Δυαδικός Πολλαπλασιασμός Αραιών Πινάκων (Sparse BMM)

Τσουμπλέκας Γεώργιος
AEM: 9359
gktsoump@ece.auth.gr

Αραϊλόπουλος Βασίλειος
AEM: 9475
varailop@ece.auth.gr

Σεπτέμβριος 2021

Ο κώδικας της εργασίας βρίσκεται εδώ.

Εισαγωγή

Ζητούμενο της εργασίας αυτής είναι η υλοποίηση του Δυαδικού Πολλαπλασιασμού Αραιών Πινάκων (Boolean Sparse Matrix Multiplication - BMM) χρησιμοποιώντας δύο μεθόδους παραλληλοποίησης.

Μια εφαρμογή του BMM, που μας είναι γνώριμη από την πρώτη εργασία του μαθήματος, είναι στον υπολογισμό των τριγώνων (triangle counting) ενός γράφου. Πιο συγκεκριμένα, είδαμε ότι χρειαζόταν ο υπολογισμός της έκφρασης $A \odot (AA)$, όπου A ο πίνακας γειτνίασης του γραφήματος. Γενικά, ο πίνακας A σε πολλές εφαρμογές είναι αραιός με αποτέλεσμα το BMM να βρίσκει εδώ άμεση εφαρμογή.

Μια ακόμα σημαντική εφαρμογή είναι στο πρόβλημα transitive closure σε γράφους. Για την επίλυση αυτού του προβλήματος είναι απαραίτητος ο υπολογισμός της έκφρασης $A^+ = A + A^2 + ... + A^n$, το οποίο περιλαμβάνει τον πολλαπλασιασμό του πίνακα γειτνίασης (adjacency matrix) A ενός γραφήματος με τον εαυτό του. Και πάλι αν ο A είναι αραιός το BMM μας είναι ένα χρήσιμο εργαλείο.

Αλλά και πέρα από την θεωρία δικτύων, η εφαρμογή του BMM σε προβλήματα όπως το Finite Elements Method (FEM) και τομείς όπως η δυναμική ρευστών και η μηχανική μάθηση μεταξύ άλλων το καθιστούν ως ένα πολύ σημαντικό πρόβλημα των μαθηματικών και της επιστήμης των υπολογιστών.

1 Σειριακή υλοποίηση

Για τον πολλαπλασιασμό δύο πινάκων A και B αξιοποιούνται οι δύο συμπιεσμένες μορφές (Compressed Sparse) για την περιγραφή των αραιών πινάκων (CSR και CSC). Ο πίνακας A αποθηκεύεται σε CSR μορφή, για ευκολότερη προσπέλαση των σειρών του, και ο πίνακας B σε CSC, για ευκολότερη προσπέλαση των στηλών του. Έχοντας ως δεδομένο αυτές τις μορφές, ελέγχονται όλες οι σειρές του A με όλες τις στήλες του B για να βρεθούν οι τιμές του πίνακα γινομένου C. Για να βρεθεί αν μία τιμή στον πίνακα C είναι 0 ή 1, ελέγχεται αν σε μια σειρά του A και σε μία στήλη του B υπάρχουν κοινοί δείκτες στους πίνακες και μόλις βρεθεί κοινός δείκτης αποθηκεύεται η νέα μη μηδενική τιμή στον πίνακα C και ο αλγόριθμος συνεχίζει σε επόμενη τιμή του πίνακα. Ο πίνακας C αποθηκεύεται απευθείας σε CSR μορφή.

Μια βελτίωση της σειριακής μεθόδου γίνεται με τον χωρισμό των πινάκων σε blocks. Κάθε block πρόκειται ουσιαστικά για έναν αραιό τετραγωνικό υποπίνακα του αρχικού, αποθηκευμένο σε CSR ή CSC μορφή (ανάλογα και με τον αρχικό). Ένα σημαντικό σχεδιαστικό στοιχείο των blocks είναι το μέγεθός τους σε σχέση με τον αρχικό. Μικρό b συνεπάγεται την ύπαρξη πολλών blocks και άρα μεγαλύτερη κατανάλωση μνήμης ενώ μεγάλο b δεν προσφέρει ιδιαίτερη ευχρηστία όσον αφορά την

μετέπειτα παραλληλοποίηση. Όπως θα φανεί και από τα πειραματικά αποτελέσματα στην συνέχεια ένας καλός συμβιβασμός είναι το $b = 0.1n$ όπου n η διάσταση του αρχικού πίνακα. Με την σειρά τους, τα blocks είναι και αυτά αποθηκευμένα μεταξύ τους σε CSR ή CSC μορφή (υπερπίνακας blocks). Σε αντίστοιχη με πριν λογική, ελέγχονται οι σειρές και οι στήλες των υπερπινάκων αυτών και όταν πρέπει να πολλαπλασιαστούν δύο blocks των πινάκων χρησιμοποιείται η παραπάνω συνάρτηση. Για την εύρεση ενός block του πίνακα C, χρησιμοποιείται μια συνάρτηση που εκτελεί την πρόσθεση των λογικών πινάκων που προκύπτουν από τους πολλαπλασιασμούς των blocks.

Τέλος, υλοποιήθηκαν και δυο ακόμα συναρτήσεις που εκτελούν τον filtered πολλαπλασιασμό $F \odot AB$ όπου ο F είναι επίσης αραιός πίνακας. Η διαφορά με τις παραπάνω μεθόδους είναι ότι ελέγχονται μόνο οι θέσεις του πίνακα F που περιέχουν μη μηδενικά στοιχεία. Οι υλοποιήσεις βρίσκονται στο αρχείο bmm_seq.h.

2 Παράλληλες υλοποιήσεις

2.1 Παραλληλισμός με OpenMP

Για την επίτευξη του παραλληλισμού με OpenMP, ο πίνακας A χωρίζεται σε τόσες ισότιμες οριζόντιες λωρίδες (chunks) όσα είναι και τα νήματα που καλούνται. Κάθε νήμα πολλαπλασιάζει την λωρίδα που του αντιστοιχεί με τον πίνακα B, παράγει την αντίστοιχη λωρίδα στον πίνακα C και στο τέλος της συνάρτησης οι λωρίδες του C ενώνονται σε έναν ενιαίο πίνακα. Με παρόμοιο τρόπο εκτελείται και η συνάρτηση που διαχειρίζεται τους χωρισμένους σε blocks πίνακες, στην οποία κάθε νήμα πολλαπλασιάζει μια λωρίδα από blocks. Όπως και στην σειριακή μέθοδο, υπάρχουν οι filtered εκδοχές που βασίζονται στην ίδια λογική και χωρίζουν τον πίνακα F σε οριζόντιες λωρίδες και εκτελούν τον πολλαπλασιασμό. Οι υλοποιήσεις βρίσκονται στο αρχείο blocked_bmm_parallel.h.

2.2 Παραλληλισμός με MPI

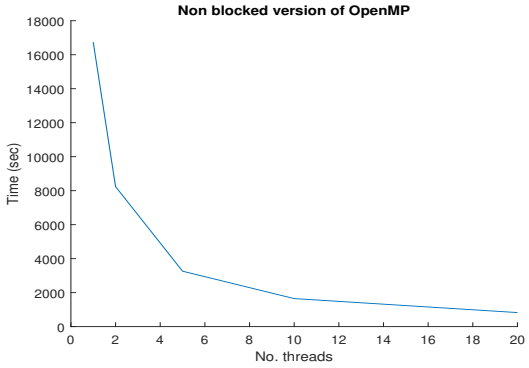
Όπως και στον παραλληλισμό με OpenMP, αντίστοιχα σε αυτήν την μέθοδο είναι ο πίνακας B που χωρίζεται σε κατακόρυφες λωρίδες. Η κεντρική διαδικασία (process) αποστέλλει σε κάθε process όλο τον πίνακα A και χρησιμοποιεί την συνάρτηση MPI_Scatterv για να στείλει ισότιμες λωρίδες του πίνακα B, οι οποίες στην συνέχεια πολλαπλασιάζονται με τον πίνακα A χρησιμοποιώντας τις σειριακές συναρτήσεις. Στη συνέχεια, οι λωρίδες του πίνακα C που παράγονται, συλλέγονται από την κεντρική διαδικασία και ενώνονται σε έναν ενιαίο πίνακα. Αντίστοιχη είναι και η λειτουργία στους blocked πίνακες. Τέλος, υπάρχουν και οι filtered εκδόσεις που χωρίζουν και τον πίνακα B και F σε κατακόρυφες λωρίδες και στην συνέχεια τους διασκορπίζουν στα υπάρχοντα processes ώστε καθένα από αυτά να παράξει το αντίστοιχο κομμάτι του πίνακα C. Οι υλοποιήσεις βρίσκονται στο αρχείο blocked_bmm_parallel_MPI.h.

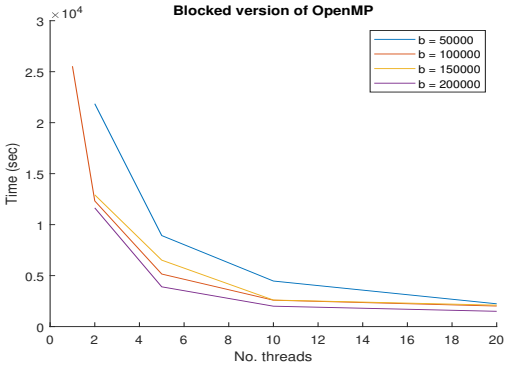
2.3 Συνδυασμός των δύο μεθόδων

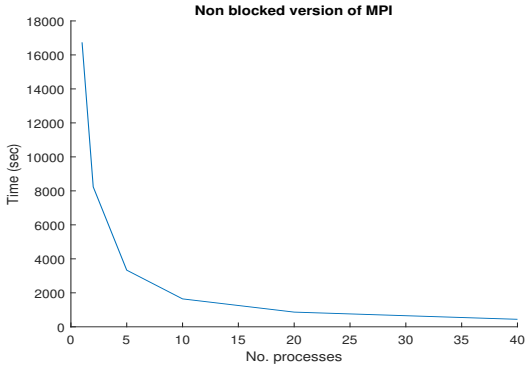
Ουσιαστικά, ακολουθείται ο ίδιος διαμοιρασμός των δεδομένων που είχαμε και στην MPI υλοποίηση και στην συνέχεια σε κάθε process έχουμε multi-threaded παραλληλισμό με την μέθοδο που παρουσιάστηκε με την OpenMP υλοποίηση. Τα προκύπτοντα κομμάτια του τελικού πίνακα πρώτα ενώνονται από τα threads σε κάθε process και στην συνέχεια τα κομμάτια του κάθε process ενώνονται και αυτά για να προκύψει ο τελικός πίνακας. Οι υλοποιήσεις βρίσκονται στο αρχείο combined_blocked_bmm_parallel.h.

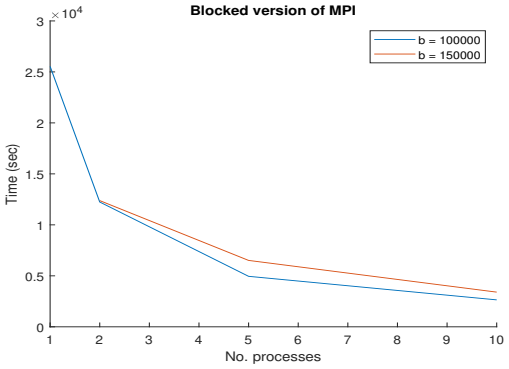
3 Πειραματικά Αποτελέσματα

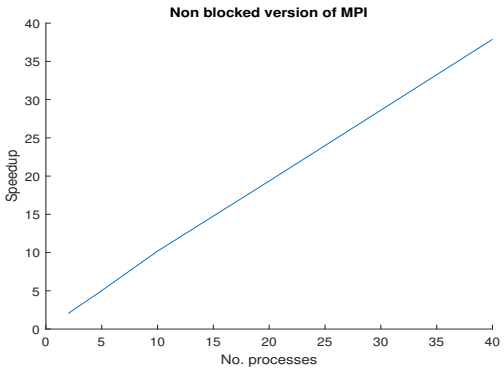
Τα πειραματικά αποτελέσματα που παρουσιάζονται στην ενότητα αυτή προέκυψαν κατόπιν εκτέλεσης των προγραμμάτων στην συστοιχία HPC του ΑΠΘ.

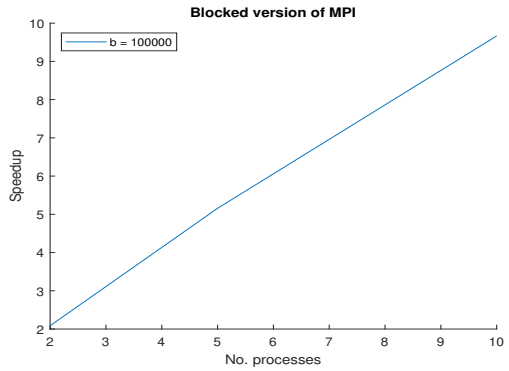

Non blocked version of OpenMP
Time (sec)
No. threads

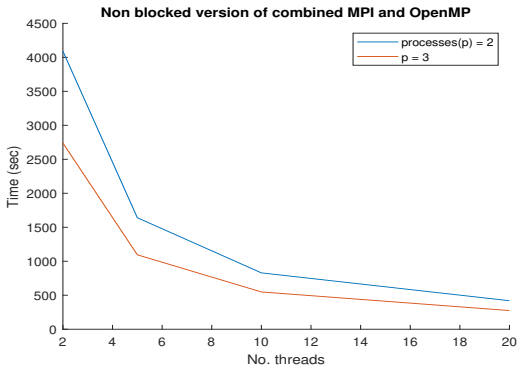

Blocked version of OpenMP
×10^4
b = 50000
b = 100000
b = 150000
b = 200000
Time (sec)
No. threads

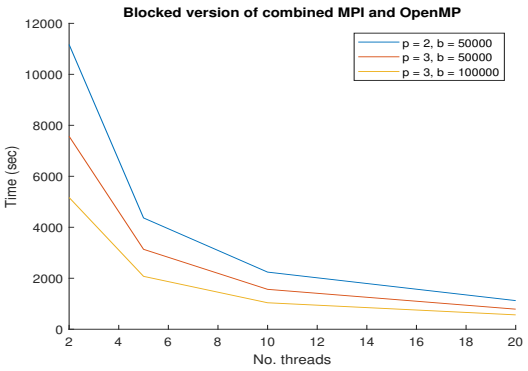

Non blocked version of MPI
Time (sec)
No. processes

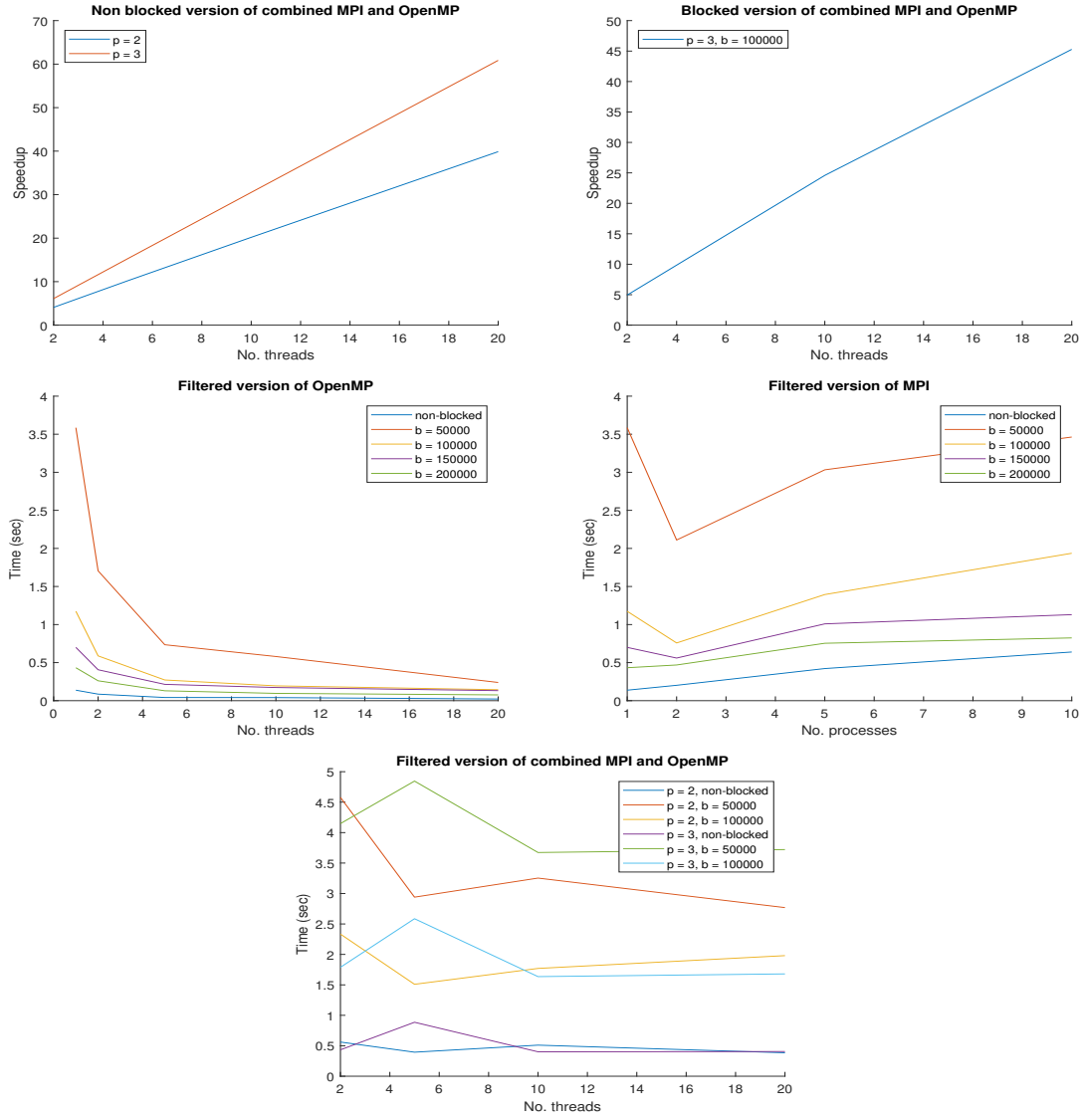

Blocked version of MPI
×10^4
b = 100000
b = 150000
Time (sec)
No. processes


Non blocked version of MPI
Speedup
No. processes


Blocked version of MPI
b = 100000
Speedup
No. processes


Non blocked version of combined MPI and OpenMP
processes(p) = 2
p = 3
Time (sec)
No. threads


Blocked version of combined MPI and OpenMP
p = 2, b = 50000
p = 3, b = 50000
p = 3, b = 100000
Time (sec)
No. threads



Τα παραπάνω αποτελέσματα προέκυψαν από τον πολλαπλασιασμό δύο πινάκων μεγέθους 10^6x10^6 με $2 \cdot 10^6$ μη μηδενικές τιμές, οι οποίοι παράχθηκαν τυχαία από το αρχείο BMM_script.m. Αρχικά, από αυτά μπορεί να συμπεράνει κανείς ότι οι δύο μέθοδοι παραλληλισμού (OpenMP και MPI) προσφέρουν πολύ παρόμοιο speedup. Για τις blocked εκδοχές του κώδικα παρατηρείται ότι όσο πιο μικρό είναι το b τόσο πιο μεγάλος γίνεται και ο χρόνος εκτέλεσης του κώδικα. Αξίζει, όμως, να σημειωθεί ότι αυτό συμβαίνει κυρίως στους πίνακες που έχουν σχετικά ομοιόμορφη κατανομή των μη μηδενικών στοιχείων τους με αποτέλεσμα ο υπερπίνακας που δημιουργείται να είναι πυκνός και να μην παραβλέπονται πολλά blocks. Σε μια δοκιμή που έγινε στον πίνακα *com-Youtube.mtx* της πρώτης εργασίας (τετραγωνικός μεγέθους τάξης 10^6 και μη μηδενικά στοιχεία $3{\cdot}10^6$), πολλαπλασιάζοντας τον με τον εαυτό του, βρέθηκε ότι στην non-blocked εκδοχή χρειάστηκαν 4 ώρες, ενώ στην blocked εκδοχή μειώθηκε στις 3.

Συνδυάζοντας τις δύο μεθόδους παραλληλισμού είναι φανερό ότι ο χρόνος εκτέλεσης βελτιώνεται δραματικά και η ταχύτερη εκτέλεση του πολλαπλασιασμού είναι 60 φορές γρηγορότερη από την σειριακή.

Τέλος, αξίζει να παρατηρήσουμε ότι όσον αφορά τις filtered εκδόσεις η χρήση μεθόδων παραλληλισμού δεν φαίνεται να προσφέρουν ιδιαίτερα οφέλη. Αντίθετα, οι υλοποιήσεις με πολλά processes σε ορισμένες περιπτώσεις μπορεί μέχρι και να είναι πιο αργές από την αντίστοιχη σειριακή. Επειδή ο πολλαπλασιασμός όταν χρησιμοποιούμε για masking κάποιον αραιό πίνακα είναι πολύ γρήγορος από μόνος του, ο χρόνος που εξοικονομούμε με την παραλληλοποίηση του δεν είναι ιδιαίτερα σημαντικός. Από την άλλη, ο χρόνος για τις επικοινωνίες αυξάνεται με αύξηση του αριθμού των processes με αποτέλεσμα, από ένα σημείο και μετά, η ύπαρξη πολλών processes να παύει να μας συμφέρει.