ΠΑΝΕΠΙΣΤΗΜΙΟ ΜΑΚΕΔΟΝΙΑΣ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΤΜΗΜΑΤΟΣ ΕΦΑΡΜΟΣΜΕΝΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΑΡΑΛΛΗΛΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΜΕ ΧΡΗΣΗ OpenMP

Διπλωματική Εργασία

του

Κοντογιάννη Γεώργιου

Θεσσαλονίκη, Φεβρουάριος 2021

ΠΑΡΑΛΛΗΛΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΜΕ ΧΡΗΣΗ OpenMP

Κοντογιάννης Γεώργιος

Δίπλωμα Πολιτικού Μηχανικού, ΑΠΘ, 2016

Διπλωματική Εργασία

υποβαλλόμενη για τη μερική εκπλήρωση των απαιτήσεων του

ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΤΙΤΛΟΥ ΣΠΟΥΔΩΝ ΣΤΗΝ ΕΦΑΡΜΟΣΜΕΝΗ ΠΛΗΡΟΦΟΡΙΚΗ

Επιβλέπων Καθηγητής
Μαργαρίτης Κωνσταντίνος

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την ηη/μμ/εεεε

Ονοματεπώνυμο 1	Ονοματεπώνυμο 2	Ονοματεπώνυμο 3
.....	.....	.....

Κοντογιάννης Γεώργιος

.....

Η σύνταξη της παρούσας εργασίας έγινε στο LaTeX

Περίληψη

Αντικείμενο της διπλωματικής εργασίας είναι η μελέτη του OpenMP, ενός προτύπου παράλληλου προγραμματισμού, που δίνει στο χρήστη τη δυνατότητα αναπτύξης παράλληλων προγραμμάτων για συστήματα μοιραζόμενης μνήμης, τα οποία είναι ανεξάρτητα από τη αρχιτεκτονική του συστήματος και έχουν μεγάλη ικανότητα κλιμάκωσης[2].

Σκοπός της εργασίας είναι η συνοπτική ανακεφαλαίωση των βασικών χαρακτηριστικών των παλαιών εκδόσεων (OpenMP 2.5), η μελέτη και περιγραφή των κύριων χαρακτηριστικών των νεότερων (3.0 και 4.5) καθώς και η υλοποίηση αλγορίθμων σειριακά και παράλληλα εκτελέσιμων, με σκοπό τη συγκριτική μελέτη των παραλλαγών του κάθε προβλήματος για την εξαγωγή συμπερασμάτων. Για την παράλληλη υλοποίηση θα γίνει χρήση της Διεπαφής Προγραμματισμού Εφαρμογών (Application Programming Interface - API) OpenMP, με χαρακτηριστικά που εισήχθησαν στις εκδόσεις OpenMP 3.0 που δημοσιεύθηκε το 2008 και OpenMP 4.5 που δημοσιεύθηκε 2015 καθώς και χαρακτηριστικά παλαιότερων εκδόσεων[3].

Τον Μάιο του 2008 κυκλοφόρησε η έκδοση του OpenMP 3.0. Στην κυκλοφορία συμπεριλήφθηκε για πρώτη φορά η έννοια των διεργασιών (Tasking) αλλά και βελτιώσεις στην υποστήριξη της διεπαφής μέσω της C++. Αποτελεί την πρώτη ενημέρωση μετά την έκδοση 2.5 με σημαντικές βελτιώσεις. Το 2011 κυκλοφόρησε η OpenMP 3.1 χωρίς αξιοσημείωτες νέες προσθήκες. Νέα χαρακτηριστικά ωστόσο, εισήχθησαν στο OpenMP 4.0 που κυκλοφόρησε τον Ιούλιο του 2013, όπου έγινε υποστήριξη της αρχιτεκτονικής cc-NUMA, του ετερογενούς προγραμματισμού, της διαχείρισης σφαλμάτων στην περιοχή παράλληλου κώδικα και της διανυσματικοποίησης μέσω SIMD. Τον Ιούλιο του 2015 σημαντική βελτίωση έγινε στα παραπάνω χαρακτηριστικά με την έκδοση OpenMP 4.5[4].

Τα προαναφερθέντα χαρακτηριστικά χρησιμοποιήθηκαν για την υλοποίηση αλγορίθμων σε διάφορες παραλλαγές, με σκοπό τη συγκριτική μελέτη τους για την εξαγωγή συμπερασμάτων αναφορικά με τη βελτίωση της απόδοσης σε σχέση με τη σειριακή υλοποίηση, τη μεταξύ τους σύγκριση καθώς επίσης, την αξιολόγηση της ευχρηστίας της υλοποίησής τους. Στόχος της έρευνας είναι η συλλογή και καταγραφή παρατηρήσεων που προκύπτουν σε κάθε υλοποίηση, για την καλύτερη κατανόηση των εννοιών του

παράλληλου προγραμματισμου.

Για την ικανότητα παραλληλοποίησης του κώδικα, απαιτούνται αλγόριθμοι που αποτελούνται απο διεργασίες ανεξάρτητες μεταξύ τους και ικανές να εκτελεστούν ταυτόχρονα, σε διαφορετικούς επεξεργαστές. Τέτοιοι αλγόριθμοι είναι οι εξής:

- μετασχηματισμός **Fourier - (Discrete Fourier Transform)**,
- ταξινόμηση **Mergesort**,
- ταξινόμηση **Quicksort**,
- διεργασία **Producer-Consumer**,
- υπολογισμός π,
- υπολογισμός πρώτων αριθμών,
- υπολογισμός εσωτερικού γινομένου,
- πολλαπλασιασμός πινάκων,
- Single precision A X plus Y - **SAXPY**,
- **Linked list** traversal

Υλοποιήσεις των παραπάνω προβλημάτων χρησιμοποιούνται στα πλαίσια της παρούσας εργασίας. Δημιουργήθηκαν παραλλαγές του κάθε προβλήματος που χρησιμοποιούν την κεντρική μονάδα επεξεργασίας (**CPU**) για σειριακή και παράλληλη εκτέλεση. Όπου είναι εφικτό υλοποιείται παραλλαγή για εκτελεσή μέσω της μονάδας επεξεργασίας της κάρτας γραφικών (**GPU**). Οι χρονικές καταγραφές συγκρίνονται μεταξύ τους. Ακόμη, γίνεται αξιολόγηση της ευχρηστίας για την υλοποίηση της κάθε παραλλαγής καθώς και καταγραφή παρατηρήσεων που τυχόν προέκυψαν.

Λέξεις Κλειδιά: Παράλληλος Προγραμματισμός, Παραλληλοποίηση, OpenMP, accelerators, offloading, vectorization, SIMD, OpenMP4.5, UDRs

Abstract

Keywords:

Ευχαριστίες

Εκφράζω τις θερμές μου ευχαριστίες στον επιβλέποντα καθηγητή κ. Κωνσταντίνο Μαργαρίτη, για την ουσιαστική του συνεισφορά στην εκπόνηση της παρούσας εργασίας.

Περιεχόμενα

Κατάλογος Εικόνων (αν υπάρχουν)

Κατάλογος Πινάκων (αν υπάρχουν)

Λίστα Συμβολισμών

1 Υλοποιημένα παραδείγματα[1]

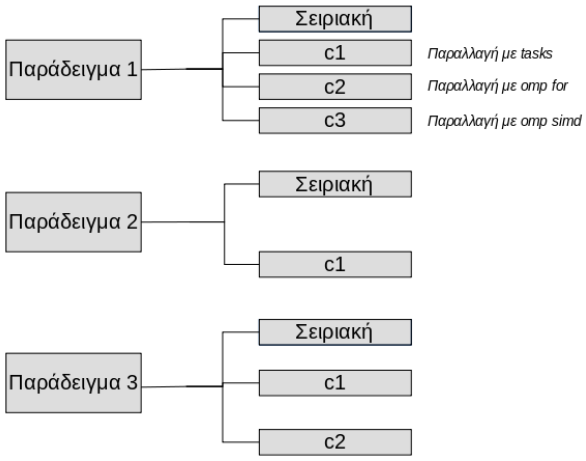
Οι ενότητες που ακολουθούν αφορούν τη μελέτη της θεωρίας που αναφέρθηκε στα προηγούμενα κεφάλαια, μέσω της υλοποίησης παραδειγμάτων. Τα προβλήματα που θα αναπτυχθούν, αντιπροσωπεύουν κατηγορίες βασικών προβλημάτων που συναντώνται συχνά σε ζητήματα παράλληλου προγραμματισμού. Για κάθε πρόβλημα υλοποιείται η σειριακή μέθοδος επίλυσης και παραλλαγές με παραλληλισμό που κυριώς χρησιμοποιεί χαρακτηριστικά που εισήχθησαν στις εκδόσεις μετά την 2.5. Στόχος είναι η σύγκριση των διαφορετικών παραλλαγών του ίδιο προβλήματος σε επίπεδο χρονικών επιδόσεων με χρήση διαγραμμάτων και πινάκων, αλλά και την εξαγωγή παρατηρήσεων και συμπερασμάτων που προκύπτουν από αυτές. Σε κάθε ανάλυση παρατίθενται και τμήματα του πηγιαίου κώδικα της επιμέρους παραλλαγής.

Τα παραδείγματα που χρησιμοποιήθηκαν αναφέρονται επιγραμματικά παρακάτω και με λεπτομερέστερη περιγραφή στα αντίστοιχα κεφάλαια τους:

- Υπολογισμός π
- Linked list traversal
- SAXPY
- Υπολογισμός πρώτων αριθμών
- Πολλαπλασιασμός πινάκων
- Quicksort
- Mergesort
- Producer-Consumer
- Discrete Fourier Transform

1.1 Μεθοδολογία σύνταξης προβλημάτων

Για όλα τα προβλήματα χρησιμοποιήθηκε η ίδια μεθοδολογία επίλυσης. Τα προβλήματα έχουν χωριστεί σε ξεχωριστούς φακέλους ανα πρόβλημα, όπου κάθε φάκελος περιέχει το βασικό πηγαίο κώδικα που ξεκινάει το πρόγραμμα και υποφακέλους που περιέχουν τις επιμέρους παραλλαγές. Με τη βοήθεια της παραδοχής κοινών ονομάτων και κοινού signature, το compile όλων των διαφορετικών παραλλαγών είναι το ίδιο, με τη μόνη διαφορά να ειναι ο φάκελος στον οποίο θα κάνει link ο μεταγλωττιστής. Με αυτό, τον τρόπο, ο βασικός κορμός του προβλήματος παραμένει ίδιος και αποφεύγεται ο διπλότυπος κώδικας.



Σχήμα 1: Διάρθρωση παραδειγμάτων στο *github.com*

Για παράδειγμα, έστω οτι επιλύεται ένα πρόβλημα που ονομάζεται *Foo* με διαφορετικές παραλλαγές μιας συνάρτησης *fun()*. Τότε δημιουργείται ένα κεντρικό αρχείο *run.cpp* που περιέχει τη *main()* και σε κάθε υποφάκελο (*c1*, *c2*) ένα αρχείο *test.cpp* με διαφορετική παραλλαγή της *fun()*. Τότε στο αρχείο run.cpp γίνεται *#include "test.hpp"* και η εντολες για *compile* θα είναι οι εξής:

g++ run.cpp ./c1/test.hpp -I c1

g++ run.cpp ./c2/test.hpp -I c2

1.2 Αναφορά αρχιτεκτονικής μηχανήματος

Τα προβλήματα που ακολουθούν εκτελέστηκαν σε μηχάνημα λειτουργικό *linux* και μεταγλωττιστή *gcc-7.5.0*. Οι προδιαγραφές υλικού του μηχανήματος που εκτελέστηκαν τα προβλήματα, αναφέρονται στο παρακάτω παράδειγμα:

Πίνακας 1: Χαρακτηριστικά Μηχανήματος Εκτέλεσης

Architecture	x86_64
CPU op-mode(s)	32-bit, 64-bit
CPU(s)	16
Thread(s) per core	1
Core(s) per socket	8
Socket(s)	2
NUMA node(s)	4
Model name	AMD Opteron(tm) Processor 6128 HE
L1d cache	64K
L2 cache	512K
L3 cache	5118K
Memory	16036

References

[1] . 0.

[2] K.Margaritis. Introduction to openmp. pdplab.it.uom/teaching/llnl-gr/OpenMP.html#Abstract. Accessed: 2020-06-25.

[3] C. T. Ruud van der Pas, E. Stotzer. *Using OpenMP. The Next Step: affinity, accelerators, tasking, SIMD*, page 59. The MIT Press, 2017.

[4] C. T. Ruud van der Pas, E. Stotzer. *Using OpenMP. The Next Step: affinity, accelerators, tasking, SIMD*, page 20. The MIT Press, 2017.