**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

Paralelno programiranje

2. domaća zadaća - *Connect 4 MPI*

Filip Voska

0036467446

Sadržaj

[1. Faze razvoja 2](#_Toc451884581)

[1.1. Podjela 2](#_Toc451884582)

[1.2. Komunikacija 3](#_Toc451884583)

[1.3. Aglomeracija 3](#_Toc451884584)

[1.4. Pridruživanje 3](#_Toc451884585)

[2. Rezultati mjerenja 4](#_Toc451884586)

**ZAGREB, Svibanj 2016.**

# Faze razvoja

Program je implementiran korištenjem C# programskog jezika uz korištenje [MPI.NET](http://www.crest.iu.edu/research/mpi.net/) biblioteke. Prvo je implementirana (vlastita) slijedna inačica algoritma pretraživanja prostora stanja koja je naknadno paralelizirana.

Algoritam izgrađuje stablo prostora stanja igra od nekog trenutnog stanja pa sve do neke zadane dubine stabla. Svaki čvor stabla predstavlja jedno stanje igre (ploče). Iz jednog stanja u drugo se prelazi umetanjem oznake ili igrača ili računala u ploču (odgovara jednom potezu u igri). Igra može imati proizvoljan broj redaka i stupaca, a pretpostavljene dimenzije su 7 stupaca i 6 redaka.

Kako bi se odlučio potez računala, algoritam rekurzivno prolazi kroz stablo i svakom stanju dodjeljuje određenu vrijednost u ovisnosti o „kvaliteti“ poteza iz stanovišta računala.

Kako bi se objasnio paralelni algoritam, praktično je pogledati 4 faze razvoja paralelnog algoritma.

## Podjela

Ideja je podijeliti zadatke između procesa tako da svaki obrađuje jedan dio stabla, odnosno da je njegov zadatak provedba slijednog algoritma na podstablima.

Prvo se zadaje ukupna dubina do koje se u stablu ide prilikom izračuna kvalitete poteza. Zatim se određuje koliko bi zadataka minimalno trebalo stvoriti na temelju broja procesa. Konkretno, uzima sa kvadrat broja procesa. Nakon toga se gleda koliko se duboko u stablo mora ići da bi broj listova stabla bio veći ili jednak minimalnom broju procesa. Ta dubina predstavlja početno stablo koje će se obrađivati slijedno na procesu voditelju. Sada se može izračunati koliko je još potrebno ići u dubinu kako bi se došlo do zadane dubine. Iz svakog lista početnog stabla se generira po jedan zadatak. Računanje u svakom zadatku je izvršavanje slijednog algoritma gdje je početno stanje tog lista, a dubina do koje se ide je izračunata preostala dubina.

Primjer podjele:

Dimenzije: 7 stupaca, 6 redaka  
Željena dubina: 7  
Broj procesa: 4  
Preporučeni broj zadataka: 4\*4 = 16  
Dubina do koje se mora ići u početnom stablu: 2 (0 -> 1 list, 1 -> 7 lista , 2 -> 49 listova)  
Konačni broj zadataka: 49 (svaki list)  
Dubine podstabala: 7 – 2 = 5  
Podjela zadataka po procesima: p#0 -> 13, p#1 -> 12, p#2 -> 12, p#3 -> 12

U ovom primjeru podjele, proces voditelj će izgraditi početno stablo do dubine 2, što će izgenerirati 49 zadataka (po jedan za svaki list). Svaki zadatak će zatim izgraditi podstablo dubine 5 na kojemu se provodi slijedni algoritam.

## Komunikacija

Nakon što proces voditelj izgradi početno stablo i zadatke (u navedenom primjeru, do dubine 2), zadatci se šalju procesima radnicima. Proces voditelj je ujedno i radnik jer inače ne bi imao nikakvog posla dok čeka radnike da provedu račun.

Radnici po primitku zadataka provode slijedni algoritam (u navedenom primjeru, do dubine 5). Nakon što je pojedini radnik obavio sve svoje zadatke, šalje rezultate natrag voditelju.

Voditelj prikuplja sve rezultate te unosi odgovarajuće vrijednosti u listove početnog podstabla te dovršava odlučivanje poteza računala provođenjem slijednog algoritma na početnom podstablu.

## Aglomeracija

Potrebno je postići dobar balans između broja zadataka i broj procesa. Veliki broj zadataka ujedno podrazumijeva i veću dubinu početnog stabla, što znači da će proces voditelj prilikom generiranje zadataka te prilikom prikupljanja zadataka i provođenja slijednog algoritma nad početnim stablom imati više posla. Prilikom testiranja se pokazalo da performanse padaju kako se povećavaju broj zadataka pojedinog procesa i dubina početnog podstabla. Pogledajmo primjer postavki algoritma u Tablici 1.

Tablica 1 Primjer postavki algoritma

|  |  |  |
| --- | --- | --- |
|  | Postavke #1 | Postavke #2 |
| Željena dubina | 7 | 7 |
| Broj procesa | 4 | 4 |
| Broj zadataka | 49 (~12 po procesu) | 343 (~85 po procesu) |
| Dubina početnog stabla | 2 | 3 |
| Dubina podstabla u zadatcima | 5 | 4 |
| Vrijeme odluke [s] | **7.68** | **12.25** |

Na primjeru se vidi da veća dubina početnog stabla i veći broj zadataka po procesu rezultira sporijim radom cjelokupnog algoritma, u navedenom primjeru čak ~40% sporije. Za ovu konkretnu izvedbu algoritma, čini se idealnim održavati broj zadataka po procesu na oko 10, što za manji broj procesa (do 8) uglavnom znači odabir dubine početnog stabla 2 (tj. 49 zadataka ukupno). Dubina početnog stabla 3 (343 zadataka) je preporučljiva tek kada broj procesa prelazi nekoliko desetina (20-30).

Kako bi se smanjila količina komunikacije, rezultati od radnika se ne šalju pojedinačno za svaki odrađeni zadatak, već skupno u jednoj poruci nakon što radnik odradi sve zadatke. To je moguće jer voditelj tako i onako ne može krenuti s provedbom algoritma na početnom stablu dok ne dobije sve rezultate od svih radnika, a on je ujedno i sam radnik pa je poželjno da ga se ne prekida tokom rada, već se svi rezultati prikupe kada su svi radnici (uključujući i voditelja) gotovi.

## Pridruživanje

Proces voditelj generirane zadatke dodjeljuje procesima radnicima (i samome sebi) po kružnom principu (za P procesa, zadatak broj X se dodjeljuje procesu X mod P).

# Rezultati mjerenja

## Metodologija

Mjerenja su izvršena na računalu s 4-jezgrenim 3.4GHz procesorom uz uključeni HyperThreading. U labosu je bilo nekih tehničkih problema u vezi korištene MPI tehnologije/biblioteka pa mjerenja nije bilo moguće provesti u labosu. Međutim, pokazalo se da se HyperThreading dobro nosi s implementacijom i za veći broj procesa od fizičkog broja jezgri.

Algoritam je postavljen da radi do dubine 7. Mjerenje se pokreće nakon što korisnik odigra svoj prvi potez, a zaustavlja se u trenutku kada se donese odluka o potezu računala. Takvo mjerenje uključuje svo čekanje, rad i komunikaciju koje je potrebno za donošenje odluke o potezu računala.

Za svaki pojedini broj procesa/zadataka izvršena su tri mjerenja te je pri konačnom prikazu podataka korištena srednja vrijednost tih triju mjerenja.

## Broj zadataka

Kao što je već spomenuto ranije, idealno je da broj zadataka koje obrađuje pojedini proces bude oko 5 do 10. Konkretni broj ukupnog broja zadataka u ovisnosti o korištenom broju procesa prikazan je u Tablici 2. Mjerenje za 8 procesa je izvršeno dva puta, jednom uz ukupan broj zadataka 49, a jednom uz 343.

Tablica 2 Broj ukupnih zadataka u ovisnosti o broju procesa

|  |  |
| --- | --- |
| **Broj procesora** | **Broj zadataka** |
| 1 | 1 |
| 2 | 7 |
| 3 | 49 |
| 4 | 49 |
| 5 | 49 |
| 6 | 49 |
| 7 | 49 |
| 8 | 49 |
| 8 | 343 |

## Rezultati mjerenja

Vrijednosti rezultata mjerenja mogu se vidjeti u Tablici 3 i Tablici 4.

Tablica 3 Vremena

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Broj procesora** | **Broj zadataka** | **Mjerenje #1** | **Mjerenje #2** | **Mjerenje #3** | **Vrijeme [s]** |
| 1 | 1 | 28,50 | 29,08 | 28,78 | 28,79 |
| 2 | 7 | 16,40 | 16,49 | 16,41 | 16,43 |
| 3 | 49 | 10,00 | 9,88 | 9,51 | 9,80 |
| 4 | 49 | 7,78 | 7,65 | 7,91 | 7,78 |
| 5 | 49 | 5,98 | 5,96 | 5,91 | 5,95 |
| 6 | 49 | 5,40 | 5,41 | 5,41 | 5,41 |
| 7 | 49 | 4,34 | 4,35 | 4,30 | 4,33 |
| 8 | 49 | 4,33 | 4,35 | 4,32 | 4,33 |
| 8 | 343 | 9,64 | 9,67 | 9,67 | 9,66 |

Tablica 4 Učinkovitost i ubrzanje

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Broj procesora** | **Broj zadataka** | **Vrijeme [s]** | **Savršeno [s]** | **Učinkovitost** | **Sav. Učink.** | **Ubrzanje** | **Sav. Ubrz.** |
| 1 | 1 | 28,79 | 28,79 | 100,00% | 100,00% | 1,00 | 1,00 |
| 2 | 7 | 16,43 | 14,39 | 87,59% | 100,00% | 1,75 | 2,00 |
| 3 | 49 | 9,80 | 9,60 | 97,95% | 100,00% | 2,94 | 3,00 |
| 4 | 49 | 7,78 | 7,20 | 92,50% | 100,00% | 3,70 | 4,00 |
| 5 | 49 | 5,95 | 5,76 | 96,76% | 100,00% | 4,84 | 5,00 |
| 6 | 49 | 5,41 | 4,80 | 88,74% | 100,00% | 5,32 | 6,00 |
| 7 | 49 | 4,33 | 4,11 | 94,97% | 100,00% | 6,65 | 7,00 |
| 8 | 49 | 4,33 | 3,60 | 83,04% | 100,00% | 6,64 | 8,00 |
| 8 | 343 | 9,66 | 3,60 | 37,25% | 100,00% | 2,98 | 8,00 |

## Grafovi učinkovitosti i ubrzanja te komentar

Vidljivo je da se HyperThreading dosta dobro nosi sa većim brojem procesa. Vidljiv je pad učinkovitosti i ubrzanja za broj procesa 6, 7 i 8 uz 49 zadataka u usporedbi s idealnom učinkovitošću i ubrzanjem.

Znatan pad ubrzanja i učinkovitosti vidljiv je kada povećamo broj zadataka s 49 na 343, što je rezultat veće komunikacije, dugotrajnijeg generiranje zadataka te dugotrajnije obrade prikupljenih rezultata od strane radnika na procesu voditelju.

Zanimljivo je uočiti da su učinkovitost i ubrzanje veći za neparan broj procesa.