Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Luka Bajić

Inženiring vzporednih algoritmov

diplomsko delo

visokošolski študijski program
prve stopnje
računalništvo in informatika

Mentor: doc. dr. Jurij Mihelič

Ljubljana, 2024

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Kandidat: Luka Bajić
Naslov: Inženiring vzporednih algoritmov
Vrsta naloge: Diplomska naloga na visokošolskem programu prve stopnje Računalništvo in informatika
Mentor: doc. dr. Jurij Mihelič

Opis:
V diplomski nalogi je študent preizkusil različne tehnike za povzporejanje algoritma za izračun DTW (dynamic time warping) razdalje, ki sem jih nato še dopolnil. Predvidevam, da bi enake tehnike bile uporabne tudi za nekatere druge probleme, kot je razdalja LCS (longest common subsequence) ali Levenshteinova razdalja. V nalogi je torej treba te tehnike sprogramirati za omenjena problema (lahko le en problem, če bo veliko dela), nato pa jih smiselno eksperimentalno ovrednotiti.

Title: Parallel algorithm engineering

Description:
opis diplome v angleščini

Zahvaljujem se mentorju doc. dr. Juriju Miheliču za pomoč pri izdelavi diplomske naloge.

Kazalo

Povzetek

Abstract

1 Uvod **1**

1.1 Motivacija 1

1.2 Struktura 1

2 Osnovna implementacija **3**

2.1 LCS 3

2.2 Levenshteinova razdalja 3

3 Vzporedni algoritmi **5**

3.1 Pristop naprej-nazaj 5

3.2 Diagonalni pristop 6

4 Rezultati **7**

4.1 Podatki 7

5 Zaključki **9**

5.1 Sklep 9

5.2 Nadalnji razvoj 9

Seznam uporabljenih kratic

kratica	angleško	slovensko
LCS	longest common subsequence	najdaljša skušna podsekvenca
DBMS	database management system	sistem za upravljanje podatkovnih baz
SVM	support vector machine	metoda podpornih vektorjev

Povzetek

Naslov: Inženiring vzporednih algoritmov

Avtor: Luka Bajić

V vzorcu je predstavljen postopek priprave diplomskega dela z uporabo okolja LaTeX. Vaš povzetek mora sicer vsebovati približno 100 besed, ta tukaj je odločno prekratek. Dober povzetek vključuje: (1) kratek opis obravnavanega problema, (2) kratek opis vašega pristopa za reševanje tega problema in (3) (najbolj uspešen) rezultat ali prispevek diplomske naloge.

Ključne besede: vzporedni algoritmi, večnitnost, Levenshteinova razdalja, LCS.

Abstract

Title: Parallel Algorithm Engineering

Author: Luka Bajić

This sample document presents an approach to typesetting your BSc thesis using LaTeX. A proper abstract should contain around 100 words which makes this one way too short.

Keywords: computer, computer, computer.

Poglavje 1

Uvod

V diplomskem delu so opisani različni pristopi za optimizacijo algoritmov za računanje Levenshteinove razdalje in najdaljše skupne podsekvence (??) (LCS - longest common subsequence) z uporabo vzporednega procesiranja.

1.1 Motivacija

1.2 Struktura

Drugo poglavje opiše algoritma, njuno praktično uporabnost in njuno implementacijo z uporabo dinamičnega programiranja. Tretje poglavje predstavi dva glavna pristopa paralelizacije - diagonalni in naprej-nazaj. V četrtem poglavju so prikazani rezultati meritev na različnih velikostih vhodnih podatkov. V petem poglavju sledijo še sklepi in ideje za nadalnji razvoj.

Poglavje 2

Osnovna implementacija

V tem poglavju je predstavljena implementacija algoritma za računanje Levenshteinove razdalje in LCS razdalje z uporabo dinamičnega programiranja. V tem kontekstu je namenjena predvsem za časovno primerjavo med sekvenčnim in vzporednim algoritmom.

2.1 LCS

2.2 Levenshteinova razdalja

Poglavje 3

Vzporedni algoritmi

Osnovna struktura vzporednega algoritma za oba problema je enaka kot njuna rešitev iz prejšnjega poglavja - še vedno uporabljamo dvodimenzionalno tabelo za hranjenje vmesnih vrednosti, torej je prostorska zahtevnost enaka - O(n*m). Razlika je samo v tem, da vrednosti, ki niso medsebojno odvisne, lahko računamo vzporedno z uporabo koncepta večnitnosti (angl. multithreading). Za to obstajata dva glavna pristopa - diagonalni in naprej-nazaj.

3.1 Pristop naprej-nazaj

V prejšnjem poglavju smo po tabeli iterirali od leve proti desni, ter od vrha navzdol, torej za i = 0, 1, ..., n in j = 0, 1, ..., m, nakar smo prišli do končne rešitve na poziciji (i, j). Brez težav lahko počnemo ravno obratno - iteriramo od i = n, n-1, ... 0 in j = m, m-1, ..., 0 in v tem primeru dobimo rešitev na poziciji (0, 0). Izkaže se, da lahko z uporabo dveh niti poženemo oba načina istočasno. Tabelo v abstraktnem smislu prepolovimo na dva dela - ena nit računa zgornjo polovico, druga pa spodnjo. Na ta način, vsaj teoretično, dosežemo dvakratno pohitritev.

3.2 Diagonalni pristop

Težava pristopa naprej-nazaj je, da dejansko uporablja samo dve niti, oziroma ga je težko dodatno paralelizirati, ker se morajo niti v tem primeru medseboj čakati in ne dosežemo maksimalne optimalnosti, oziroma se čas morda celo poslabša zaradi prevelikega overheada.

Zato če želimo uporabiti več kot dve niti, raje koristimo diagonalni pristop, ki nam teoretično omogoča hkratno uporabo do k niti (k = dolžina diagonale = dolžina krajšega izmed dveh nizov). Ta pristop deluje, ker se izkaže, da so elementi na diagonalah vedno medsebojno neodvisni, algoritem namreč dostopa samo do podatkov v celicah (i, j-1), (i-1, j) in (i-1, j-1), paziti moramo samo, da se diagonale računajo v pravem vrstnem redu, torej da začnemo s celico (0, 0) in končamo s celico (n, m). To najlažje dosežemo tako, da z zunanjo zanko iteriramo po diagonalah (teh je vedno n+m-1) in nato v vsaki iteraciji vzporedno računamo vse celice na posamezni diagonali.

[slikovni prikaz]

Poglavje 4

Rezultati

4.1 Podatki

Poglavje 5

Zaključki

5.1 Sklep

5.2 Nadalnji razvoj

Kot je razvidno iz poglavja 4., smo z vzporednimi algoritmi dosegli občutno pohitritev že na običajnih osebnih računalnikih. V nadalnje bi bilo zanimivo preveriti ali je čas izvajanja morda še krajši na zmogljivejših sistemih z ogromno količino jeder, oziroma ali je mogoče algoritme še dodatno prilagoditi za tovrstne sisteme.