Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών Ε.Μ.Π.

ΣΥΣΤΗΜΑΤΑ ΠΑΡΑΛΛΗΛΗΣ ΕΠΕΞΕΡΓΑΣΙΑΣ
2016-2017

Άσκηση 2
Παράλληλη επίλυση του αλγορίθμου Floyd-Warshall σε πολυπύρηνες αρχιτεκτονικές

Βαρηά Χρυσούλα - 03112105
Κουτσανίτη Ειρήνη - 03112135

Οι υλοποιήσεις των προγραμμάτων έγιναν τόσο με τη χρήση OpenMp όσο και με TBBs. Οι αντίστοιχοι κώδικες περιέχονται στο zip φάκελο μαζί με αντίστοιχα scripts, τα οποία δείχνουν τον τρόπο με τον οποίο μεταγλωττίστηκαν και εκτελέστηκαν.Τα πειραματικά αποτελέσματα των εκτελέσεων των σειριακών εκδόσεων όπως παρουσιάστηκαν στην ενδιάμεση αναφορά είναι τα ακόλουθα:

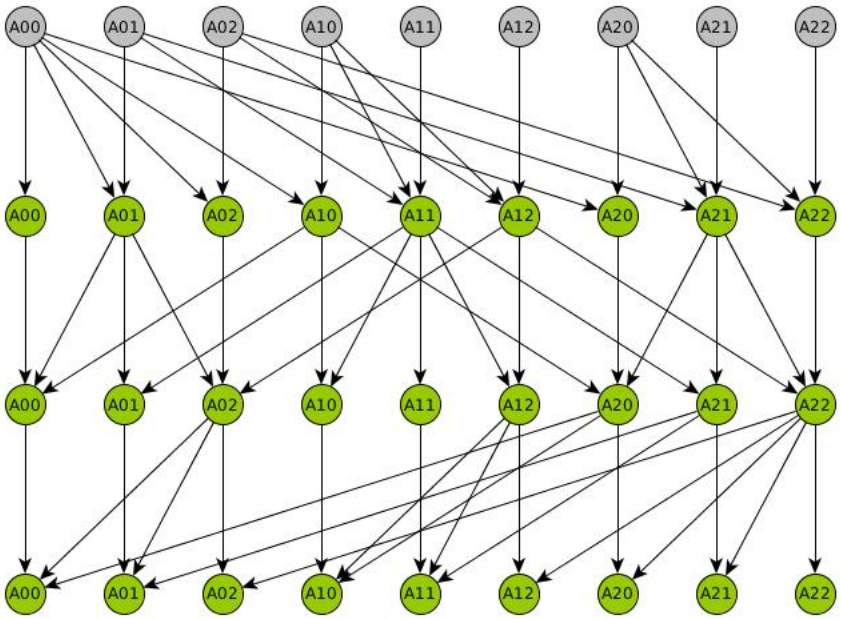
	64×64	1024×1024	2048×2048
fw.c	1.3739	11.22	93.5341
fw_sr.c, B=128	0.6625	4.8154	36.4538
fw_tiled.c, B=128	0.6928	4.8491	35.9686

Υπενθυμίζεται ότι για τις γραφικές παραστάσεις του speedup χρησιμοποιείται πάντα η καλύτερη τιμή για κάθε είσοδο και όχι η τιμή του αντίστοιχου σειριακού αλγορίθμου.

Standard έκδοση αλγορίθμου Floyd-Warshall

Παραλληλοποίηση:

Ο γράφος εξαρτήσεων παρουσιάζεται στο ακόλουθο σχήμα για πλέγμα 2x2:



Σημειώνεται ότι οι τιμές του πίνακα κάθε χρονική στιγμή εξαρτώνται από τις τιμές της προηγούμενης επανάληψης. Στο παράδειγμα των διαφανειών (Υλοποίηση παράλληλων προγραμμάτων σελ.16), η υλοποίηση του σειριακού κώδικα γίνεται με χρήση δύο πινάκων σε κάθε επανάληψη. Αντίθετα στην παράλληλη υλοποίηση χρησιμοποιείται μόνο ένας πίνακας με αποτέλεσμα να δημιουργούνται επιπλέον εξαρτήσεις, οι οποίες δεν παρουσιάζονται στο σχήμα.

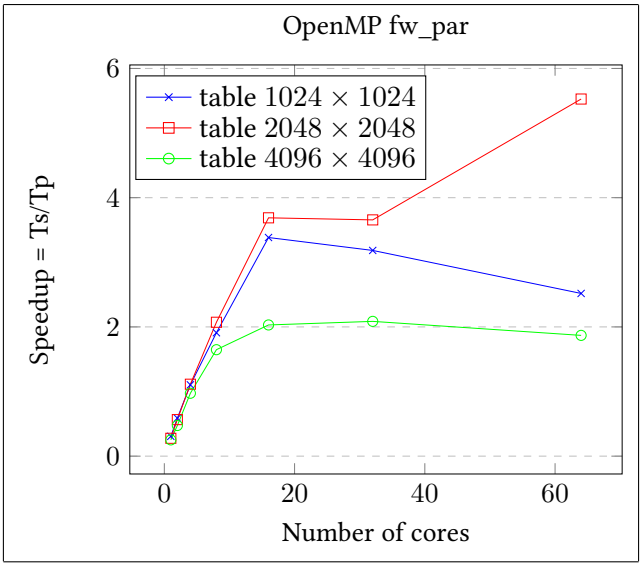
Συγκεκριμένα για το υπολογισμό κάποιας τιμής A[i][j] γίνεται έλεγχος των τιμών A[i][k] και A[k][j]. Οι αναφερθείσες τιμές του πίνακα μπορεί να έχουν ήδη υπολογιστεί από κάποιο άλλο νήμα την ίδια χρονική στιγμή, επομένως το αποτέλεσμα της ανάθεσης τιμής A[i][j] μπορεί να μην είναι ίδιο σε κάθε βήμα στην παράλληλη και τη σειριακή έκδοση του αλγορίθμου. Η ορθότητα της παράλληλης υλοποίησης έγκειται στο γεγονός ότι γίνονται κ έλεγχοι για κάθε τιμή του πίνακα, οι οποίοι διορθώνουν αυτή την ασάφεια και το τελικό αποτέλεσμα είναι ίδιο για τις δύο υλοποιήσεις. Επίσης

πρέπει να αναφερθεί ότι η εκτέλεση του προγράμματος γίνεται σωστά με την προϋπόθεση ότι δεν υπάρχουν αρνητικοί κύκλοι στο γράφημα.

Για την υλοποίηση της παράλληλης έκδοσης openMP χρησιμοποιείται parallel for με private τιμές τις μεταβλητές i,j για κάθε νήμα και shared τη μεταβλητή k του εξωτερικού βρόχου. Το εξωτερικό loop k δεν μπορεί να παραλληλοποιηθεί, ενώ τα δύο εσωτερικά loops των i,j έχουν αυτή την δυνατότητα. Επιλέγεται ωστόσο η παραλληλοποίηση μόνο του μεσαίου loop i, διότι αν παραλληλοποιηθεί και το εσωτερικό loop j αυξάνεται το overhead λόγω της αύξησης του πλήθους των νημάτων και η επίδοση της εκτέλεσης χειροτερεύει.

Ακολουθούν τα αποτελέσματα των πειραματικών μετρήσεων και το διάγραμμα επιτάχυνσης του προγράμματος openMP:

Time(sec)	1024 x 1024	2048 x 2048	4096 x 4096
1	2.1599	17.4256	143.0654
2	1.1254	8.5182	75.8905
4	0.6003	4.3261	36.9569
8	0.3469	2.324	21.8331
16	0.1958	1.3056	17.7156
32	0.2081	1.317	17.2449
64	0.2631	0.8713	19.2454



Παρατηρήσεις:

- Για τη σειριακή έκδοση του αλγορίθμου, έγινε επίσης υλοποίηση με χρήση TBBs, όμως η επίδοση του προγράμματος ήταν σε μεγάλο βαθμό χειρότερη, επομένως επιλέχθηκε να μην παρουσιαστούν οι αντίστοιχες μετρήσεις. Η υλοποίηση του κώδικα ωστόσο περιλαμβάνεται στην αναφορά. Η διαφορά που παρατηρήθηκε στην επίδοση μπορεί να οφείλεται στο γεγονός ότι η openMP υλοποιείται σε χαμηλότερο επίπεδο από ότι τα TBBs, τα οποία είναι υλοποίηση σε software επίπεδο και επομένως για απλές μορφές παραλληλοποίησης όπως η συγκεκριμένη είναι καλύτερη επιλογή η χρήση της openMP.

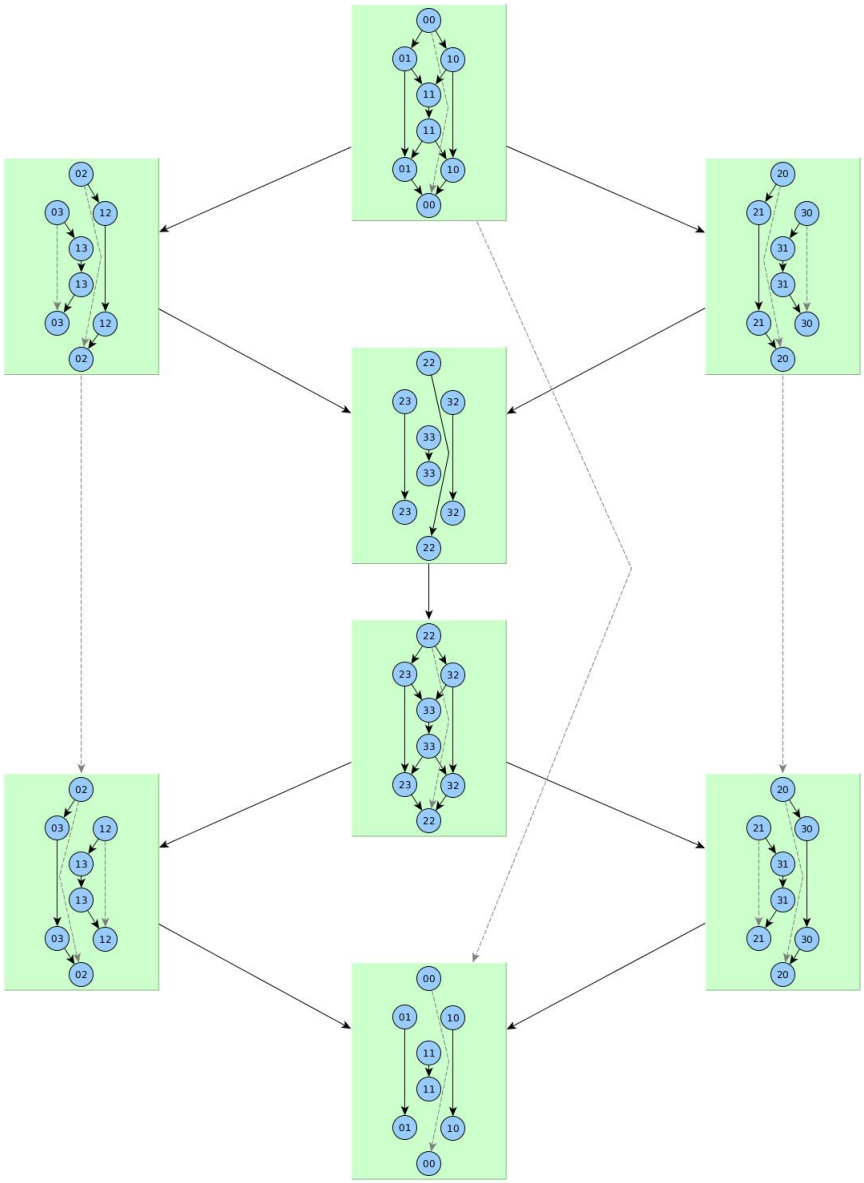
- Επίσης η εκτέλεση του προγράμματος με χρήση affinity δεν παρουσίασε κάποια βελτίωση στην επίδοση.

Recursive έκδοση αλγορίθμου Floyd-Warshall

Παραλληλοποίηση:

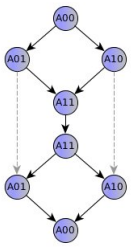
Ο γράφος εξαρτήσεων, για κάθε μπλοκ που υπολογίζεται, παρουσιάζεται παρακάτω για έναν πίνακα, του οποίο το μέγεθος είναι 4 φορές το μέγεθος του μπλοκ.

00	01	02	03
10	11	12	13
20	21	22	23
30	31	32	33



Εναλλακτικά οι εξαρτήσεις μιας επανάληψης παρουσιάζονται περιληπτικά στη συνέχεια για κάθε βήμα της αναδρομής, όπως είχαν περιγραφεί στην ενδιάμεση αναφορά:

1) A00 <- A00, A00
2) A01 <- A00, A01 και A10 <- A10, A00
3) A11 <- A10, A01
4) A11 <- A11, A11
5) A10 <- A11, A10 και A01 <- A01, A11
6) A00 <- A01, A10



Η υλοποίηση της συγκεκριμένης έκδοσης έγινε με χρήση openMP και TBBs. Παραθέτονται και οι δύο υλοποιήσεις προκειμένου να γίνει σύγκριση της επίδοσης τους.

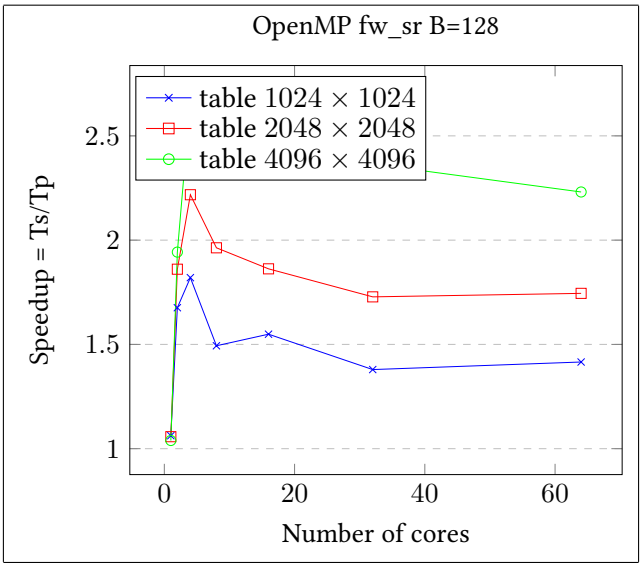
Στην υλοποίηση με openMP αρχικά ορίζεται το τμήμα του παράλληλου κώδικα πριν την κλήση της συνάρτησης FW_SR() προκειμένου να δημιουργηθούν τα νήματα εκτέλεσης μόνο μια φορά (δηλαδή όχι μέσα σε κάποιο loop). Έπειτα κάθε επιμέρους κλήση της FW_SR() ανατίθεται σε ένα νήμα με τη μορφή task. Ο συγχρονισμός στα απαιτούμενα τμήματα γίνεται με χρήση taskwait όπου είναι απαραίτητο.

Στην υλοποίηση με TBBs ορίζονται δύο groups από tasks g1, g2 μέσα στη FW_SR(). Οι g1, g2 αναλαμβάνουν την εκτέλεση των τμημάτων, τα οποία είναι κατάλληλα προς παραλληλοποίηση, ενώ όσα τμήματα είναι σειριακά εκτελούνται ως ένα κεντρικό task. Ο συγχρονισμός επιτυγχάνεται με κλήση wait() και δημιουργούνται επιμέρους task groups αναδρομικά, λόγω των κλήσεων FW_SR() μέσα στην ίδια. Η συγκεκριμένη υλοποίηση εκτελέστηκε επίσης με χρήση allocators,όμως δεν παρατηρήθηκε βελτίωση της επίδοσης, επομένως δεν αναγράφονται τα αντίστοιχα αποτελέσματα.

Ακολουθούν τα αποτελέσματα των πειραματικών μετρήσεων και τα αντίστοιχα διαγράμματα επιτάχυνσης των δύο υλοποιήσεων για B = 128:

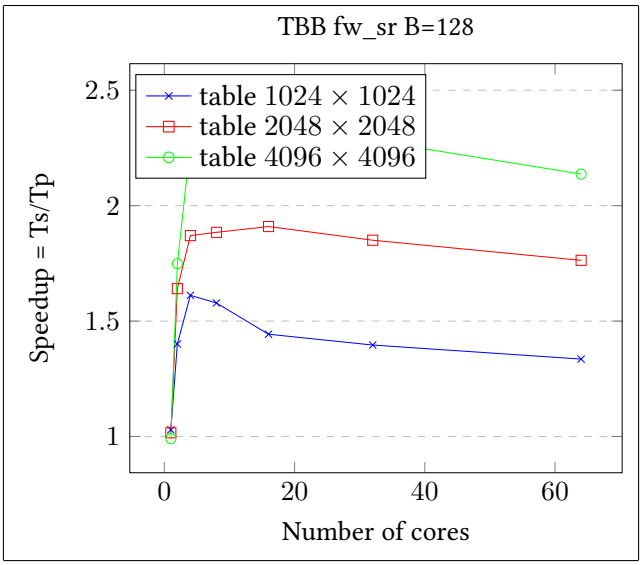
OpenMP:

Time(sec)	1024 x 1024	2048 x 2048	4096 x 4096
1	0.6236	4.5535	34.6119
2	0.3953	2.5881	18.5154
4	0.364	2.1713	13.4527
8	0.4436	2.4528	15.1442
16	0.4277	2.5856	14.4211
32	0.4803	2.7867	15.2096
64	0.468	2.7599	16.1228



TBBs:

Time(sec)	1024 x 1024	2048 x 2048	4096 x 4096
1	0.643872	4.73995	36.3043
2	0.472874	2.93541	20.568
4	0.411086	2.57485	15.8932
8	0.419732	2.55555	14.5787
16	0.459084	2.52184	14.9733
32	0.474499	2.60267	15.7595
64	0.496119	2.73119	16.8335



Παρατηρήσεις:

- Αρχικά παρατηρείται ότι στην υλοποίηση με χρήση openMP υπάρχει κάποια βελτίωση της επίδοσης σε σχέση με αυτής στην ενδιάμεση αναφορά (περίπου 5sec). Στο αρχικό πρόγραμμα της ενδιάμεσης αναφοράς η δημιουργία των νημάτων γινόταν σε κάθε επανάληψη μέσα στη FW_SR(), το οποίο ίσως προκαλούσε επιπλέον overhead συγχρονισμού λόγω της αναδρομικής κλήσης (δηλαδή μεγάλος αριθμός επιμέρους νημάτων-παιδιών). Στη συγκεκριμένη υλοποίηση ορίζεται η ομάδα των νημάτων μια μόνο φορά πριν την πρώτη κλήση της συνάρτησης και η εκτέλεση των επιμέρους tasks γίνεται μόνο από τα συγκεκριμένα νήματα.

- Όπως φαίνεται από τις μετρήσεις του χρόνου, η επίδοση της openMP έκδοσης είναι λίγο γρηγορότερη από την αντίστοιχη των TBBs. Οι δύο υλοποιήσεις είναι παρόμοιας λογικής, επομένως αυτό μπορεί να οφείλεται όπως αναφέρθηκε προηγουμένως, στην εσωτερική δομή της openMP, η οποία αποτελεί μοντέλο πιο κοντινό στο hardware επίπεδο από ότι τα TBBs.

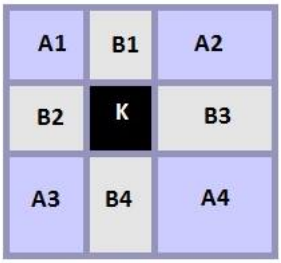
- Τέλος σημειώνεται ότι η επιτάχυνση είναι και στις δύο υλοποιήσεις όμοια, ενώ σε σύγκριση με την standard έκδοση του αλγορίθμου παρατηρείται ότι για μέγεθος πίνακα 4096 x 4096 η αναδρομική έκδοση είναι λίγο καλύτερη. Στις υπόλοιπες διαστάσεις ωστόσο, η αναδρομική έκδοση έχει πολύ χαμηλότερη επιτάχυνση,το οποίο ίσως προκύπτει από το overhead του συγχρονισμού που καθυστερεί τους επιμέρους υπολογισμούς του πίνακα.

Tiled έκδοση αλγορίθμου Floyd-Warshall

Παραλληλοποίηση:

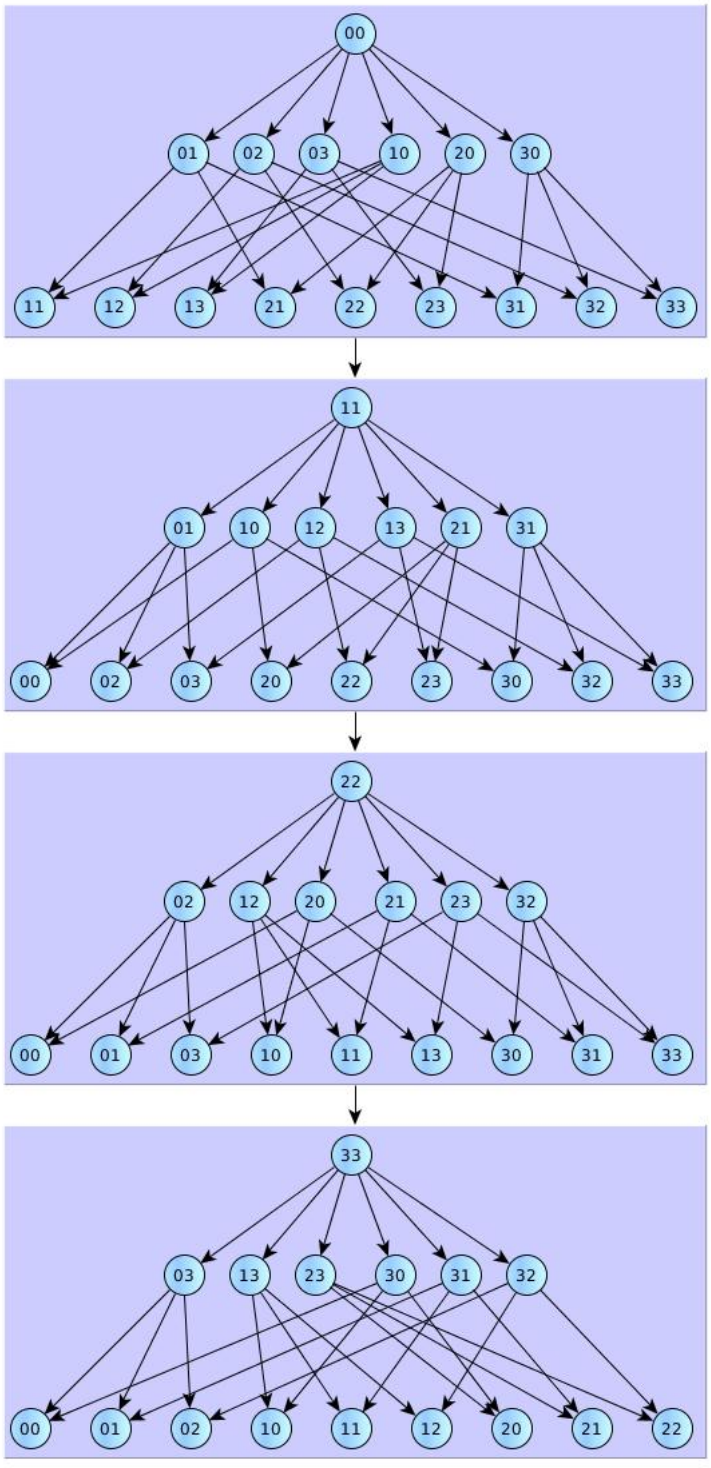
Η αναλυτική περιγραφή των εξαρτήσεων των τιμών του πίνακα A σε κάθε επανάληψη παρουσιάστηκε στην ενδιάμεση αναφορα. Για λόγους πληρότητας επαναλαμβάνεται στο συγκεκριμένο χωρίο:

Ο αρχικός πίνακας A χωρίζεται σε 9 επιμέρους πίνακες όπως φαίνεται στο σχήμα.



Αρχικά υπολογίζεται ο κεντρικός υποπίνακας K, ο οποίος εξαρτάται από τις δικές του τιμές. Έπειτα υπολογίζονται οι υποπίνακες B1, B2, B3 και B4, οι οποίοι για την ανανέωση των στοιχείων τους, χρησιμοποιούν τα δεδομένα του K, αλλά δεν έχουν καμία εξάρτηση μεταξύ τους (επομένως ο υπολογισμός τους μπορεί να παραλληλοποιηθεί). Τέλος μπορούν να υπολογιστούν τα στοιχεία των "άκρων" A1, A2, A3 και A4, οι οποίοι χρησιμοποιούν τις τιμές όλων των παραπάνω πινάκων. Σημειώνεται ότι τα επιμέρους for loops μπορούν να παραλληλοποιηθούν,καθώς για τον υπολογισμό 1 στοιχείου στο εσωτερικό του υποπίνακα δεν απαιτούνται νέες τιμές άλλων στοίχειων. Αυτή η διαδικασία ισχύει για δεδομένο k, δηλαδή ο εξωτερικός βρόχος εκτελείται σειριακά.

Ο αντίστοιχος γράφος εξαρτήσεων παρουσιάζεται στη στη συνέχεια για έναν πίνακα, του οποίου το μέγεθος N είναι 3 φορές μεγαλύτερο από το μέγεθος του μπλοκ B.



Η υλοποίηση της tiled έκδοσης του αλγορίθμου έγινε με χρήση openMP και TBBs.

Στην openMP έγιναν δύο υλοποιήσεις, οι οποίες παρουσιάζουν όμοια επίδοση (fw_tiled1.c fw_tiled2.c). Η διαφορά έγκειται στον τρόπο χρονοδρομολόγησης των υπολογισμών στα νήματα. Στην έκδοση fw_tiled1.c χρησιμοποιείται δυναμική χρονοδρομολόγηση με chunk size = 1. Με αυτό τον τρόπο όποιο νήμα τελειώνει τον υπολογισμό του υποπίνακα του μπορεί να χρονοδρομολογείται για την εκτέλεση κάποιου άλλου τμήματος του πίνακα. Στη δεύτερη περίπτωση γίνεται χρήση στατικής δρομολόγησης με αυξημένο chunk size = 512. Επομένως κάθε νήμα έχει συγκεκριμένη περιοχή

δεδομένων προς επεξεργασία. Στην τελευταία υλοποίηση έγιναν διάφορες δοκιμές μεγέθους chunk, αλλά η συγκεκριμένη σημείωνε την καλύτερη επίδοση. Επισημαίνεται ότι οι δύο εκδόσεις έχουν παρόμοια υλοποίηση και επιδόσεις, αλλά γίνεται ξεχωριστή αναφορά σε αυτές για να τονιστεί ότι η χρονοδρομολόγηση με διαφορετικό τρόπο και chunk μπορεί να επιφέρει αρκετές αλλαγές στην επίδοση.

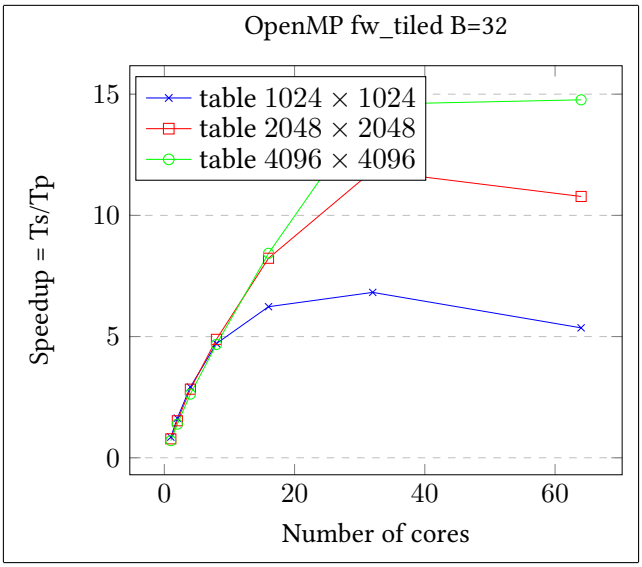
Για την εκτέλεση των δύο προγραμμάτων τέθηκε η μεταβλητή GOMP_CPU_AFFINITY για την εκμετάλλευση της τοπικότητας των δεδομένων.

Στην υλοποίηση του προγράμματος των TBBs fw_tiled_for_auto.cpp χρησιμοποιείται parallel_invoke()προκειμένου να γίνει παράλληλη εκτέλεση των επιμέρους tasks που ορίζονται. Στο πρόγραμμα φαίνεται ότι αυτή η παραλληλοποίηση είναι δυνατή σε δύο τμήματα, στα οποία εμπεριέχονται 4 loops με κλήσεις της συνάρτησης FW(). Καθένα από τα παραπάνω loops μπορεί να τμηματοποιηθεί με τη χρήση parallel_for και με αυτό το τρόπο διαμερίζεται το αρχικό range του και κάθε worker thread αναλαμβάνει τον υπολογισμό ενός επιμέρους task του. Επίσης γίνεται χρήση ενός auto_partitioner για καλύτερο load balancing και χρήση allocators για την αντιμετώπιση false sharing και scalability bottlenecks, ενώ δίνεται στο χρήστη δυνατότητα να ρυθμίσει το grainsize, μέσω των μεταβλητών Gx, Gy. Μετά από εκτέλεση του προγράμματος για διάφορες τιμές εισόδου διαπιστώθηκε ότι καλύτερη επίδοση παρατηρείται για Gx = Gy =1. Τέλος επισημαίνεται ότι έγινε και αντίστοιχη δοκιμή του affinity_partitioner και οι επιδόσεις ήταν παρόμοιες.

Ακολουθούν τα αποτελέσματα των πειραματικών μετρήσεων και τα αντίστοιχα διαγράμματα επιτάχυνσης για B=32 στην υλοποίηση με OpenMP και B=64 στην υλοποίηση με TBB:

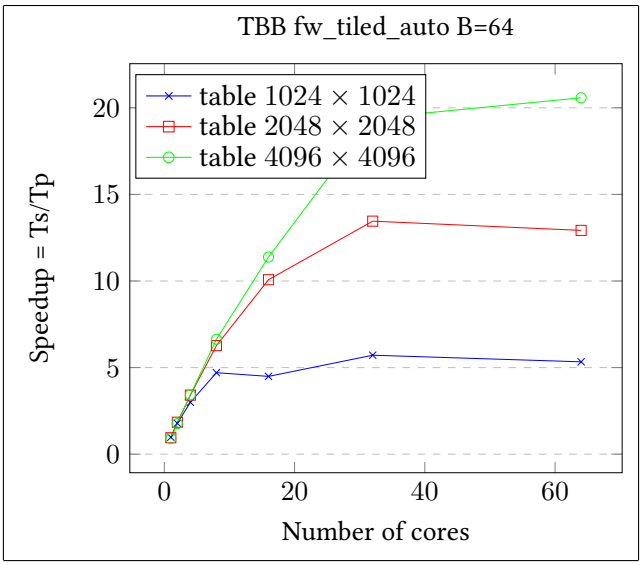
OpenMP: Παρουσιάζονται οι μετρήσεις μόνο για την έκδοση fw_tiled1.c, καθώς είναι παρόμοιες με αυτές της δεύτερης υλοποίησης:

Time(sec)	1024 x 1024	2048 x 2048	4096 x 4096
1	0.7766	6.1613	50.9861
2	0.4051	3.1543	25.9248
4	0.2269	1.701	13.6953
8	0.1406	0.9855	7.6942
16	0.1063	0.585	4.2641
32	0.0971	0.4066	2.4665
64	0.1236	0.4467	2.4357



TBBs:

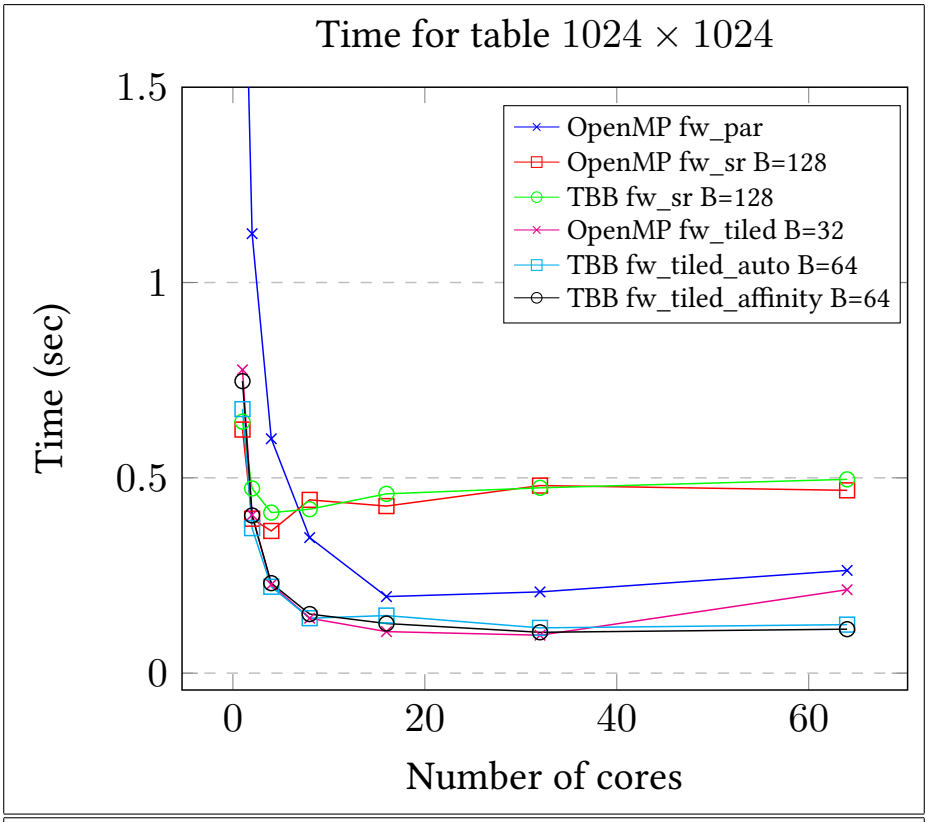
Time(sec)	1024 x 1024	2048 x 2048	4096 x 4096
1	0.675934	5.08564	40.116
2	0.371206	2.61158	20.3734
4	0.221244	1.41227	10.5016
8	0.140783	0.768177	5.42144
16	0.147499	0.478006	3.16048
32	0.115936	0.357869	1.854
64	0.124218	0.372655	1.74781

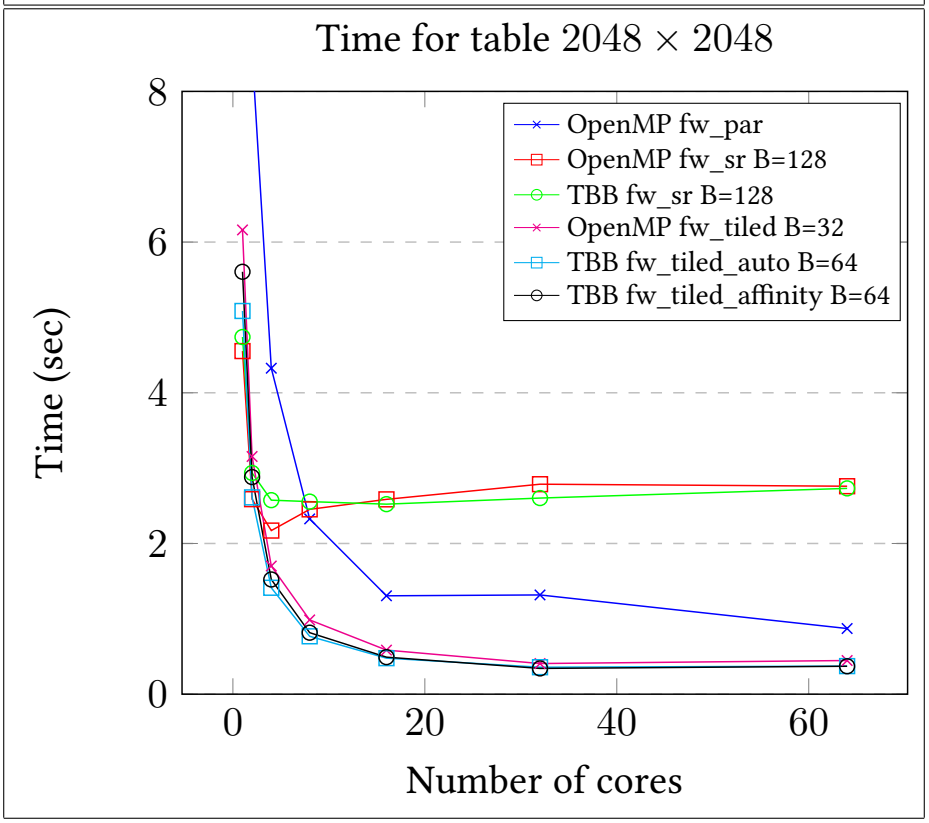


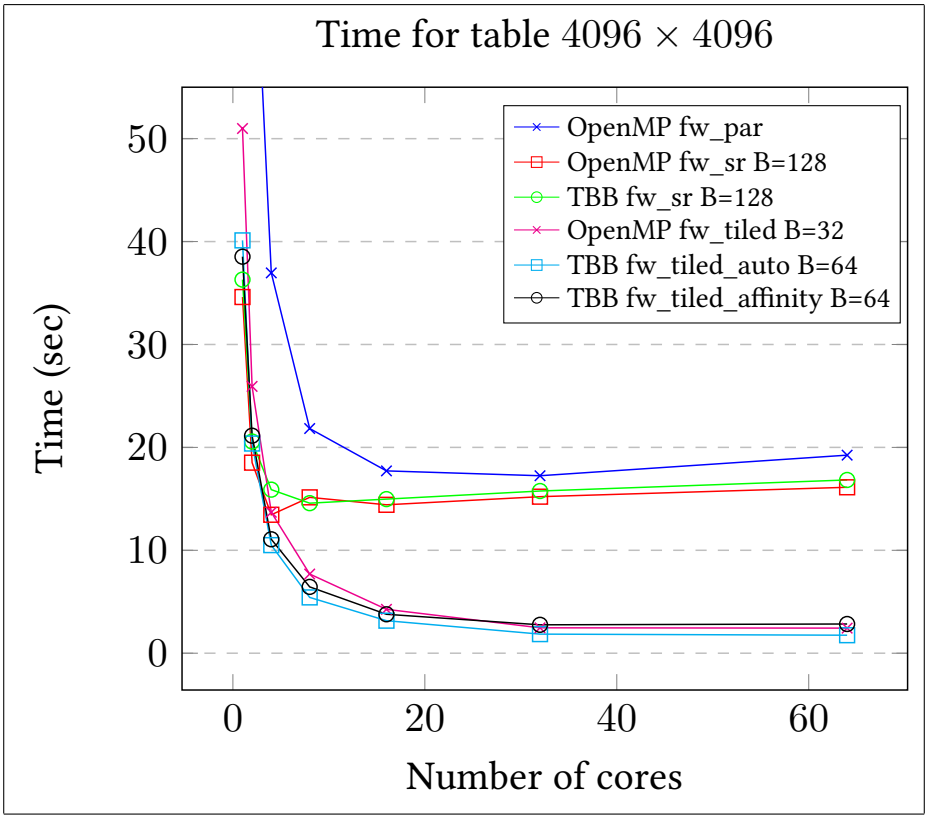
Παρατηρήσεις:

- Σε σύγκριση με την υλοποίηση των προγραμμάτων της ενδιάμεσης αναφοράς υπάρχει κάποια βελτίωση χρόνου περίπου 2.5sec, η οποία οφείλεται κυρίως στην εκμετάλλευση του affinity.
- Ανάμεσα στις υλοποιήσεις με χρήση openMP και TBBs, η δεύτερη είναι γρηγορότερη για μεγαλύτερα μεγέθη πίνακα. Αυτή η διαφορά μπορεί να οφείλεται στην εκμετάλλευση των allocators και partitioners, τα οποία υλοποιούνται εσωτερικά στη δομή των TBBs. Αντίθετα η χρήση της openMP έχει πιο περιορισμένες δυνατότητες (θα μπορούσε να αποτελεί υποσύνολο των χειρισμών των TBBs).
- Τέλος αξίζει να παρατηρηθεί ότι η tiled έκδοση του αλγορίθμου παρουσιάζει γενικά την καλύτερη επιτάχυνση και στις δύο υλοποιήσεις (openMP και TBBs) συγκριτικά με τις προηγούμενες εκδόσεις. Με άλλα λόγια ο ανασχηματισμός/τροποποίηση του κώδικα στη συγκεκριμένη περίπτωση οδηγεί σε καλύτερη παραλληλοποίηση και συγχρονισμό των νημάτων (το overhead των πολλαπλών threads δεν επικαλύπτει την επίδοση της παραλληλοποίησης).

Ακολουθούν τα συγκεντρωτικά διαγράμματα με τους χρόνους κάθε υλοποίησης του αλγορίθμου:


Time for table 1024 × 1024
Time (sec)
Number of cores
0
0.5
1
1.5
0
20
40
60
OpenMP fw_par
OpenMP fw_sr B=128
TBB fw_sr B=128
OpenMP fw_tiled B=32
TBB fw_tiled_auto B=64
TBB fw_tiled_affinity B=64


Time for table 2048 × 2048
Time (sec)
Number of cores
0
2
4
6
8
0
20
40
60
OpenMP fw_par
OpenMP fw_sr B=128
TBB fw_sr B=128
OpenMP fw_tiled B=32
TBB fw_tiled_auto B=64
TBB fw_tiled_affinity B=64



Γενικές Παρατηρήσεις:

- Όπως επισημαίνεται στην ενδιάμεση αναφορά, σε όλες σχεδόν τις παραπάνω υλοποιήσεις παρατηρείται βελτίωση του χρόνου εκτέλεσης με την αύξηση των νημάτων. Η αύξηση των νημάτων οδηγεί σε ταυτόχρονη εκτέλεση περισσότερων υπολογισμών, άρα επιταχύνεται η εκτέλεση του προγράμματος.

- Στην έκδοση της αναδρομικής μορφής του Floyd-Warshall, τόσο στην υλοποίηση με χρήση openMP όσο και με TBBs παρατηρείται ότι για πλήθος νημάτων μεγαλύτερο από 8, η επίδοση δυσχεραίνει. Αυτό μπορεί να οφείλεται στον αριθμό των νημάτων, καθώς με την αύξηση του πλήθους τους απαιτείται η χρήση επεξεργαστικών μονάδων από παραπάνω υποσυστήματα (Numa Nodes). Αυτό συνεπάγεται την ανάγκη για επικοινωνία μεταξύ των προαναφερθέντων κόμβων, το οποίο οδηγεί στη διαδοχική ανταλλαγή μηνυμάτων, με σκοπό τη μεταφορά των νέων δεδομένων, στα οποία έγινε κάποια εγγραφή. Η συγκεκριμένη μορφή επικοινωνίας αυξάνει την αναμονή των επεξεργαστών και επομένως την καθυστέρηση της εκτέλεσης του προγράμματος. Επίσης με την αύξηση των νημάτων, δημιουργείται μεγαλύτερη ανάγκη συγχρονισμού και επομένως αυξάνεται το συνολικό overhead.

- Το μέγεθος της εισόδου επηρεάζει άμεσα την επίδοση εκτέλεσης του κάθε αλγορίθμου. Σχετικά παραδείγματα παρατέθηκαν στην ενδιάμεση αναφορά, επομένως στην παρούσα γίνεται παράθεση μόνο των καλύτερων επιδόσεων παραλληλοποίησης για κατάλληλα μεγέθη εισόδου. Ενδεικτικά όμως αναφέρουμε ότι στην υλοποίηση της tiled έκδοσης η αλλαγή του Grainsize έχει άμεση επίπτωση στο χρόνο εκτέλεσης. Επίσης οι καλύτερες επιδόσεις της αναδρομικής και tiled έκδοσης του αλγορίθμου παρατηρούνται για διαφορετικά B.

- Εκτός από το μέγεθος της εισόδου, η επίδοση κάθε υλοποίησης είναι άρρηκτα συνδεδεμένη με την αρχιτεκτονική του sandman. Χαρακτηριστικά η standard έκδοση του αλγορίθμου φαίνεται

να παραλληλοποιείται καλύτερα από την αναδρομική έκδοση. Ωστόσο ο χρόνος εκτέλεσης της δεν είναι καλύτερος, γεγονός το οποίο μάλλον οφείλεται στο μοντέλο αρχιτεκτονικής της μνήμης.

- Σε όλες σχεδόν τις περιπτώσεις (με εξαίρεση την standard έκδοση του αλγορίθμου) οι επιδόσεις με χρήση TBBs είναι καλύτερες από αυτές της openMP. Επίσης αναφέρεται ότι καλύτερος χρόνος για όλες τις διαστάσεις του πίνακα επιτυγχάνεται από την tiled (TBB) υλοποίηση. Συγκεκριμένα: για N= 4096 έχουμε 1.74781 sec με 64 νήματα, για N=2048 έχουμε 0.357869 sec με 32 νήματα και για N=1024 έχουμε 0.0971 sec με επίσης 32 νήματα.