 7. Cena za uklanjanje jednog matricnog mnozenja; nekoliko matricnih sabiranja ali je taj broj const

3) Koraci strasenovog metoda

Metod se sastoji od 4 koraka:

1. Podeliti matrice A,B,C na podmatrice n/2 x n/2. Ovaj korak uzima Teta(1) vremena
2. Napraviti 10 matrica - Teta(n^2) vremena
3. Rekurzivno izracunati 7 matrica
4. Izracunati zeljene podmatrice za matricu C - Teta(n^2) vremena

4) Rad Raspon i Paralelizam P-Matrix-Multiply-Rec.

Rad : Podela matrica u Teta(1) vremenu. Osam rekurzivnih mnozenja podmatrica i sabiraju se.

T1(n) = 8 T1(n/2) + Teta(n^2)

Po masteru : T1(n) = Teta(n^3)

Raspon: T(inf)(n) = Teta(lg^2 n) metodom zamene

Paralelizam: T1(n)/T(inf)(n)= Teta(n^3 / lg^2n)

5) Rad Raspon i Paralelizam Strasenovog metoda

Rad : T1(n) = Teta(n^lg 7)

Raspon : Sedam rekurzivnih poziva izvrsava paralel. Dobija se ista rekurencija kao za P-M-M-R => T(inf)(n) = Teta(lg^2 n)

Paralelizam : T1(n)/T(inf)(n) = Teta(n^lg7 / lg^2 n)

6) Rad Raspon i Paralelizam Merge Sorta-P

Rad : T1(n) = 2T1(n/2) + Teta(n) = Teta(n lg n)

Raspon : Kako se dva rekurzivna poziva mogu izvrsiti || => T(inf)(n) = T(inf)(n/2) + Teta(n) = Teta(n)

Paralelizam : T1(n)/T(inf)(n) = Teta(lg n)

7) Objasni Binary Search

Poziv procedure uzima Teta(lg n) serijskog vremena u najgorem slucaju.

n = r - p + 1 velicina podniza na kome se procedura izvrsava

Posto je Binary Search serijska procedura, njen rad i raspon su u najgorem slucaju Teta(lg n) oba

Predavanje 5

1) Koraci u paralelizaciji programa

1. dekompozicija, 2. dodela, 3.orkestracija, 4. preslikavanje

2) ?

1. Dekompozicija ima cilj razbijanja zadatka na vise medjusobno nezavisnih podzadataka

2. Dodela ima cilj da grupise zadatke u procese.

3. Orkestracija resava prenos podataka izmedju procesa koji proizvode podatak i onih koji te podatke koriste

4. Preslikavanje procesa na jezgra viseprocesorskog sistema.

3) ?

Bernstajnov uslov

Izražavanje algoritma

Pronalaženje paralelizma

Struktura algoritma

Konstruisanje programa

Pomoćne strukture

Izvedbeni mehanizmi

4) Vrste dekompozicija

Vrste dekompozicija koje se mogu koristiti radi pronalaženja paralelizma su:

1. Dekompozicija zadataka
2. Dekompozicija podataka
3. Dekompozicija protočne obrade

5)?

Примењујемо шаблоне декомпозиције

Декомпозиција задатака који одговарју ажурирању једног атома (итерацији једне петље)

Шаблоне анализе зависности

Анализа контролних зависности

Анализа зависности података

Евалуација (оцена) пројекта

Узимамо у обзир циљну архитектуру

Анализирамо да ли подаци имају просторне особине за ефикасно баратање њиховим зависностима

6) Bernstajnov uslov

Imamo jedan skup memorijskih lokacija Ri iz kog zadatak 1 čita podatke, i skup memorijskih lokacija Wj u koje drugi zadatak piše podatke, da se ta dva zadatka mogu izvršiti paralelno ako i samo ako su zadovoljeni uslovi:

R - ulaz

W - izlaz

a)presek R1 i W2 je prazan skup

b)R2 presek W1 je prazan skup

c)W1 presek W2 je prazan skup

7) Imena projektantskih sablona

Paralelizam zadataka, podeli i zavladaj, geometrijska dekompozicija, rekurzivni podaci, protocna obrada, koordinacija na bazi dogadjaja.

8) Podrzavajuce strukture

SPMD (engl. Single Program Multiple Data);

Paralelizam petlje;

Master/Worker

Fork/Join

9) Vrste redukcija

Серијска редукција

Редукција заснована на стаблу

Рекурзивно удвајајућа редукција

TEST BR. 2

Predavanje 7

1) Šta Cilk nudi kao proširenje C/C++ jezika?

1. Tri kljucne reci(spawn, sync, for)
2. Hiper objekat
3. Naznake za nizove
4. Osnovne funkcije
5. Pragma SIMD

3) Kada ponašanje Cilk programa nije definisano?

Ponašanje Cilk programa nije definisano u slučaju da nemamo definisanu serijalizaciju tog programa

4) Od dve linije izvršenja u Cilk programu, koja je ranija?

Ako posmatramo dve linije izvršenja u Cilk programu, ranija je ona koja se izvršava pre u serijskom izvršenju, naravno za iste ulazne podatke.

5) Koje uslove mora da zadovolji kontrolna promenljiva cilk_for petlje?

Kontrolna promenljiva mora biti inicijalizovana, može biti int, pokazivač ili tip klase i ne sme biti const ili volatile.

6) Šta je važno zapamtiti u vezi grainsize pragma direktive?

Važno je zapamtiti da veličina grainsize utiče samo na prvi sledeći for i nema uticaj na naredne for petlje.

7) Gde se nalazi implicitni sync u Cilk programima?

Cilk blok koda ili funkcija ima implicitni sync na kraju tog bloka ako se pre synca nalazi spawn.

8

```
int fib(int n)
{
   if (n < 2) return n;
   int x = cilk_spawn fib(n-1);
   int y = fib(n-2);
   cilk_sync;
   return x+y;
}
```

8) Šta je hiperobjekt, a šta je pogled?

Hiperobjekt spada u specijalne objekte, koji omogućavaju siguran pristup djeljenim objektima

Obraćanje hiperobjektu rezultuje u referenci, koja se naziva pogled

10) Čemu služi operator sekcije u CEAN? Koji je njegov opšti format?

Operator sekcije u CEAN bira vise elemenata niza za paralelnu operaciju, a njegov opsti format je

<array base>[<lower bound>:<length>:<stride>]

11) Napiši Cilk program u jednoj liniji koji sabira jednodimenzionalne nizove x i y i rezultat upisuje u niz z.

```
z[:] = x[:] + y[:];
```

12) Napiši Cilk program u jednoj liniji koji pomoću redukcije računa sumu niza: int x[] = {1,2,3};

int suma = __sec_reduce_add(x[:]);

13) Napisati kratak Cilk program (par linija) koji pomoću mapiranja računa niz y tako što kvadrira odgovarajuće elemente niza x.

```
int my_square(int a) {return a * a;}
int x[] = {101,102, ...  ...150};
int y[50];
y[:] = my_square( x[:] )
```

Predavanje 8

1) Koja su dva tipa jednostavnih paralelni petlji u TBB?

1. parallel_for koji omogucuava paralelizaciju for perlje
2. parallel_reduce koji omogucava paralelizaciju jednostavnih petlji pomocu reduktora

2) Koja su dva tipa složenih paralelni petlji u TBB?

Parallel_do i protočna obrada.

3) Koji tipovi objekata za podelu iteracionog prostora paralelnih petlji postoje i čemu oni služe?

Auto_partitioner

simple_partitioner

affinity_partitioner

Sluze za odredjivanje (sugerisanje kompajleru) broja serijskih iteracija u bloku paralelne for petlje

4) Kako se podešava parametar grainsize?

Podesava se empirijski, testirajuci razlicite vrednosti. Cilj je napraviti grainsize koji nije previse mali da ne bi cena pravljenja task-a bila veca od cene obrade samog posla, ali i da nije previse veliki jer u tom slucaju ne iskoriscavamo punu propusnu moc paralelne obrade.

5) Kada ima smisla koristiti affinity_partitioner?

affinity_partitioner ima smisla koristiti kada:

1. Obradu čine par operacija po pristupu podatku
2. Podaci nad kojima petlja radi mogu da stanu u bafer(cache)
3. Petlja ili slična petlja se izvodi nad istim podacima

6) Koje funkcije moramo da napišemo za funktor za parallel_reduce?

Moramo da preklopimo operator (), pored običnog konstruktora potreban nam je i konstruktor deljenja("splitting" konstruktor) i metoda join.

7) Kada se koristi parallel_do?

Parallel_do se koristi u situacijama kada nam nije poznat unapred iteracioni prostor, odnosno ne znamo koliki je prostor.

8) Kako se prave filtri i protočne obrade u TBB?

Protočna obrada se pravi pozivom funkcije tbb::parallel_pipeline, a filtri tj. pojedinačni stepeni protočne obrade se prave pozivima funkcije tbb::make_filter.

9) U kojim režimima može da rade filtri u protočnim obradama u TBB?

Filteri mogu rade u 3 rezima:

1. serial_in_order - Serijski rezim sa ocuvanjem redosleda podataka
2. serial_out_of_order -Serijski rezim bez ocuvanja redosleda podataka
3. parallel - Rezim paralelne obrade

10) Šta ograničava propusnost paralelne obrade u TBB?

1. Brojem podataka koji mogu istovremeno da se obradjuju
2. Najsporiji filter je usko grlo protocne obrade
3. Moze biti ogranicena velicinom prozora

11) Koje vrste protočnih obrada postoje u TBB?

12) Koje metode koriste TBB kontejneri?

Fino zaključavanje - samo delovi kontejnera koji se moraju zaključati se i zaključavaju

Tehnike bez zaključavanja - kada niti uračunavaju i popravljaju efekte drugih niti koje ih ometaju

13) Koje vrste kontejnera postoje u TBB?

У TBB-у постоје 3 врсте контејнера:

-мапа (concurrent_hash_map)

-вектор (concurrent_vector)

-ред (concurrent_queue, concurrent_bounded_queue)

14) Koji su nedostatci redova? Objasniti ih.

Ред је уско грло, због одржавања ФИФО редоследа. Нит може чекати беспослена у реду.

Ред је пасивна структура, убачени елемент се може охладити у баферу.

Ако елемент преузима нит у другом језгру, он и све што референцира мора се пребацити у бафер тог језгра.

15) Kako se realizuje međusobno isključivanje niti u TBB?

Međusobno isključivanje niti u TBB-u se realizuje pomoću muteksa (mutex) i ključa (lock).

Muteks je objekat koji se može zaključati, prethodno dobijenim ključem. Samo jedna nit može imati ključ, ostale moraju čekati.

16) Objasniti dva načina zaključavanja muteksa u TBB?

17) Koje vrste muteksa postoje u TBB?

spin_mutex, spin_rw_mutex,

queuing_mutex, queuing_rw_mutex,

null_mutex,null_rw_mutex

18) Koje su patologije ključeva? Objasniti ih.

Međusobno blokiranje (deadlock), koje nastaje ako postoji krug niti, u kom svakak nit drži bar jedan ključ i čeka na muteks, koji je zaključala druga nit, a ni jedna nit ne želi da oslobodi svoje ključeve.

Pravljenje konvoja niti (convoying), koje nastaje tako što OS prekine nit koja drži ključ, druge niti moraju da sačekaju da prekinuta nit oslobodi kjuč.

19) Koje su prednosti i nedostaci atomskih operacija?

Prednosti atomskih operacija su da su mnogo brže od ključeva, a brže su zato što nema zaključavanja niti konvoja. Nedostatak atomskih operacija je da rade samo ograničen skup operacija.

20) Koje vrste memorijskih alokatora postoje i čemu oni služe?

1. scalable_allocator<T> rjesava problem skalabilnosti

2. cache_aligned_allocator<T> rjesava problem laznog dijeljenja linije bafera, do kojeg dolazi kada dvije niti pristupaju razlicitim rijecima iste linije

21) Zašto su TBB zadaci su puno “lakši” od pthreads niti?

Postoje 2 razloga zašto su TBB zadaci puno "lakši" od pthreads niti:

1. Zadatak (task) u TBB je mala rutina (nema kontekst).

2. TBB zadaci ne istiskuju jedan drugoga (ne postoji "preemption", kada zadatak krene sa radom on radi sve do svoga kraja).

22) Koje relacije postoje između TBB zadataka u grafu zadataka? Čemu služi refcount?

Razlikujemo relacije predak-potomak, i prethodnik-naslednik.

Ako imamo zadatak A kao pretka, i zadatak B kao potomka, tada je zadatak A naslednik zadatku B u smislu redosleda izvršavanja zadataka, odnosno A će se izvršiti posle B.

Tu nam služi refcount kao brojač referenci. Refcount nekog zadatka pretstavlja broj njegovih prethodnika, odnosno broj zadataka koji treba da se izvrše pre nego što taj zadatak može da počne svoje izvršavanje. Po potrebi ga uvećavamo za jedan, u slučaju operacije čekanja. (..wait_for_all()).

23) Koju strategiju primenjuje TBB raspoređivač? Zašto?

TBB распоређивач примењује стратегију обраде графа задатака која се заснива на баласирању између обраде по дубини и обраде по ширини. Обрада по дубини омогућава ефикасно коришћење хеш меморије и минимизира потребан меморијски простор који користи програм у неком тренутнку, док обрада по ширини повећава паралелизам.

24) Koje tehnike koristimo prilikom pisanja paralelnih programa sa grafovima zadataka?

Rekurzivan lanac reakcija

Prosledjivanje nastavka

Zaobilaženje raspoređivača

Ponovna upotreba

Prazni zadaci

Opšti aciklični grafovi zadataka

Predavanje 9

1) Koje alate koristi Parallel Advisor i čemu oni služe?

1. Survey koji otkriva mesta na kojima se troši najviše vremena. Može nam dati performanse za određene linije koda.

2. VS editor za unos oznaka o mogućnosti paralelizacije. Nakon toga se oznake zamenjuju stvarnim kodom iz paralelnog okruženja koje koristimo.

3. Suitability koji služi za procenu performansi za označene delove. Koristi matematičko modeliranje. Zahteva Release konfiguraciju.

4. Correctness prikazuje moguće probleme deljenja podataka na osnovu oznaka paralelnih mesta, zadataka, ključeva. Zahteva Debug konfiguraciju.

2) Čemu služi alat Parallel Amplifier?

Alat sluzi za davanje infromacija o performansi programa kao sto su: identifikovanje vrucih tacaka, odredjivanje najbolje sekcije koda za optimizaciju i pronalazenje one koja ne iskoriscuje raspolozivo vreme, informacije o UI operacijama unutar programa(kako, gdje i zasto nastaju), pronalazenje objekata odgovornih za sinhronizaciju i njihov uticaj na performanse, analiziranje performansi sa razlicitim algoritmima, sinhronizacionim metodama ili brojevima niti.

3) Kako alat Parallel Amplifier definiše pojam Ciljni paralelizam (Target Concurrency)?

4) U kojoj funkciji je paralelizam bio loš? Zašto? Kako je problem rešen?

5) Šta obezbeđuje OpenCL?

6) Koje šablone podržava OpenCL?

7) Šta je OpenCL kernel?

8) Šta je radna stavka?

9) Šta je NDO?

10) Kako se u OpenCL postiže skalabilnost?

11) Kako sarađuju radne stavke unutar iste radne grupe?

12) Koje komponente definiše OpenCL model platforme?

13) Šta je kontekst u OpenCL modelu? Čemu on služi?

14) Gde se izvršavaju radne stavke u OpenCL modelu platforme?

15) Šta znači labav model izvršenja radnih stavki?

16) Gde se izvršava radna grupa u OpenCL modelu platforme?

17) Gde se nalaze globalne sinhroizacione tačke u OpenCL modelu izvršenja?

18) Koje tri vrste OpenCL komandi postoje?

19) Koje su tri primarne sinhronizacione tačke u OpenCL modelu izvršenja?

20) Koje vrste OpenCL redova komandi postoje? Po čemu se razlikuju?

Predavanje 11

1) Koje su glavne osobine SPP?

Glavne osobine strukturnog paralelnog programiranja su to da postoji konačan broj upravljačkih struktura, zasnovan je na šablonima koji se zovu šabloni algoritamske strategije (ŠAS).

2) Na kom nivou je ŠAS?

ŠAS su na srednjem nivou apstrakcije i nalaze se između projektantskih šablona i implementacionih šablona.

3) Koji su delovi ŠAS?

Svaki ŠAS (šablon) ima dva dela: semantiku I implementaciju.

4) Koje grupe ŠAS postoje?

Grupe ŠAS su:

I Grupa (Šablon kompozicije)

II Grupa (Šabloni strukturne serijske kontrole toka)

III Grupa (Šabloni strukturne paralelne kontrole toka)

IV Grupa (Šabloni serijskog rukovanja podacima)

V Grupa (Šabloni paralelnog rukovanja podacima)

VI Grupa (Preostali paralelni šabloni)

VII (Nedeterministički šabloni)

5) Koji su simboli u dijagramima za prikazivanje ŠAS?

Konvencija:

- zadaci se pokazuju pravougaonim simbolima

- podaci ovalnim simbolima

- grupisani zadaci su u pravouglim okruženjima

- grupisani podaci su u zaobljenim okruženjima

- dodatni simboli su u obliku raznih poligonalnih oblika

- zavisnosti se obeležavaju strelicama, a treba izbegavati strelice koje pokazuju na gore, s obzirom na to da vreme teče od gore ka dole. Izuzetak su iteracije.

6) Čemu odgovara visina dijagrama za ŠAS?

Visina odgovara rasponu šablona, samo ukoliko se ne postoje strelice naviše.

7) Šta su blokovi zadataka?

Blokovi zadataka u dijagramima šablona su mesta za programski kod ili druge šablone.

8) Koja je osnovna predpostavka ŠAS Ugnježdavanje?

Osnovna pretpostavka je da svi sabloni podrzavaju ugnjezdavanje i da je dubina ugnjezdavanja neogranicena, takodje, sadrzavajuci sablon ne sme uvoditi nikakva ogranicenja koja bi ogranicavala koju vrstu sablona moze da sadrzi.

9) Navesti šablone serijske kontrole toka.

U grupu šablona serijske kontrole toka spadaju :

- Sekvenca
- Iteracija
- Izbor
- Rekurzija

10) Navesti šablone paralelne kotrole toka.

Grananje-Pridruživanje

Preslikavanje

Obrada suseda

Redukcija

Skeniranje

Rekurencija

Predavanje 12

1) Koji su šabloni serijskog rukovanja podacima?

Šabloni serijskog rukovanja podacima su: slučajno čitanje i pisanje, dodela steka, dodela memorije, funkcijski objekti i objekti.

Kako se u SPP rešava problem aliasa?

2)Alias :

Aliasi predstavljaju pokazivače koji pokazuju na isti objekat.

4)Šta su lambda funkcije?

Ламбда функције су врсте функцијских објеката (функтора), карактеристични су по томе што их користимо као функције а рукујемо као са подацима.

5) Kako se mogu generisati funkcijski objekti?

Funkcijski objekti se mogu generisati statički i dinamički.

6) Koji problem se pojavljuje kod šablona paralelnog rukovanja podacima?

Glavni problem koji se pojavljuje kod šablona paralelnog rukovanja podacima jeste data rase, odnosno trka do podataka. Ona se izbegava ukoliko se izbegne menjanje deljenih podataka, jer trka nastaje kada bar jedna nit piše u deljeni podatak

7) Koji su šabloni paralelnog rukovanja podacima?

Šabloni paralelnog rukovanja podacima su:

1. Pakovanje
2. Protočna obrada
3. Geometrijska dekompozicija
4. Skupljanje
5. Razbacivanje

8) Šta je glavna prednost šablona Pakovanje?

Шаблон се користи за елиминисање некоришћеног простора у збирци података, и као такав је користан кад се споји са шаблоном "пресликавање" да би се избегао непотребан излаз из шаблона. Такође може послужити да се сузи потребан меморијски пропусни опсег, и да се емулира контреола тока на SIMD машинама (и то са добром асимптотском перформансом)

9) Šta je mana šablona Protočna obrada?

Protočna obrada ima određen broj stepeni protočne obrade, i nisu skalabilne.

10) Koji problemi se pojavljuju kod šablona Geometrijska dekompozicija?

Problemi koji se javljaju kod šablona Geometrijska dekompozicija jeste kako rukovati graničnim uslovima i kada su ulazni i izlazni domeni deljivi na pločice iste veličine.

11) Koji problem se pojavljuje kod šablona Razbacivanje? Kako se on rešava?

Problem se pojavljuje ako dva upisa idu u istu lokaciju, moguće je trka do podataka. Ovaj problem je moguće rešiti na nekoliko načina :

Korišćenje asocijativnih operatora za kombinovanje elemenata

Nedeterminističko biranje jednog od više elemenata

Pridruživanje prioriteta pojedinim elementima

12) Po čemu se razlikuju serijska i superskalarna sekvenca?

Razlika je u tome što ako nema ivičnih efekata zadaci teku ili paralelno ili u nekom redosledu koji je različit od redosleda iz izvornog koda programa. Redosled je uslovljen zavisnošću podataka koja mora biti poznata raspoređivaču.

13) Šta je moguće realizovati pomoću šablona Buduće vrednosti?

Може се искористити за имплементацију графова задатака, као и за имплементацију неких других шаблона.

14) Po čemu se razlikuju sekvencijalni i spekulativni izbor?

Spekulativni izbor generalizuje sekvencijalni izbor. Razlikuju se po tome, što kod spekulativnog izbora obe alternative mogu da se izračunavaju paraleleno, i nakon određivanja uslova, suvišna grana se otkazuje.