|  |  |  |
| --- | --- | --- |
|  | Univerzitet u Istočnom Sarajevu  Elektrotehnički fakultet |  |

**Floyd – Warshall-ov algoritam**

Seminarski rad

Studije: Prvi ciklus

Odsjek: Računarstvo i informatika

Predmet: Paralelni računarski sistemi

Studenti: Mentor:

Kristina Knežević, 2060 Doc. dr Nikola Davidović

Tamara Elez, 2068

Istočno Sarajevo, septembar 2024

# **Sadržaj**

[Uvod 3](#_Toc175316731)

[1. Paralelno računarstvo 4](#_Toc175316732)

[1.1. Tipovi paralelizma 5](#_Toc175316733)

[1.2. Strategije paralelnog programiranja 5](#_Toc175316734)

[Literatura 7](#_Toc175316735)

## Uvod

Sadašnje stanje razvoja integrisanih kola ukazuje da ubrzavanje mikroprocesora jednostavnim povećanjem radne frekvencije više neće biti moguće. Tome doprinose fizička ograničenja minijaturizacije, transmisiona granica bakarnog provodnika itd. Usljed ove činjenice, danas je razvoj mikroprocesora prije svega usmjeren na uvećanje broja procesorskih jezgara, time uvećavajući značaj paralelnog programiranja i konkurentne obrade podataka uopšte. Dok se ne desi neki veći prodor u industriji mikroelektronike, paralelizacija ostaje jedini način da se pristupi bilo kakvoj masovnoj obradi.

Paralelni računski sistemi predstavljaju jedan od ključnih aspekata modernog računarskog inženjeringa, omogućavajući ubrzavanje izvođenja kompleksnih algoritama putem istovremenog izvršavanja zadataka. Jedan od takvih algoritama je Floyd-Warshall-ov, koji se koristi za pronalaženje najkraćih puteva između svih parova čvorova u grafu. Ovaj algoritam ima široku primjenu u raznim oblastima, uključujući mrežne protokole, analizu socijalnih mreža, i optimizaciju logističkih mreža.

Floyd-Warshall algoritam u svom serijskom obliku ima vremensku složenost što ga čini računski intenzivnim i vremenski zahtjevnim za grafove sa velikim brojem čvorova. Paralelizacija ovog algoritma predstavlja značajan pokušaj ka povećanju efikasnosti i smanjenju vremena izvršavanja.

U radu istražujemo metode i tehnike paralelizacije Floyd-Warshall algoritma implementiranog u programskom jeziku C, koristeći OpenMP tehnologiju. Rad će biti strukturisan tako da prvo razmotrimo osnovne koncepte paralelizacije i tehnologije koje se koriste za implementaciju paralelnih računarskih sistema, uz fokus na OpenMP tehnologiji, koja omogućava jednostavnu i efikasnu paralelizaciju programa pisanih u C jeziku. Nakon toga, pružićemo detaljan pregled algoritma, uključujući njegovu funkcionalnost i primjenu. U nastavku rada detaljno ćemo opisati proces paralelizacije Floyd-Warshall algoritma, uključujući analizu performansi i uporednu evaluaciju serijske i paralelne verzije algoritma.

Cilj ovog istraživanja je pokazati kako se performanse algoritma mogu značajno poboljšati korištenjem paralelizacije, te pružiti smjernice za implementaciju sličnih rješenja u praksi.

# Paralelno računarstvo

Paralelno računarstvo predstavlja brže riješavanje problema korišćenjem većeg broja procosora.

Kod paralelnog računarstva u užem smislu postoji dijeljena memorija između više procesora, dok kod distribuiranog računarstva svaki procesor poseduje svoju lokalnu memoriju, tj. svoj lokalni adresni prostor. Paralelni računar je računarski sistem sa više procesora koji podržava paralelno programiranje. Distribuiran računar je sistem sa distribuiranom memorijom u kome su elementi obrada povezan sa mrežom. Višejezgrani procesor je procesor koji sadrži više jedinica obrade (jezgara) na istom čipu.

Paralelni računari se mogu grubo klasifikovati prema nivou na kojem hardver podržava paralelizam: gdje višejezgrani i višeprocesorski računari imaju više elemenata obrade unutar jedne mašine, a klasteri koristite više računara da rade na istom zadatku. Specijalizovane paralelne računarske arhitekture se ponekad koriste zajedno sa tradicionalnim procesorima, za ubrzavanje specifičnih zadataka.

Paralelno programiranje je programiranje u jeziku koji dozvoljava da se eksplicitno zada kako će se razlagati delovi izračunavanja, tj. kako će biti izvršeni konkurentno na različitim procesorima. Tako da možemo reći da je to oblik programiranja u kojem se zadaci podijele na manje podzadatke koje se izvršavaju istovremeno na više procesora ili jezgara unutar jednog računarskog sistema.

Najveća motivacija za razvoj paralelnih sistema predstavljaju tzv. Grand Challenge problemi. To su fundamentalni problemi nauke i inženjerstva, koji su kompleksni i njihovo rješavanje putem numeričkih simulacija zahtjeva izuzetno brze računare.

Prednosti rada na paralelnim i distribuiranim sistemima mogu se sumirati u sljedećim stavkama: brzina, efikasnost resursa i mogućnost rješavanja kompleksnih problema.

Neki od nedostataka su da je teže pisati paralelne računarske programe nego sekvencijalne, jer konkurentnost uvodi nekoliko novih klasa potencijalnih softverskih grešaka. Komunikacija i sinhronizacija između različitih podzadataka su obično jedne od najvećih prepreka za dobijanje dobrih performansi paralelnog programa. Glavni problemi koji se uzimaju u obzir su particionisanje, sinhronizacija, zavisnost, kao i balansiranje opterećenja.

# Tipovi paralelizma

Paralelna obrada postoji u više oblika:

* na nivou bita
* na nivou instrukcije
* paralelizam podataka
* funkcionalni paralelizam

Paralelizam na nivou bita se odnosi na mogućnost procesora da istovremeno obradi više bitova podataka u jednoj operaciji i ova forma paralelnog izvršavanja instrukcija je bazirana na povećanju dužine procesorskih riječi. Ako procesor mora izvršiti sabiranje dvaju brojeva koji se sastoje od 32 bita na 32-bitnom procesoru, to će biti obavljeno u jednom ciklusu. Ali ako se koristi 64-bitni procesor, on može obraditi dvije takve operacije u istom vremenu, efektivno udvostručujući brzinu.

Paralelizam na nivou instrukcija se postiže kada se više operacija izvodi u jednom ciklusu, što se radi ili njihovim izvršavanjem istovremeno ili korišćenjem praznina između dve uzastopne operacije koje se stvaraju zbog kašnjenja. Postoji jedna specifična nit izvršenja procesa i razlikuje se od konkurentnosti po tome što ona uključuje dodjeljivanje više niti jezgru CPU-a u strogoj alternaciji. Pojedinosti o nitima će biti pojašnjene u narednim polgavljima rada.

Paralelizam podataka je tip paralelizma u kome nezavisni procesi primjenjuju istu operaciju na različite elemente skupa podataka. Svi taskovi mogu da se izvršavaju konkurentno. U višeprocesorskom sistemu koji izvršava jedan skup instrukcija, paralelizam podataka se postiže kada svaki procesor obavlja isti zadatak na različitim distribuiranim podacima. U nekim situacijama, jedna izvršna nit kontroliše operacije nad svim podacima. U drugim, različite niti kontrolišu operaciju, ali izvršavaju isti kod.

Funkcionalni paralelizam je tip paralelizma u kome nezavisni podzadaci izvršavaju funkcije nad istim ili različitim elementima podataka. Stepen konkurentnosti je limitiran brojem konkurentnih podzadataka. Uobičajeni tip ovakvog paralelizma je *pipeline*, koji se sastoji od premještanja jednog skupa podataka kroz niz odvojenih zadataka gdje svaki zadatak može da se izvrši nezavisno od drugih.

# Strategije paralelnog programiranja

U savremenom paralelnom računarstvu, prisutno je, u manjoj ili većoj mjeri, četiri praktična pristupa, i to:

* Proširenje kompajlera u smislu dodavanja mogućnosti da sekvencijalne programe automatski prevodi u paralelne
* Proširenje postojećeg jezika dodavanjem novih paralelnih operacija
* Dodavanje novog paralelnog sloja na postojeći sekvencijalni jezik
* Uvođenje potpuno novog jezika koji prirodno podržava paralelizam

Kod proširenja kompajlera, zadatak je modifikacija postojećeg kompajlera dodavanjem mogućnosti da automatski detektuje paralelizam u sekvencijalnim programima i da odatle proizvede paralelni izvršni fajl. Prednost ovakvog pristupa je ta da programeri ne bi morali da paralelizuju svoj kôd, niti da uče paralelno programiranje, već samo da nastave da koriste jednostavnije sekvencijalne jezike, a da paralelizaciju ostave kompajleru i operativnom sistemu. Mane ovog pristupa su što, ukoliko je programer zakomplikovao kod, kompajler sa velikom vjerovatnoćom neće moći ni da pronađe potencijalni paralelizam. Dakle, ovaj pristup funkcioniše isključivo kod jednostavnih konstrukcija, petlji isl. Razvijeni su mnogi eksperimentalni kompajleri ovog tipa, ali nijedno rešenje nije produkcionog kvaliteta.

Najlakši, najjeftiniji i najpopularniji pristup paralelnim programiranju je putem proširenja postojećeg jezika jer zahtjeva samo razvoj biblioteke rutina. Svodi na dodavanje paralelnih funkcija u postojeći sekvencijalni jezik, uključujući funkcije za kreiranje i terminiranje paralelnih procesa, njihovu sinhronizaciju, međusobnu komunikaciju isl. Primjeri iz prakse su MPI, PVM, POSIX niti, OpenMP idr. Međutim, mana je u tome što kompajleri ne učestvuju u generisanju paralelnog koda, niti omogućavaju hvatanje grešaka. Zbog toga je otežano otklanjati greške, čak i u jednostavnim programima.

Jedan od navedenih načina je dodavanje paralelnog sloja sekvencijalnom jeziku. Paralelni program možemo da posmatramo kao da je dvoslojan. Donji sloj je jezgro u kome proces manipuliše sopstevnom porcijom podataka da bi proizveo svoju porciju rezultata. Ovaj sloj može da se implementira u postojećem sekvencijalnom programskom jeziku. Gornji sloj kontroliše kreiranje i sinhronizaciju procesa, kao i dijeljenje podataka među procesima. Paralelni sloj može da bude isprogramiran nekim paralelnim jezikom. Razvijeno je nekoliko istraživačkih prototipova, ali nijedan komercijalni sistem ovog tipa nije zaživio.

Posljednji pristup je razvijanje paralelnog programskog jezika. Primer je jezik Occam, sa potpuno novom sintaksom, koji podržava i paralelno i sekvencijalno izvršavanje procesa. Drugi način je dodavanje paralelnih konstrukcija u već postojeći programski jezik. Primeri su High Performance Fortran i C\* te kompajlersko rešenje CUDA kompanije nVidia, koje dodaje specijalne instrukcije za programiranje grafičkih procesora. Prednost se ogleda u tome što programer predočava paralelizam samom kompajleru, što povećava vjerovatnoću da će izvršni program dostići visoke performanse. Nedostatak je da se zahtjeva razvoj novih kompajlera. Proizvođačima su potrebne godine za razvoj kvalitetnog kompajlera za svoj paralelni sistem. Drugo, novi jezici možda i neće biti standardizovani i onda proizvođači odlučuju da ne prave kompajler za te jezike na svojim mašinama.

Dok se rad na razvijanju paralelizujućih kompajlera i paralelnih programskih jezika visokog nivoa nastavlja, u praksi najčešće korišten pristup ostaje upotreba postojećeg jezika sa paralelnim konstrukcijama niskog nivoa. MPI, pthreads i OpenMP dominiraju savremenim svijetom paralelnog računarstva. Dobija se prilično visoka efikasnost, kao i portabilnost koda, ali po cijenu nešto težeg kodiranja i otklanjanja grešaka.

## Literatura

[1] Miloš Ivanović, *Paralelno programiranje – skripta sa primjerima*, Prirodno-matematički fakultet Univerziteta u Kragujevcu, Kragujevac, 2016