Μαλακούδης Σταύρος 9368 samalako@ece.auth.gr

Ανάλυση σειριακών και παράλληλων αλγορίθμων για την εύρεση των ανά-κόμβο τριγώνων σε μεγάλους γράφους .

V3

Η λογική για την εύρεση των τριγώνων είναι η εξής:

- Πηγαίνω σε κάθε μη μηδενικό στοιχείο του άνω τριγωνικού πίνακα, έστω το A[i][j].
- Η ύπαρξη του A[i][j] σημαίνει ότι υπάρχει ακμή μεταξύ του i- και j- κόμβου, επομένως πηγαίνω σε κάθε μη μηδενικό στοιχείο στην j- γραμμή, έστω το A[j][k].
- Ομοίως , η ύπαρξη του A[j][k] σημαίνει ότι υπάρχει ακμή μεταξύ του j- και k- κόμβου, άρα πάω στην i- σειρά και ψάχνω να βρω αν υπάρχει το στοιχείο k.
- Ο αλγόριθμος είναι στημένος έτσι ώστε να μην γίνονται περιττά iterations ή έλεγχοι. Δηλαδή , μόλις βρεθεί ένα στοιχείο, προχωράει στην επόμενη σειρά, αφού γνωρίζουμε ότι δεν υπάρχει περίπτωση να ξαναβρεθεί το ίδιο στοιχείο στην ίδια σειρά. Επίσης αφού τα indexes είναι αποθηκευμένα με αύξουσα σειρά, μόλις φτάσει ο αλγόριθμος σε index μεγαλύτερο από αυτό που ψάχνει, τότε ακυρώνει την αναζήτηση. Αυτές οι δύο “μικροβελτιώσεις” συνέβαλαν στη μείωση του σειριακού χρόνο περίπου κατά 40-50% κατά μέσο όρο.

V4

Με σκοπό την υλοποίηση του $C = A \otimes (A \cdot A) \cdot \frac{e}{2}$

- Πηγαίνω (ομοίως με την V3) σε κάθε μη μηδενικό στοιχείο σε κάθε γραμμή, έστω το A[i][j], υλοποιώντας με αυτόν τον τρόπο το Hadamard product.
- Για να βρεθεί το C[i][j] στο A[i][j], αρκεί να κάνω τον πολλαπλασιασμό της i- γραμμής επί την j- στήλη. Επειδή ο πίνακας είναι συμμετρικός, αρκεί να κάνω τον πολλαπλασιασμό i- γραμμή επί j- γραμμή.
- Το αποτέλεσμα του πολλαπλασιασμού είναι το πλήθος των ζευγών των στοιχείων της i- και j- γραμμής που βρίσκονται στην ίδια στήλη, το οποίο εξισώνεται με το C[i][j].
- Ομοίως με την V3 , η εύρεση των κοινών στοιχείων γίνεται με σειριακή αναζήτηση (έγινε προσπάθεια για την υλοποίηση δυαδικής αναζήτησης, η οποία ήταν τουλάχιστον μια τάξη μεγέθους πιο γρήγορη- αλλά τα τρίγωνα έβγαιναν λάθος για μεγάλους πίνακες).
- Για ευκολία, ο πίνακας C είναι μονοδιάστατος και όχι δισδιάστατος (ίσως εδώ οφείλεται η μεγάλη διαφορά στην απόδοση σε σχέση με την V3).
- Αφού υπολογιστεί το $A \odot A \cdot A$, προστίθενται τα στοιχεία της κάθε γραμμής μεταξύ τους και το αποτέλεσμα διαιρείται διά 2 ώστε να προκύψει το ζητούμενο.

Σημείωση: Υπάρχει αισθητή διαφορά στους χρόνους εκτέλεσης μεταξύ V3 και V4 . Αυτό , ίσως οφείλεται στο γεγονός ότι η πρώτη δουλεύει με τον άνω τριγωνικό πίνακα μόνο (άρα μισά στοιχεία) ενώ η V4 με ολόκληρο. Επίσης, στην V4 εμφανίζονται και χρησιμοποιούνται επιπλέον μεταβλητές, των οποίων η προσπέλαση ενδεχομένως προσθέτει περισσότερη πολυπλοκότητα και επομένως υπάρχει μεγαλύτερη καθυστέρηση.

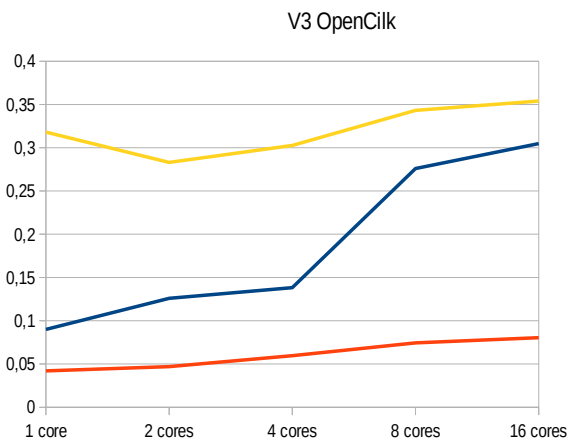
Σύγκριση και σχολιασμός αποτελεσμάτων: Η ανάλυση των χρόνων σε 1-2 και 4 πυρήνες έγινε σε τοπικό τερματικό οπότε η συστοιχία χρησιμοποιήθηκε για 8 και 16 πυρήνες (πολλαπλοί έλεγχοι απέδειξαν ότι οι χρόνοι εκτέλεσης με 4 πυρήνες το πολύ, είναι πανομοιότυποι στο τοπικό τερματικό και τη συστοιχία . Οι πίνακες που δόθηκαν ήταν συνολικά 5. Ορισμένοι , παρόλο που είχαν κοινά χαρακτηριστικά (ίδιο αριθμό κόμβων , ακμών και τριγώνων) παρουσίασαν τελείως διαφορετικές ιδιότητες.

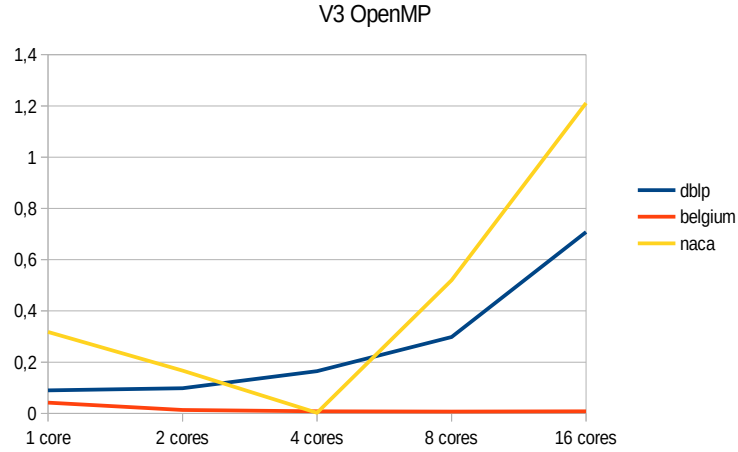
Μαλακούδης Σταύρος 9368 samalako@ece.auth.gr

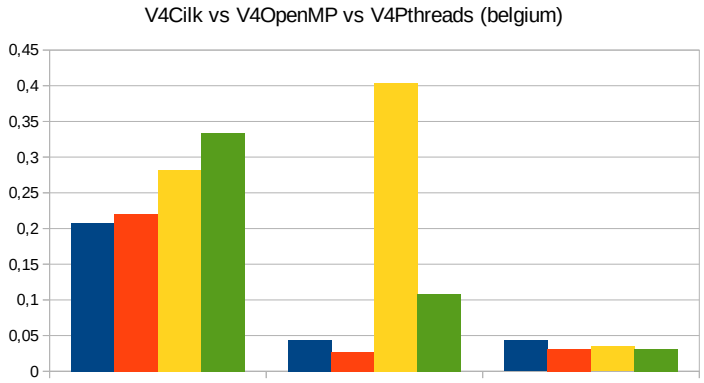
Χρησιμοποιήθηκαν οι εξής πίνακες. Κατά σειρά: Κόμβοι, ακμές, τρίγωνα, πηλίκο ακμές/κόμβοι

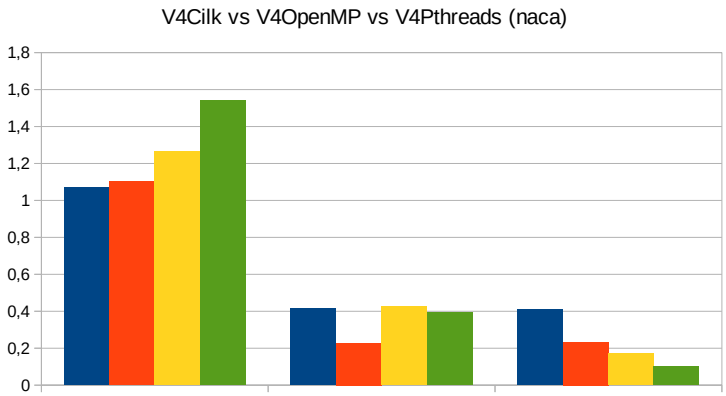
dblp	326186	807700	1676652	2.47619456383781
youtube	1134890	2987624	3056386	2.63252297579501
belgium	1441295	1549970	2420	1.0754009415144
naca	1039183	3114818	2075635	2.99737197394492
mycielskian	6143	613871	0	99.9301644147811

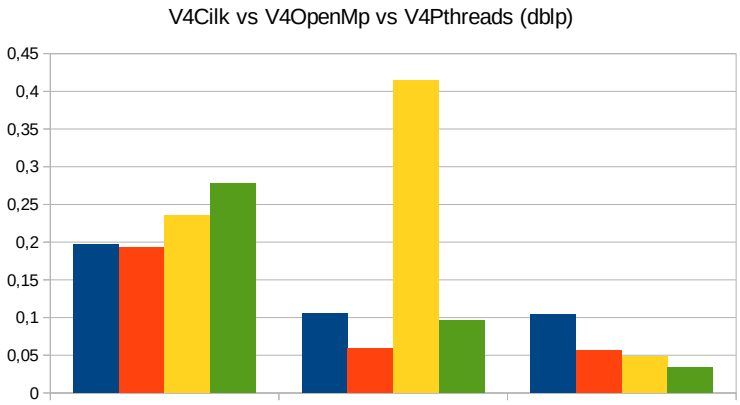
Οι πίνακες dblp, belgium και naca, τόσο στην V3 όσο και στην V4 χρειάστηκαν περισσότερο χρόνο και με OpenCilk και με OpenMP. Εξαίρεση αποτελεί η εκτέλεση V4 με Pthreads, που φαίνεται να βελτιώνει την απόδοση.







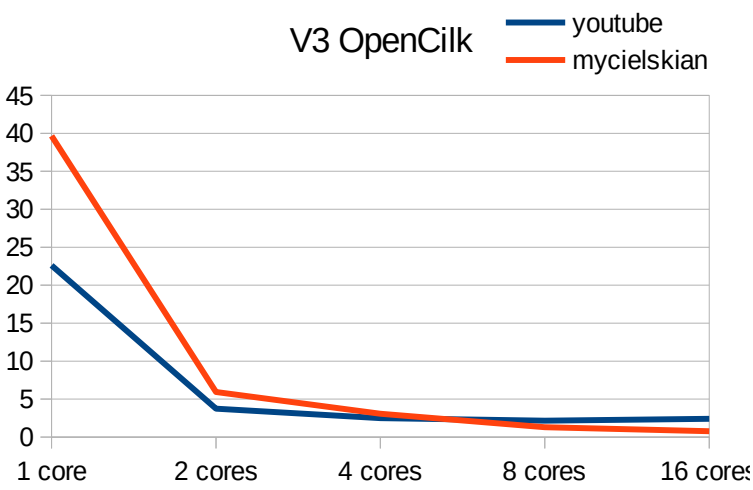


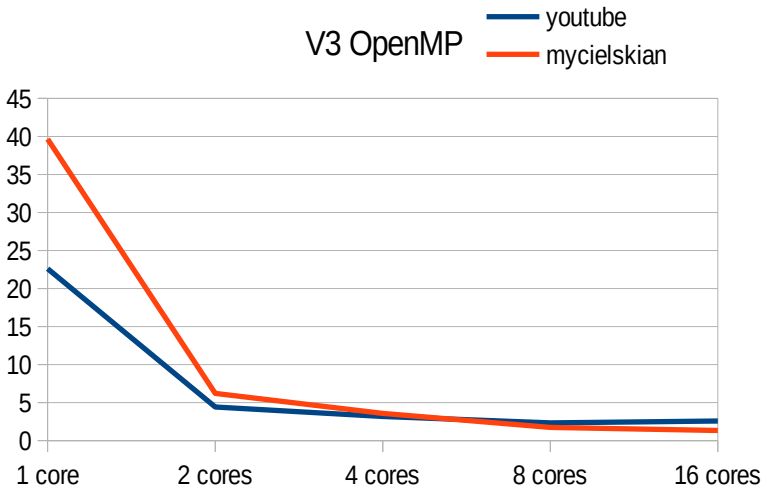


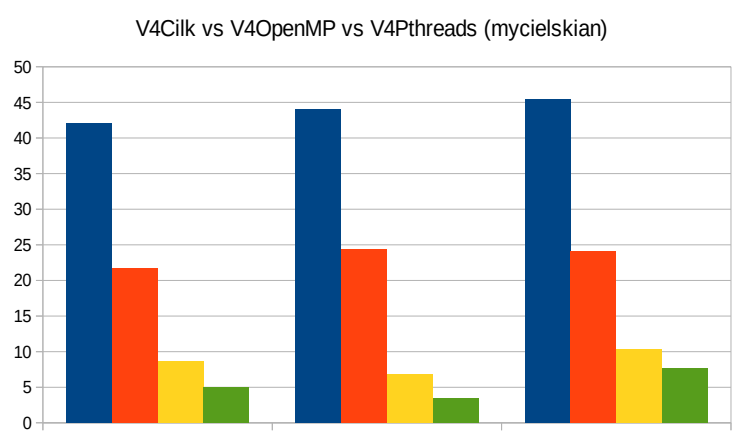
Υπόμνημα: Κάθε χρώμα συμβολίζει εκτέλεση με 2-4-8-16 πυρήνες αντίστοιχα. Η V4OpenMP είχε data races στην συστοιχία, με κάθε αριθμό πυρήνων, ενώ στο δικό μου τερματικό τρέχει κανονικά, για κάθε αριθμό πυρήνων . Επομένως, τα αποτελέσματα από την V4OpenMP καλό είναι να αγνοηθούν. Φαίνεται ξεκάθαρα ότι η υλοποίηση με Pthreads **έχει αντίθετη επίδραση** σε σχέση με οποιαδήποτε άλλη μέθοδο παραλληλοποίησης. Αυτό ίσως οφείλεται στην απλή φύση των Pthreads και τον λίγο χρόνο που χρειάζεται το σύστημα για την εκκίνηση, τον τερματισμό και την υλοποίηση τους (όπως και αναφέρεται στη δοσμένη βιβλιογραφία). Δηλαδή το κόστος παραλληλοποίησης με cilk και open MP είναι μεγαλύτερο από την βελτιστοποίηση.

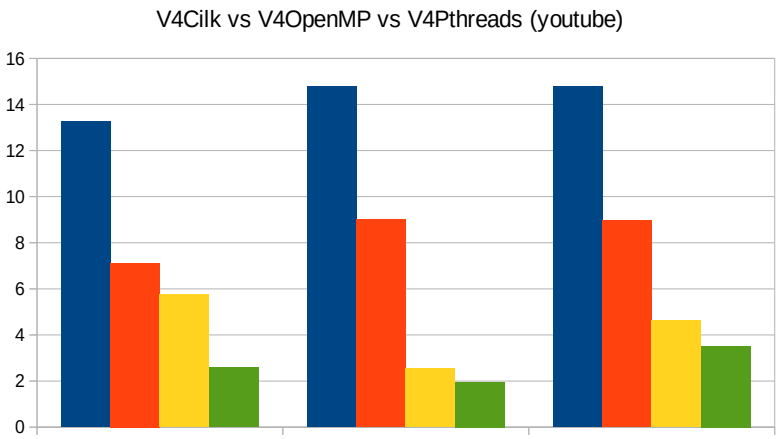
Μαλακούδης Σταύρος 9368 samalako@ece.auth.gr

Οι πίνακες youtube και mycielskian έχουν την αντίθετη συμπεριφορά στην παραλληλοποίηση, με κάθε μέθοδο, σε σχέση με τους προηγούμενους. Μάλιστα φαίνεται ξεκάθαρα ότι η βελτίωση στην απόδοση κατά τη μετάβαση από 1 στους 2 πυρήνες είναι υπερ-γραμμική, ενώ έπειτα υπάρχει κορεσμός.









Σύγκριση μεταξύ OpenCilk, OpenMP & Pthreads: Φαίνεται εμφανώς ότι η Cilk έχει καλύτερη επίδοση για 2-4 πυρήνες, ενώ η OpenMP για περισσότερους από 4.

Σχολιασμός απόδοσης σε σχέση με το μέγεθος: Το δείγμα των πινάκων που χρησιμοποιήθηκαν ήταν μικρό και δεν μπορούν να προκύψουν καθαρά συμπεράσματα. Για παράδειγμα, ο youtube και ο naca έχουν ίδιας τάξης μεγέθους κόμβους, ακμές αλλά και τρίγωνα. Όμως , συμπεριφέρονται τελείως διαφορετικά σε κάθε αλγόριθμο και κάθε υλοποίηση. Αυτό ίσως φταίει στη μορφή του πίνακα, δηλαδή στο που βρίσκονται “μαζεμένα” τα στοιχεία. Δηλαδή , ο youtube ίσως έχει περιοχές που είναι πολύ πυκνές και περιοχές που είναι πολύ αραιές. Έτσι , μπορεί συνολικά να θεωρείται αραιός , αλλά τοπικά πυκνός. Ομοίως ο mycielskian που είχε το λιγότερο πλήθος κόμβων , ακμών και τριγώνων χρειάστηκε την περισσότερη ώρα. Παρατηρώ ότι είναι πιο πυκνός από τους υπόλοιπους, δηλαδή έχει πολλές περισσότερες ακμές σε σχέση με τους κόμβους του, οπότε ίσως αυτό να είναι μια ένδειξη. *Φαίνεται ότι όσο πιο πυκνός είναι ο πίνακας, τόσο περισσότερη ώρα χρειάζεται, ανεξάρτητα με το μέγεθος που έχει*. Αν υπήρχαν περισσότεροι δοκιμαστικοί πίνακες, τότε το συμπέρασμα θα ήταν πιο προφανές.

Όλα τα αρχεία που χρησιμοποιήθηκαν, οι πηγαίοι κώδικες, οι πίνακες και τα αποτελέσματα βρίσκονται συγκεντρωμένα στο: Drive