Παράλληλη Συνεργατική Διήθηση.

Ο αλγόριθμος Slope One.

Διπλωματική εργασία της
Καρύδη Ευθαλίας
ΑΜ: 1006

Επιβλέπων καθηγητής
Μαργαρίτης Κωνσταντίνος

Πρόγραμμα Μεταπτυχιακών Σπουδών του Τμήματος
Εφαρμοσμένης Πληροφορικής του Πανεπιστημίου
Μακεδονίας

Κατεύθυνση Συστημάτων Υπολογιστών

Θεσσαλονίκη, 2012

Ευχαριστίες

Στο σημείο αυτό θα ήθελα να ευχαριστήσω τον κ.Κωνσταντίνο Μαργαρίτη, επιβλέποντα καθηγητή μου, για την πολύτιμη καθοδήγηση και βοήθεια που μου παρείχε σε όλα τα στάδια της δημιουργίας της εργασίας αυτής, για τα ενθαρρυντικά του λόγια που αποτέλεσαν σημαντική πηγή δύναμης, καθώς και για την εμπιστοσύνη που μου έδειξε αναθέτοντάς μου αυτό το θέμα.

Τον Δρ. Εμμανουήλ Βοζαλή για το ενδιαφέρον του και την παροχή του αρχικού υλικού και τον υποψήφιο διδάκτορα Χαράλαμπο Κουζινόπουλο για τη βοήθειά του σε διάφορα ζητήματα.

Επίσης, θα ήθελα να ευχαριστήσω τους συμφοιτητές με τους οποίους περάσαμε πολλές ώρες μελέτης στο εργαστήριο, για την πολύτιμη παρουσία τους, και φυσικά τους φίλους και την οικογένειά μου.

Περίληψη

Τα συστήματα συστάσεων συνεργατικής διήθησης εισάγουν στη μεθοδολογία για την παραγωγή των συστάσεων ένα σημαντικό παράγοντα, τη γνώμη των χρηστών του συστήματος. Γι αυτό το λόγο τα συστήματα συστάσεων εμφανίζουν σημαντική διάδοση στο διαδίκτυο. Έτσι, το ολοένα αυξανόμενο πλήθος χρηστών και αντικειμένων τέτοιων συστημάτων, καθιστά αναπόφευκτη τη ραγδαία αύξηση του όγκου των δεδομένων. Μία από τις κυριότερες προκλήσεις στα συστήματα συστάσεων, που οφείλεται στην πληθώρα δεδομένων, είναι η παραγωγή συστάσεων υψηλής ποιότητας στο συντομότερο δυνατό χρονικό διάστημα. Η γρήγορη επεξεργασία των δεδομένων καθίσταται αναγκαία ώστε τα συστήματα συστάσεων να συμβαδίζουν με τις αυξημένες απαιτήσεις ταχύτητας χωρίς να θυσιάζεται η ποιότητα.

Στην παρούσα εργασία μελετούνται αλγόριθμοι συνεργατικής διήθησης και παρουσιάζονται οι παραλληλοποιημένες εφαρμογές που συναντώνται στη σύγχρονη βιβλιογραφία. Επιλέγεται προς παραλληλοποίηση ο αλγόριθμος slope one, διότι παρουσιάζει πλεονεκτήματα όπως το ότι είναι γρήγορος και αποτελεσματικός και τα δεδομένα μπορούν να ενημερώνονται δυναμικά. Μελετώνται δύο υλοποιήσεις του με χρήση των εντολών προγραμματισμού MPI και OpenMP. Τέλος, αναλύονται τα αποτελέσματα των υλοποιήσεων αυτών, σύμφωνα με τα οποία επιτυγχάνεται βελτίωση του συνολικού χρόνου εκτέλεσης εως και 9,5 φορές έναντι του ακολουθιακού.

Λέξεις κλειδιά: Συνεργατική Διήθηση, Συστήματα Συστάσεων, Αλγόριθμος Slope One, Παράλληλος Προγραμματισμός, MPI, OpenMP.

Abstract

Collaborative filtering based recommender systems introduce the users' opinion to the procedure of the recommendations' generation. Hence, recommender systems have gained wide popularity. Thus, as the number of users and items of such systems is increasing, so inevitably does the amount of data. One of the most challenging factors in recommender systems, which is caused due to the data abundance, is achieving high quality recommendations in the shortest time possible. Consequently, a great need is emerging, the achievement of quick data processing, in order to accomplish high quality recommender systems of great speed.

This thesis presents a survey of collaborative filtering algorithms and their up to date parallel implementations. The slope one algorithm is chosen for parallelization, due to the presence of advantages such as its speed and efficacy and the dynamically updatable data. Two parallel implementations are introduced, using MPI and OpenMP, and the experimental results are evaluated. According to the results, a speedup improvement of 9,5 is achieved over the sequential implementation.

Keywords: Collaborative Filtering, Recommender Systems, Slope One Algorithm, Parallel Programming, MPI, OpenMP.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Κεφάλαιο 1

Εισαγωγή.

Ένα από τα κυριότερα προβλήματα της τεχνολογίας πληροφοριών είναι ο συνεχώς αυξανόμενος όγκος δεδομένων. Ο ρυθμός με τον οποίο πληθαίνουν τα δεδομένα καθιστά την εύρεση της κατάλληλης πληροφορίας ολοένα και δυσκολότερη και ωθεί στην αύξηση του ενδιαφέροντος για την αποδοτικότερη διαχείρισή της. Η συνεργατική διήθηση προσπαθεί να αξιοποιήσει τις σχετικές με την εκάστοτε αναζήτηση πληροφορίες, χρησιμοποιώντας τις προτιμήσεις των χρηστών ενός συστήματος, με στόχο την παραγωγή αποτελεσμάτων μεγάλης ακρίβειας.

Μία από τις πιό διαδεδομένες περιπτώσεις στις οποίες εφαρμόζεται η συνεργατική διήθηση, είναι τα συστήματα συστάσεων. Σε αυτά χρησιμοποιούνται οι προτιμήσεις των χρηστών για τα αντικείμενα του συστήματος, έτσι ώστε να καθοδηγούν τη διαδικασία παραγωγής συστάσεων. Λόγω του πολύ μεγάλου όγκου δεδομένων που πρέπει να επεξεργασθούν για να παραχθούν οι συστάσεις, είναι αναγκαία η εύρεση τρόπων, ώστε να πραγματοποιείται η διαδικασία της παραγωγής τους όσο το δυνατόν ταχύτερα. Η ανάπτυξη παράλληλων και κατανεμημένων συστημάτων επιτρέπει τη γρήγορη επεξεργασία μεγάλου όγκου δεδομένων. Επομένως, η εφαρμογή τους στα συστήματα συστάσεων συνεργατικής διήθησης παρουσιάζει μεγάλη αποδοχή.

Στόχος της παρούσας εργασίας είναι η μελέτη των κυριότερων αλγορίθμων συνεργατικής διήθησης και η εφαρμογή τεχνικών παραλληλοποίησης σε έναν από αυτούς, με σκοπό την υλοποίηση μιας ταχύτερης εφαρμογής. Επιπλέον, παρουσιάζονται οι παράλληλες εφαρμογές συνεργατικής διήθησης που έχουν υλοποιηθεί τα τελευταία χρόνια, με σκοπό να αποτελέσει η παρουσίασή τους τη βάση για τη συλλογική αξιολόγηση και σύγκρισή τους.

Παρακάτω περιγράφεται η δομή που ακολουθείται στη συνέχεια της εργασίας. Στο δεύτερο κεφάλαιο περιγράφονται οι βασικές έννοιες των συστημάτων συστάσεων συνεργατικής διήθησης, παρουσιάζονται οι κυριότεροι αλγόριθμοι συνεργατικής διήθησης και οι μετρικές αξιολόγησής τους. Τέλος, γίνεται αναφορά στα κυριότερα προβλήματα των συστημάτων συνεργατικής διήθησης και στις προκλήσεις που παρουσιάζονται στην υλοποίησή τους.

Στο τρίτο κεφάλαιο εξηγούνται οι βασικές έννοιες της παράλληλης επεξεργασίας και οι τεχνικές ανάπτυξης παράλληλων και κατανεμημένων εφαρμογών. Γίνεται μία σύντομη περιγραφή των λειτουργιών των διεπαφών OpenMP και MPI και περιγράφονται οι μετρικές αξιολόγησης παράλληλων προγραμμάτων. Στη συνέχεια, παρατίθεται ένα παράδειγμα παραλληλοποίησης στη γλώσσα προγραμματισμού C.

Στο τέταρτο κεφάλαιο δίνεται μία περιγραφή του τρόπου λειτουργίας των προγραμματιστικών πλαισίων που χρησιμοποιούνται για την υλοποίηση παράλληλων εφαρμογών συνεργατικής διήθησης.

Στο πέμπτο κεφάλαιο περιγράφονται οι παράλληλοι αλγόριθμοι συνεργατικής διήθησης, που αποτελούν την τάση στη σύγχρονη έρευνα του κλάδου. Πραγματοποιείται με αυτό τον τρόπο μία απόπειρα συγκέντρωσης των παράλληλων εφαρμογών συνεργατικής διήθησης και επισημαίνεται η ανάγκη συγκριτικής αξιολόγησής τους.

Στο έκτο κεφάλαιο παρουσιάζεται αναλυτικά ο αλγόριθμος slope one και υλοποιούνται δύο παράλληλες εφαρμογές του. Μία σε σύστημα μοιραζόμενης μνήμης, με χρήση της διεπαφής OpenMP, και μία σε υβριδικό σύστημα μοιραζόμενης και κατανεμημένης μνήμης με OpenMP και MPI. Πραγματοποιείται πειραματική μελέτη των εφαρμογών αυτών και αναλύονται τα αποτελέσματα που προκύπτουν.

Στη συνέχεια ακολουθούν τα συμπεράσματα που πηγάζουν από την εκπόνηση της εργασίας και δίνονται προτάσεις για την περαιτέρω ενασχόληση με το θέμα της παράλληλης συνεργατικής διήθησης.

Ακολουθούν τα παραρτήματα, στα οποία περιγράφονται τα σύνολα δεδομένων που χρησιμοποιήθηκαν, οι θεωρητικοί υπολογισμοί του χρόνου επικοινωνίας για την υβριδική εφαρμογή, παρατίθενται πίνακες που περιέχουν τα αποτελέσματα των πειραμάτων και περιγράφεται ένα αριθμητικό παράδειγμα του αλγορίθμου slope one. Τέλος, επισυνάπτεται ένας δίσκος δεδομένων που περιέχει τον πηγαίο κώδικα των εφαρμογών που υλοποιήθηκαν και τα σύνολα δεδομένων.

Κεφάλαιο 2

Συνεργατική Διήθηση και Συστήματα Συστάσεων.

Πάντα οι άνθρωποι αξιοποιούσαν τη γνώμη άλλων για να βοηθηθούν στο σχηματισμό απόψεως πάνω σε κάποιο αντικείμενο και να δημιουργήσουν μια πρώτη εικόνα για το αν αυτό είναι συναφές ή όχι προς τα ενδιαφέροντά τους. Αυτή η δραστηριότητα χρησιμοποιείται σήμερα στην τεχνολογία μέσω των συστημάτων συστάσεων (Recommender Systems).

Τα συστήματα συστάσεων είναι μηχανισμοί παραγωγής συστάσεων για τα αντικείμενα ενός συνόλου. Για το σχηματισμό των συστάσεων χρησιμοποιούνται αλγόριθμοι που επεξεργάζονται ένα σύνολο δεδομένων το οποίο αποτελείται από αξιολογήσεις (ratings) χρηστών του συστήματος για κάποια από τα αντικείμενα του συστήματος. Οι συστάσεις μπορούν να παραχθούν με διάφορους τρόπους, όπως, βασιζόμενες στις προηγούμενες αξιολογήσεις αντικειμένων από τον χρήστη για τον οποίο εφαρμόζεται η διαδικασία παραγωγής συστάσεων ή σύμφωνα με το πλήθος και την ποιότητα των αξιολογήσεων που υπάρχουν για κάθε αντικείμενο.

Οι αξιολογήσεις των χρηστών του συστήματος είναι στην πραγματικότητα μία συσχέτιση μεταξύ χρηστών και αντικειμένων. Κάθε χρήστης του συστήματος έχει τη δυνατότητα να εκφράσει τη γνώμη του για τα αντικείμενα του συστήματος μέσω των αξιολογήσεων. Οι αξιολογήσεις μπορούν είτε να έχουν τη μορφή κλίμακας τιμών, εκφράζοντας έτσι ένα μέτρο για το πόσο αρεστό είναι ένα αντικείμενο στο χρήστη, είτε να υπάρχουν μόνο δύο τιμές αξιολόγησης, μία για την έγκριση ενός αντικειμένου και μία για την απόρριψή του. Ακόμα, είναι δυνατόν να υπάρχει μία μόνο τιμή που να δηλώνει την ύπαρξη έμμεσης αξιολόγησης, δηλαδή ότι ο χρήστης αγόρασε ή είδε ένα αντικείμενο, αλλά σε αυτή την περίπτωση η παρουσία αξιολόγησης δεν διευκρινίζει αν ο χρήστης που είδε το αντικείμενο τελικά σχημάτισε για αυτό θετική γνώμη ή αν τελικά το απέρριψε.

Η διαδικασία διαχωρισμού και εκτίμησης αντικειμένων μέσω της γνώμης άλλων, περιγράφεται με τον όρο ″συνεργατική διήθηση″(Collaborative filtering). Τα συστήματα συστάσεων που χρησιμοποιούν τις μεθόδους συνεργατικής διήθησης παράγουν συστάσεις ή προβλέψεις για συγκεκριμένο χρήστη και για ένα ή περισσότερα αντικείμενα. Οι συστάσεις είναι σύνολα αντικειμένων που σχηματίζει το σύστημα, τα οποία θεωρεί ότι θα αρέσουν στο χρήστη, χωρίς να προσδιορίζεται πόσο αρεστά θα είναι. Ενώ οι προβλέψεις είναι συγκεκριμμένες τιμές που παράγονται για κάθε αντικείμενο και αντιπροσωπεύουν την τιμή που το σύστημα θεωρεί ότι ο χρήστης θα έδινε για να αξιολογήσει το κάθε αντικείμενο. Η χρήση των

αλγορίθμων συνεργατικής διήθησης είναι πολύ διαδεδομένη στα συστήματα συστάσεων και η ανάγκη επεξεργασίας ολοένα και μεγαλύτερου όγκου δεδομένων δίνει συνεχώς ώθηση στην επινόηση νέων γρηγορότερων και αποδοτικότερων αλγορίθμων, που στοχεύουν στην παραγωγή προσωποποιημένων και ποιοτικών για τους χρήστες συστάσεων.

Οι βασικές λειτουργίες των συστημάτων συνεργατικής διήθησης συνοψίζονται στη σύσταση αντικειμένων χρησιμοποιώντας το μέσο αριθμό εκτιμήσεων του χρήστη χωρίς να παράγουν προσωποποιημένες προβλέψεις εκτιμήσεων, ή στην παραγωγή προβλέψεων για δεδομένο αντικείμενο ή για δεδομένο σύνολο αντικειμένων. Ο υπολογισμός των προβλέψεων μπορεί να είναι πολύ πιο απαιτητικός από την παραγωγή συστάσεων [48], όμως τα αποτελέσματα που αποφέρει η χρήση τους είναι προσωποποιημένα και έτσι είναι πιθανότερο να ικανοποιήσουν τις απαιτήσεις του χρήστη και να αυξήσουν την εμπιστοσύνη του στο σύστημα συστάσεων.

Στη συνέχεια της ενότητας αυτής θα παρουσιασθούν μερικοί από τους σημαντικότερους αλγόριθμους συνεργατικής διήθησης και οι κυριότερες μετρικές που χρησιμοποιούνται για την αξιολόγησή τους.

2.1 Σημειογραφία.

Οι αλγόριθμοι συνεργατικής διήθησης χρησιμοποιούνται για την παραγωγή συστάσεων αντικειμένων σε χρήστες. Το σύνολο των χρηστών συμβολίζεται με $U = \{u_1, u_2, ..., u_m\}$ και το σύνολο των αντικειμένων με $I = \{i_1, i_2, ..., i_n\}$. Με $I_u \subseteq I$ συμβολίζεται το σύνολο των αντικειμένων που ο χρήστης u έχει αξιολογήσει, και με $U_i \subseteq U$ το σύνολο των χρηστών που έχουν αξιολογήσει το αντικείμενο i.

Ο πίνακας R περιέχει τις αξιολογήσεις των χρηστών για τα αντικείμενα. Κάθε σειρά του πίνακα αυτού αντιστοιχεί σε ένα χρήστη και κάθε στήλη σε ένα αντικείμενο. Ο πίνακας αυτός είναι πολύ αραιός, γεγονός που αποτελεί ένα από τα κυριότερα προβλήματα των συστημάτων συστάσεων συνεργατικής διήθησης. Με $r_{i,j}$ συμβολίζεται η αξιολόγηση του χρήστη i για το αντικείμενο j και με $pred(i, j)$ η πρόβλεψη της αξιολόγησης του χρήστη i για το αντικείμενο j. Με $\bar{r}_i$ συμβολίζεται η μέση τιμή των εκτιμήσεων του χρήστη i για τους αλγορίθμους βάσει χρηστών (user based) ή η μέση τιμή όλων των αξιολογήσεων του αντικειμένου i στους αλγορίθμους βάσει αντικειμένων (item based). Η εγγύτητα μεταξύ των χρηστών i και k συμβολίζεται με $sim(i, k)$.

2.2 Αλγόριθμοι συνεργατικής διήθησης.

Τα συστήματα συστάσεων κατηγοριοποιούνται σύμφωνα με τον τρόπο με τον οποίο γίνονται οι συστάσεις, σε συστήματα που βασίζονται στο περιεχόμενο των αντικειμένων, σε συστήματα που παράγουν συστάσεις με συνεργατική διήθηση και σε υβριδικές εφαρμογές που περιλαμβάνουν στοιχεία και από τους δύο τρόπους προσέγγισης.

Τα συστήματα συνεργατικής διήθησης διακρίνονται σε συστήματα που χρησιμοποιούν τεχνικές βάσει μνήμης και σε συστήματα που χρησιμοποιούν τεχνικές βάσει μοντέλου. Επίσης κατηγοριοποιούνται σε πιθανοτικά και σε μή-πιθανοτικά συστήματα. Τα συστήματα που εφαρμόζουν αλγόριθμους βάσει μνήμης, παράγουν τις προβλέψεις των αξιολογήσεων βασι-

ζόμενα σε ολόκληρο το σύνολο των αντικειμένων που έχουν προηγουμένως αξιολογηθεί από τους χρήστες. Για να υπολογισθεί η πρόβλεψη της αξιολόγησης ενός αντικειμένου από ένα χρήστη, λαμβάνονται υπόψη όλες οι προβλέψεις άλλων χρηστών για το ίδιο αντικείμενο. Σε αυτή τη διαδικασία συνήθως συνυπολογίζονται οι χρήστες με τη μεγαλύτερη εγγύτητα. Σε αντίθεση με αυτά τα συστήματα, τα συστήματα που εφαρμόζουν αλγόριθμους βάσει μοντέλου, χρησιμοποιούν τις αξιολογήσεις για να μάθουν ένα μοντέλο και στη συνέχεια χρησιμοποιούν το μοντέλο για να παράγουν τις προβλέψεις. Στη συνέχεια παρουσιάζονται συνοπτικά αλγόριθμοι συνεργατικής διήθησης που ανήκουν στις παραπάνω κατηγορίες [48].

Ένας από τους πιό βασικούς αλγόριθμους για την παραγωγή προβλέψεων είναι ο αλγόριθμος βάσει μέσης πρόβλεψης χρήστη (Per user average). Σε αυτόν, η πρόβλεψη υπολογίζεται σύμφωνα με την εξίσωση $pred(u,i) = \bar{u}$, όπου $\bar{u}$ είναι ο μέσος όρος των προβλέψεων του χρήστη u [27].

Ένας αλγόριθμος βάση μοντέλου είναι ο αλγόριθμος προσαρμοσμένου συνημιτόνου βάσει αντικειμένων (Adjusted Cosine Item Based) [47, 35]. Σε αυτό τον αλγόριθμο η πρόβλεψη γίνεται ώς εξής:

$$pred(u,j) = \frac{\sum_{i=1}^{n} r_{ui} \cdot sim(u,i)}{\sum_{i=1}^{n} sim(j,i)}$$

Όπου $sim(i,j)$ είναι η μετρική ομοιότητας συνημιτόνου μεταξύ των αντικειμένων i και j που υπολογίζεται ώς εξής:

$$sim(i,j) = \frac{\sum_{l=i}^{k}(r_{ui} - \bar{r_u})(r_{uj} - \bar{r_u})}{\sqrt{\sum_{l=1}^{k}(r_{ui} - \bar{r_u})^2}\sqrt{\sum_{l=1}^{k}(r_{uj} - \bar{r_u})^2}}$$

2.2.1 Αλγόριθμοι βάσει χρηστών.

Οι αλγόριθμοι αυτοί είναι από τους πιό διαδεδομένους αλγόριθμους συνεργατικής διήθησης. Η διαδικασία παραγωγής συστάσεων εκτελείται σε τρία βήματα:

- Υπολογισμός της εγγύτητας μεταξύ του χρήστη για τον οποίο παράγονται οι συστάσεις και των υπόλοιπων χρηστών του συστήματος.
- Επιλογή ενός υποσυνόλου των χρηστών (σχηματισμός γειτονιάς) σύμφωνα με την εγγύτητά τους με το χρήστη για τον οποίο παράγονται οι συστάσεις.
- Υπολογισμός των προβλέψεων χρησιμοποιώντας τις αξιολογήσεις των χρηστών της γειτονιάς που επιλέχθηκε στο προηγούμενο βήμα.

Σε αυτή την οικογένεια αλγορίθμων περιλαμβάνονται αλγόριθμοι που συνδυάζουν διάφορες τεχνικές για τον υπολογισμό κάθε βήματος.
Για τον υπολογισμό της εγγύτητας μεταξύ των χρηστών, οι πιό διαδεδομένες τεχνικές είναι:

- Ο συντελεστής συσχέτισης Pearson (Pearson Correlation Coefficient).

  Υπολογίζεται συγκρίνοντας τις αξιολογήσεις για όλα τα αντικείμενα που έχουν εκτιμηθεί ταυτόχρονα από τους χρήστες της γειτονιάς και από τον χρήστη για τον οποίο παράγονται οι συστάσεις.

$$sim(i,k) = \frac{\sum_{j=1}^{l}(r_{ij} - \bar{r_i})(r_{kj} - \bar{r_k})}{\sqrt{\sum_{j=1}^{l}(r_{ij} - \bar{r_i})^2 \sum_{j=1}^{l}(r_{kj} - \bar{r_k})^2}}$$

Υπολογίζεται η εγγύτητα των χρηστών u_i και u_k για τα l αντικείμενα για τα οποία καί οι δύο χρήστες έχουν δώσει αξιολόγηση στο σύστημα. Υπάρχουν και παραλλαγές αυτής της μετρικής, όπως ο συντελεστής συσχέτισης Pearson με περιορισμούς, όπου υπολογίζεται η εγγύτητα μεταξύ των χρηστών όπως προηγουμένως, αλλά αντί για τη μέση εκτίμηση του χρήστη χρησιμοποιείται η μέση εκτίμηση βάσει της κλίμακας που χρησιμοποιείται, και ο συντελεστής συσχέτισης Pearson με χρήση βαρών, όπου λαμβάνεται υπόψη το πλήθος των κοινών αντικειμένων που έχουν αξιολογήσει οι χρήστες [17].

- Εγγύτητα συνημιτόνου - διανύσματος (Cosine/Vector Similarity).

  Οι χρήστες μπορούν να θεωρηθούν ώς διανύσματα διαστάσεως ίσης με το πλήθος των αντικειμένων, στα οποία περιέχονται οι τιμές των εκτιμήσεων του χρήστη για το κάθε αντικείμενο. Η εγγύτητα μεταξύ των χρηστών u_i και u_k υπολογίζεται με τη βοήθεια του συνημιτόνου της γωνίας μεταξύ των διανυσμάτων των χρηστών [15]. Αν η τιμή είναι κοντά στο 1 η εγγύτητα μεταξύ των χρηστών είναι μεγάλη, ενώ αν είναι κοντά στο 0 είναι μικρή.

$$sim(i,k) = \sum_j \frac{r_{ij}}{\sqrt{\sum_j r_{ij}^2}} \frac{r_{kj}}{\sqrt{\sum_j r_{kj}^2}}$$

- Μέση τετραγωνική διαφορά.

  Υπολογίζει την εγγύτητα των χρηστών υπολογίζοντας τη μέση διαφορά των αξιολογήσεων των αντικειμένων που έχουν εκτιμήσει καί οι δύο χρήστες. Χρησιμοποιείται ώς κατώφλι η τιμή L και οι διαφορές που είναι μεγαλύτερες από αυτή την τιμή δεν λαμβάνονται υπόψην.

$$sim(i,k) = \frac{L - \frac{\sum_{j \in I_i \cap I_k}(u_{ij} - u_{kj})^2}{|I_i \cap I_k|}}{L}$$

  Όπου $|I_i \cap I_k|$ είναι το πλήθος των αντικειμένων που έχουν εκτιμήσει ταυτόχρονα οι χρήστες u_i και u_k και I_i το σύνολο των αντικειμένων που έχει εκτιμήσει ο χρήστης u_i.

Η επιλογή των χρηστών της γειτονιάς γίνεται συνήθως επιλέγοντας τους χρήστες των οποίων η εγγύτητα είναι μεγαλύτερη από την τιμή ενός προκαθορισμένου κατωφλιού ή επιλέγοντας Ν χρήστες με τη μεγαλύτερη εγγύτητα με το χρήστη για τον οποίο παράγονται οι προβλέψεις. Ένας άλλος τρόπος επιλογής της γειτονιάς είναι με χρήση του σχήματος αθροιστικής γειτονιάς (Aggregate Neighborhood Scheme) [45], που είναι πολύ χρήσιμο κυρίως στα πολύ αραιά σύνολα δεδομένων διότι επιτρέπει όλοι οι χρήστες που συμπεριλαμβάνονται στη γειτονιά να την επηρεάσουν.

Ο υπολογισμός των συστάσεων μπορεί να γίνει με διάφορους τρόπους, όπως την επιλογή από τα αντικείμενα που έχουν εκτιμήσει οι χρήστες της επιλεγμένης γειτονιάς, αυτών που εμφανίζονται συχνότερα, δηλαδή που έχουν εκτιμηθεί από τους περισσότερους χρήστες της γειτονιάς ή με χρήση κανόνων συσχέτισης.

Ο υπολογισμός της γειτονιάς των χρηστών είναι ακριβός υπολογιστικά, διότι απαιτείται σύγκριση ενός χρήστη με όλους τους υπόλοιπους χρήστες του συστήματος. Για τη μείωση του χρόνου υπολογισμού της γειτονιάς, αλλά και της απαραίτητης μνήμης εφαρμόζονται τεχνικές δειγματοληψίας ή συσταδοποίησης.

Οι προβλέψεις της αξιολόγησης του χρήστη i για το αντικείμενο k, όταν η γειτονιά των χρηστών περιέχει l χρήστες δίνονται από τον παρακάτω τύπο, όπου με r_j συμβολίζεται η εκτίμηση του χρήστη i για το αντικείμενο j [53].

$$pred(i,k) = \bar{r_i} + \frac{\sum_{j=1}^{l} sim(j,k) * (r_j - \bar{r_k})}{\sum_{j=1}^{l} sim(j,k)}$$

2.2.2 Αλγόριθμοι βάσει αντικειμένων.

Οι αλγόριθμοι αυτής της κατηγορίας ακολουθούν την ίδια λογική με τους αλγόριθμους βάσει χρηστών, με τη διαφορά ότι υπολογίζουν εγγύτητα μεταξύ αντικειμένων. Το μεγαλύτερο πλεονέκτημα που παρουσιάζουν αυτοί οι αλγόριθμοι είναι ότι η εγγύτητα μεταξύ των αντικειμένων είναι αρκετά στατική, ώστε να επιτρέπει να γίνεται η πραγματοποίηση των υπολογισμών για την εύρεση γειτονιάς online. Επίσης, οι αλγόριθμοι αυτοί παρουσιάζουν πολύ καλύτερη απόδοση από τους αλγόριθμους βάσει χρηστών [47]. Η διαδικασία παραγωγής συστάσεων είναι παρόμοια με αυτή των αλγορίθμων βάσει χρηστών, υπολογίζεται η εγγύτητα μεταξύ αντικειμένων, επιλέγονται αυτά που έχουν μεγαλύτερη εγγύτητα με το αντικείμενο για το οποίο παράγεται η πρόβλεψη, και στη συνέχεια χρησιμοποιούνται τα επιλεγμένα αντικείμενα στην παραγωγή της πρόβλεψης.

Για τον υπολογισμό της εγγύτητας μεταξύ αντικειμένων χρησιμοποιούνται μετρικές όπως:

- Ο συντελεστής συσχέτισης Pearson.

$$sim(j,k) = \frac{\sum_{i=1}^{l}(r_{ij} - \bar{r_j})(r_{ik} - \bar{r_k})}{\sqrt{\sum_{i=1}^{l}(r_{ij} - \bar{r_j})^2 \sum_{i=1}^{l}(r_{ik} - \bar{r_k})^2}}$$

- Η εγγύτητα συνημιτόνου - διανύσματος (Cosine/Vector similarity).

$$sim(j,k) = \sum_{i=1}^{l} \frac{r_{ij}}{\sqrt{\sum_{i=1}^{l} r_{ij}^2}} \frac{r_{ik}}{\sqrt{\sum_{i=1}^{l} r_{ik}^2}}$$

- Η προσαρμοσμένη εγγύτητα συνημιτόνου (Adjusted Cosine Similarity).

$$sim(j,k) = \frac{\sum_{i=1}^{l}(r_{ij} - \bar{r_i})(r_{ik} - \bar{r_i})}{\sqrt{\sum_{i=1}^{l}(r_{ij} - \bar{r_i})^2 \sum_{i=1}^{l}(r_{ik} - \bar{r_i})^2}}$$

Ο υπολογισμός των προβλέψεων μπορεί να γίνει με τη χρήση βεβαρυμένου αθροισμάτος (weighted sum) όπου προσθέτονται οι εκτιμήσεις που έχει δώσει στο σύστημα ο χρήστης

για τον οποίο παράγονται οι προβλέψεις πολλαπλασιασμένες με την εγγύτητα μεταξύ των αντικειμένων. Ο υπολογισμός της πρόβλεψης της αξιολόγησης του χρήστη i για το αντικείμενο k, όταν ο χρήστης i έχει αξιολογήσει l αντικείμενα δίνεται από τον εξής τύπο, όπου r_j είναι η εκτίμηση του χρήστη i για το αντικείμενο j:

$$pred(i,k) = \frac{\sum_{j=1}^{l} sim(j,k) * r_j}{\sum_{j=1}^{l} |sim(j,k)|}$$

Οι προβλέψεις μπορούν να προσεγγισθούν και με παλινδρόμηση. [47, 15]

2.2.3 Αλγόριθμοι που χρησιμοποιούν τεχνικές μείωσης διαστάσεων.

Οι μέθοδοι παραγοντοποίησης πινάκων, όπως η SVD (Singular Value Decomposition), εφαρμόζονται στα συστήματα συστάσεων για να εντοπισθούν οι λανθάνουσες σχέσεις μεταξύ των χρηστών και των αντικειμένων και να παραχθεί η πρόβλεψη για το πόσο αρεστό θα είναι ένα συγκεκριμένο αντικείμενο σε έναν χρήστη. Παράγουν μια αναπαράσταση χαμηλών διαστάσεων του πίνακα R των αξιολογήσεων των χρηστών για τα αντικείμενα του συστήματος επιτρέποντας έτσι στους αλγορίθμους που βασίζονται στη συνεργατική διήθηση να επιτυγχάνουν καλή κλιμάκωση στα μεγάλα σύνολα δεδομένων και ταυτόχρονα να παράγουν ποιοτικές συστάσεις [45].

Άλλες μέθοδοι για τη μείωση των διαστάσεων είναι επίσης δημοφιλείς, όπως η LSI (Latent Semantic Indexing), που χρησιμοποιεί την SVD, και η Ανάλυση Κύριων Συνιστωσών, PCA (Principal Component Analysis), που υπολογίζει τις ιδιοτιμές των πινάκων που περιέχουν τις τιμές εγγύτητας μεταξύ χρηστών ή αντικειμένων και για τις k μεγαλύτερες επιστρέφει τα αντίστοιχα ιδιοδιανύσματα δημιουργώντας έτσι έναν k-διάστατο χώρο. Οι κυριότερες μέθοδοι μείωσης διαστάσεων περιγράφονται στη συνέχεια.

2.2.3.1 Η μέθοδος SVD.

Στη μέθοδο SVD ο $m \times n$ πίνακας R με $m \geq n$ αναπαρίσταται ώς $R = USV'$, όπου οι πίνακες U και V είναι ορθοκανονικοί διαστάσεων $m \times m$ και $n \times n$ αντίστοιχα, $r = rank(R)$ και ο πίνακας S είναι διαγώνιος διάστασης $m \times n$.

Η SVD υπολογίζεται βρίσκοντας πρώτα τις ιδιοτιμές και τα ιδιοδιανύσματα των πινάκων RR' και $R'R$. Τα ιδιοδιανύσματα του $R'R$ σχηματίζουν τις στήλες του πίνακα V και τα ιδιοδιανύσματα του RR' τις στήλες του πίνακα U. Οι τιμές στον πίνακα S είναι οι τετραγωνικές ρίζες των ιδιοτιμών ενός από τους πίνακες RR' ή $R'R$ και είναι ταξινομημένες στην κύρια διαγώνιο του S κατά φθίνουσα τάξη. Ο πίνακας S περιέχει τόσα μή μηδενικά στοιχεία στην κύρια διαγώνιό του όση και η τάξη του πίνακα R. Είναι δηλαδή της μορφής:

$$S = \begin{pmatrix} \sigma_1 & 0 & \dots & 0 & \\ 0 & \sigma_2 & \dots & 0 & \\ \vdots & \vdots & \ddots & \vdots & \mathbf{0} \\ 0 & 0 & \dots & \sigma_r & \\ & & \mathbf{0} & & \end{pmatrix}$$

όπου $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r$ και $\sigma_i \neq 0$ [1, 51].

Ο SVD αλγόριθμος για συνεργατική διήθηση ξεκινάει υπολογίζοντας τον πίνακα χρηστών-αντικειμένων R στον οποίο περιέχονται οι αξιολογήσεις των χρηστών για τα αντικείμενα. Ο πίνακας αυτός είναι πολύ αραιός, και με τη χρήση μεθόδων μείωσης διαστάσεων, η αναπαράστασή του περιορίζεται σε μία προσέγγισή του, τάξεως r [45]. Στη συνέχεια υπολογίζεται η μέση τιμή $\bar{r}_i$ κάθε γραμμής του R και η μέση τιμή $\bar{c}_j$ κάθε στήλης του R. Οι κενές θέσεις του πίνακα R συμπληρώνονται με την αντίστοιχη τιμή $\bar{c}_j$ κάθε στήλης και έτσι λαμβάνεται ένας γεμάτος πίνακας R_f. Από τα στοιχεία αυτού του πίνακα αφαιρείται η αντίστοιχη τιμή $\bar{r}_i$ κάθε γραμμής και λαμβάνεται ο πίνακας A.
Στον πίνακα A εφαρμόζεται SVD και από τον πίνακα S που προκύπτει διατηρούνται μόνο τα πρώτα k στοιχεία της κύριας διαγωνίου. Έτσι, λαμβάνονται οι πίνακες S_k, διαστάσεων $k \times k$, U_k, διαστάσεων $m \times k$ και V_k, διαστάσεων $k \times n$ [46], [16].
Σε αυτό τον αλγόριθμο συνυπολογίζεται και η εγγύτητα μεταξύ χρηστών και η εγγύτητα μεταξύ χρηστών και αντικειμένων. Μία ακόμα διαφορά από τον κλασικό αλγόριθμο συνεργατικής διήθησης είναι ότι για τον υπολογισμό των προβλέψεων δεν υπολογίζονται βάρη. [15]
Η παραγωγή προβλέψεων για τον χρήστη i και το αντικείμενο j γίνεται ώς εξής:

$$pred(i,j) = \bar{r}_i + U_k \sqrt{S_k^T(i)} \sqrt{S_k} V_k^T(j).$$

Πολλές παραλλαγές αυτού του αλγόριθμου έχουν μελετηθεί, όπως η SVD++ που είναι μια διαφοροποίηση της κανονικοποιημένης SVD, η οποία λαμβάνει υπόψη χαρακτηριστικά χρηστών και αντικειμένων, και μία προσέγγιση της SVD που ενσωματώνει στο μοντέλο μεθόδους που βασίζονται στην εύρεση γειτονιάς, που παρουσιάζονται στο [26]. Στο [3] παρουσιάζεται μία βελτιωμένη έκδοση της κανονικοποιημένης SVD προσθέτοντας βάρη, και μία με χρήση λιγότερων παραμέτρων. Αλγόριθμοι που βασίζονται στην SVD μέθοδο παρουσιάζονται στο [16] και πολλές παραλλαγές της SVD μεθόδου αναλύονται στο [29].

2.2.3.2 Ανάλυση Κύριων Συνιστωσών PCA.

Η Ανάλυση Κύριων Συνιστωσών (PCA) είναι μία μέθοδος που βρίσκει ένα μετασχηματισμό των αρχικών μεταβλητών του συνόλου δεδομένων σε ένα σύνολο μή συσχετιζόμενων μεταβλητών, έτσι ώστε να αναπαρίσταται όσο το δυνατόν περισσότερο μέρος των πληροφοριών με λιγότερες μεταβλητές.

Σχηματίζεται ένας πίνακας με τα δεδομένα, όπου κάθε στήλη περιέχει τις τιμές ενός χαρακτηριστικού και αφαιρείται ο μέσος κάθε χαρακτηριστικού από τα δεδομένα. Στη συνέχεια υπολογίζεται ο πίνακας των συνδιακυμάνσεων και οι ιδιοτιμές και τα ιδιοδιανύσματά του. Τα ιδιοδιανύσματα που αντιστοιχούν στις μικρότερες ιδιοτιμές αγνοούνται και δημιουργείται ένας πίνακας που περιέχει στις στήλες του τα υπόλοιπα. Πολλαπλασιάζοντας τον ανάστροφο

αυτού του πίνακα από αριστερά με τον ανάστροφο του αρχικού πίνακα δεδομένων, προκύπτει ένας πίνακας μικρότερων διαστάσεων από τον αρχικό, που περιέχει τα αρχικά δεδομένα συναρτήσει των ιδιοδιανυσμάτων που έχουν επιλεχθεί [12].

Ο αλγόριθμος PCA για συνεργατική διήθηση [16], στις κενές θέσεις του πίνακα R των αξιολογήσεων των χρηστών για αντικείμενα, τοποθετεί το μέσο κάθε γραμμής $\bar{r_i}$, δηλαδή τη μέση τιμή των αξιολογήσεων κάθε χρήστη. Με αυτό τον τρόπο λαμβάνεται ο πίνακας R_f. Αφαιρείται ο μέσος κάθε στήλης $\bar{c_j}$ από τον R_f και λαμβάνεται ο πίνακας A. Στη συνέχεια εφαρμόζεται SVD στον A και διατηρούνται οι πρώτες k ιδιοτιμές. Ο πίνακας που κατασκευάζεται με αυτό τον τρόπο συμβολίζεται με A_k. Η πρόβλεψη της εκτίμησης του χρήστη i για το αντικείμενο j παράγεται σύμφωνα με τον παρακάτω τύπο:

$$pred(i,j) = \bar{c_j} + U_k\sqrt{S_k^T(i)}\sqrt{S_k}V_k^T(j).$$

2.2.3.3 Στοχαστική Μείωση Κλίσης SGD.

Ο αλγόριθμος SGD (Stochastic Gradient Descent) μειώνει τις διαστάσεις του αρχικού συνόλου δεδομένων σε f. Κάθε αντικείμενο συσχετίζεται με ένα διάνυσμα $q_i \in \mathbb{R}^f$. Τα στοιχεία του q_i εκφράζουν τη συσχέτιση του αντικειμένου i με κάθε έναν από τους παράγοντες που χαρακτηρίζουν τα αντικείμενα. Για κάθε χρήστη αντίστοιχα ορίζεται ένα διάνυσμα p_u, τα στοιχεία του οποίου εκφράζουν το ενδιαφέρον του χρήστη προς τα κυριότερα αντικείμενα που ανήκουν σε κάθε έναν από τους παράγοντες.

Ο αλγόριθμος υπολογίζει για κάθε χρήστη και αντικείμενο του συνόλου δεδομένων την πρόβλεψη της τιμής $r_{u,i}$ και το σφάλμα αυτής, $e_{u,i} = r_{u,i} - q_i'p_u$.
Στη συνέχεια τροποποιεί τις παραμέτρους κατά ποσότητα ανάλογη του γ, στην αντίθετη κατεύθυνση από αυτή της παραγόμενης κλίσης του προηγούμενου βήματος, σύμφωνα με τους παρακάτω τύπους [61].
$q_i \leftarrow q_i + \gamma(e_{u,i} \cdot p_u - \lambda \cdot q_i)$
$p_u \leftarrow p_u + \gamma(e_{u,i} \cdot q_i - \lambda \cdot p_u)$

Η προβλεπόμενη αξιολόγηση υπολογίζεται από τον τύπο:

$$pred(u,i) = \mu + b_i + b_u + q_i'p_u$$

και οι παράμετροι b_i, b_u και μ λαμβάνονται από την ελαχιστοποίηση της παράστασης:

$$min \sum (r_{u,i} - \hat{r_{u,i}})^2 + \lambda(b_i^2 + b_u^2 + ||q_i||^2 + ||p_u||^2)$$

όπου με $||.||^2$ συμβολίζεται η νόρμα Frobenius [58].

2.2.3.4 Εναλλαγή ελαχίστων τετραγώνων ALS.

Ο πίνακας R διαστάσεων $m \times n$ παραγοντοποιείται στους πίνακες U και V διαστάσεων $m \times d$ και $d \times n$ αντίστοιχα.

Η μέθοδος αυτή προσπαθεί να υπολογίσει τους πίνακες U και V. Αρχικοποιεί πρώτα τον πίνακα U, θέτοντας τη μέση αξιολόγηση ενός αντικειμένου στην πρώτη σειρά και μικρές

τυχαίες τιμές στον υπόλοιπο πίνακα. Διατηρώντας σταθερό τον πίνακα U, λύνει για τον πίνακα V ελαχιστοποιώντας την αντικειμενική συνάρτηση, δηλαδή το άθροισμα των τετραγωνικών σφαλμάτων. Στη συνέχεια, διατηρώντας σταθερό τον πίνακα V, λύνει για τον U, ελαχιστοποιώντας την αντικειμενική συνάρτηση με παρόμοιο τρόπο όπως προηγουμένως. Επαναλαμβάνει αυτά τα βήματα μέχρι την ικανοποίηση ενός κριτήριου [64].

Η πρόβλεψη της αξιολόγησης υπολογίζεται ώς το διανυσματικό γινόμενο των διανυσμάτων χαρακτηριστικών των χρηστών και των αντικειμένων [58].

$$pred(u,i) = \sum_{j=1}^{d} U_{u,j} V_{j,i}$$

Μία παραλλαγή της μεθόδου αυτής είναι η BPTF (Bayesian Probabilistic Tensor Factorization) που περιγράφεται στο [59], σύμφωνα με την οποία, στο μοντέλο συμπεριλαμβάνονται πληροφορίες σχετικές με το χρόνο και εφαρμόζεται η μέθοδος Μαρκοβιανής αλυσίδας Monte Carlo. Επίσης, στο [44] περιγράφεται η μέθοδος πιθανοτικής παραγοντοποίησης πινάκων (Probabilistic Matrix Factorization, PMF), που μοντελοποιεί τον πίνακα που περιέχει τις προτιμήσεις των χρηστών σε γινόμενο δύο πινάκων χαμηλότερων διαστάσεων.

2.2.4 Αλγόριθμοι συσταδοποίησης.

Πολλοί αλγόριθμοι συνεργατικής διήθησης για να αυξήσουν την πυκνότητα του πίνακα των αξιολογήσεων και την επεκτασιμότητά τους χρησιμοποιούν μεθόδους συσταδοποίησης. Πληθώρα τέτοιων αλγορίθμων συναντάται στη βιβλιογραφία. Εδώ, αναφέρονται ενδεικτικά μερικοί.

Ένας από αυτούς περιγράφεται στο [60]. Σύμφωνα με αυτόν, δημιουργούνται συστάδες χρηστών με χρήση του K-means, επιλέγονται s χρήστες από τις ομάδες χρηστών που παρουσιάζουν μεγαλύτερη εγγύτητα με τον χρήστη για τον οποίο ο αλγόριθμος παράγει τις προβλέψεις, υπολογίζεται η εγγύτητα μεταξύ κάθε χρήστη και του ενεργού χρήστη και επιλέγονται οι K χρήστες με τη μεγαλύτερη εγγύτητα ώς οι κοντινότεροι γείτονες. Στη συνέχεια παράγονται οι προβλέψεις με βάση τη συμπεριφορά των K χρηστών. Οι προβλέψεις παράγονται ώς εξής,

$$pred(u,j) = \bar{r_u} + \frac{\sum_{i=1}^{K} w_{ij} \cdot sim(u,i)(R_{ij} - \bar{r_i})}{\sum_{i=1}^{K} w_{ij} \cdot sim(u,i)}$$

όπου w_{ij} είναι ένας παράγοντας στάθμισης της βαρύτητας της αξιολόγησης του αντικειμένου j από τον χρήστη i.

Στο [4] δίνεται μία προσέγγιση για ένα σύστημα συνεργατικής διήθησης βασισμένο σε συστάδες, σύμφωνα με την οποία, πρώτα επιλέγεται μια μέθοδος για να γεμίσουν οι κενές θέσεις του πίνακα των αξιολογήσεων και στη συνέχεια εφαρμόζεται συσταδοποίηση για τους χρήστες. Το σύστημα συστάσεων για να προβλέψει την αξιολόγηση του χρήστη u βρίσκει πρώτα τη συστάδα στην οποία ανήκει αυτός και παράγει την πρόβλεψη ώς εξής:

$$pred(u,j) = \bar{r_u} + \frac{\sum_{i=1}^{n} (r_{i,j} - \bar{r_i}) w_{u,i}}{\sum_{i=1}^{n} w_{u,i}}$$

όπου n είναι το πλήθος των χρηστών που ανήκουν στη συστάδα που επιλέχθηκε.

Ένας πρόσφατος αλγόριθμος που βασίζεται σε συσταδοποίηση με το μοντέλο νέφους παρουσιάζεται στο [43]. Πρώτα στις αξιολογήσεις συμπεριλαμβάνεται και μία τιμή σχετιζόμενη με το χρόνο. Στη συνέχεια τα αντικείμενα συσταδοποιούνται σύμφωνα με το μοντέλο νέφους και ο αλγόριθμος υπολογίζει την εγγύτητα του αντικειμένου p του οποίου οι γείτονες αναζητούνται και του κέντρου της συστάδας. Επιλέγει τις συστάδες που έχουν τη μεγαλύτερη εγγύτητα με το αντικείμενο p και ψάχνει σε αυτές τους κοντινότερους γείτονες του αντικειμένου. Έπειτα παράγει μία λίστα k συστάσεων επιλέγοντας τα πρώτα k αντικείμενα με τη μεγαλύτερη εγγύτητα με το αντικείμενο p.

2.2.5 Άλλοι αλγόριθμοι συνεργατικής διήθησης.

Εκτός από τις παραπάνω κατηγορίες αλγορίθμων συνεργατικής διήθησης πληθώρα άλλων μεθόδων έχουν αναπτυχθεί. Στη συνέχεια αναφέρονται οι πιό διαδεδομένες.

Αλγόριθμοι που βασίζονται σε παλινδρόμηση.

Ειδική κατηγορία αλγορίθμων βάσει αντικειμένων, όπου η συσχέτιση μεταξύ των αντικειμένων υπολογίζεται χρησιμοποιώντας τη μέθοδο της γραμμικής παλινδρόμησης. Αλγόριθμοι αυτής της κατηγορίας έχουν προταθεί στο [52].

Πιθανοτικοί αλγόριθμοι.

Οι περισσότεροι πιθανοτικοί αλγόριθμοι συνεργατικής διήθησης υπολογίζουν την πιθανότητα ένας χρήστης u να αξιολογήσει το αντικείμενο i με την τιμή r. Τα μοντέλα δικτύων Bayes είναι η πιό διαδεδομένη μέθοδος, η οποία παράγει εξαρτήσεις μεταξύ χρηστών ή αντικειμένων. Σχηματίζεται ένα ξεχωριστό δέντρο αποφάσεων για κάθε αντικείμενο ή χρήστη του συστήματος [10].

Similarity Fusion.

Ο αλγόριθμος που προτάθηκε στο [53] εξετάζει και την εγγύτητα χρηστών και την εγγύτητα αντικειμένων, με σκοπό να παράγει βελτιωμένες προβλέψεις ειδικά όταν εφαρμόζεται σε πολύ αραιούς πίνακες αξιολογήσεων προσπαθώντας να αξιοποιήσει όσο το δυνατόν περισσότερες πληροφορίες παρέχονται από τον πίνακα των αξιολογήσεων.

Οικογένεια αλγορίθμων Slope One.

Η οικογένεια αυτών των αλγορίθμων υπολογίζει τη διαφορά των αξιολογήσεων αντικειμένων που έχουν δώσει οι χρήστες που έχουν εκτιμήσει και τα δύο αντικείμενα, και τη χρησιμοποιεί για να παράγει πρόβλεψη για την εκτίμηση που θα δώσει για ένα από αυτά τα αντικείμενα ένας χρήστης που έχει εκτιμήσει μόνο το ένα από τα δύο αντικείμενα. Στο κεφάλαιο 6 αναλύεται όλη η διαδικασία.

2.3 Μετρικές αξιολόγησης αλγορίθμων συνεργατικής διήθησης.

Παρά το γεγονός ότι οι αλγόριθμοι συνεργατικής διήθησης μελετούνται εδώ και αρκετά χρόνια, δεν υπάρχει μία πλήρως αποδεκτή μετρική ικανή να εκτιμήσει ταυτόχρονα όλα τα σημαντικά κριτήρια που σχετίζονται με την απόδοση ενός συστήματος συνεργατικής διήθησης. Πολλές μετρικές χρησιμοποιούνται για την εκτίμηση των διαφορετικών πτυχών των συστημάτων συνεργατικής διήθησης, και στη συνέχεια παρουσιάζονται οι κυριότερες από αυτές.

- Μέσο απόλυτο σφάλμα και Μέσο τετραγωνικό σφάλμα (MAE, MSE).

  Οι μετρικές αυτές χρησιμοποιούνται για την εκτίμηση της ακρίβειας των προβλέψεων. Υπολογίζεται η μέση απόκλιση της τιμής των παραγόμενων από το σύστημα προβλέψεων, από την τιμή της πραγματικής αξιολόγησης που παρέχει στο σύστημα ο χρήστης.

$$MAE = \frac{\sum_{j=1}^{n_i} |r_{ij} - p_{ij}|}{n_i}$$

  Με r_{ij} συμβολίζεται η αξιολόγηση του χρήστη i για το αντικείμενο j, με p_{ij} η αντίστοιχη πρόβλεψη που παράγει το σύστημα και με n_i το πλήθος των αξιολογήσεων των οποίων υπολογίζεται η απόκλιση από τις προβλέψεις για τον χρήστη i.

$$MSE = \frac{\sum_{j=1}^{n_i} (r_{ij} - p_{ij})^2}{n_i}$$

  Συχνά χρησιμοποιείται και η τετραγωνική ρίζα του μέσου τετραγωνικού σφάλματος RMSE, που δίνει έμφαση στα μεγαλύτερα σφάλματα.

- Κάλυψη (Coverage).

  Είναι το ποσοστό των αντικειμένων του συστήματος συνεργατικής διήθησης για τα οποία μπορούν να παραχθούν προβλέψεις. Η παραγωγή προβλέψεων για κάποια αντικείμενα μπορεί να είναι αδύνατη λόγω της αραιότητας του πίνακα των εκτιμήσεων.

$$Coverage = \frac{\sum_{i=1}^{m} np_i}{\sum_{i=1}^{m} n_i}$$

  Όπου n_i είναι το πλήθος των αντικειμένων για τα οποία ο χρήστης i έχει δώσει αξιολογήσεις, np_i είναι το πλήθος των αντικειμένων για τα οποία το σύστημα συστάσεων μπόρεσε να παράγει προβλέψεις και m είναι το πλήθος των χρηστών του συστήματος.

- Precision, Recall και F1.

  Οι μετρικές Precision και Recall διαφοροποιούνται στην περίπτωση που το σύστημα παράγει λίστα των βέλτιστων N συστάσεων. Δημιουργείται ένα σύνολο (Hit set) που περιλαμβάνει τα αντικείμενα που υπάρχουν ταυτόχρονα στο test set και στο σύνολο των N βέλτιστων συστάσεων.

$$precision = \frac{\text{size of hit set}}{\text{size of top-N set}} = \frac{|\{\text{test}\} \cap \{\text{top-N}\}|}{\text{N}}$$

$$recall = \frac{\text{size of hit set}}{\text{size of test set}} = \frac{|\{\text{test}\} \cap \{\text{top-N}\}|}{|\{\text{test}\}|}$$

Η μετρική F1 συνδυάζει τις Precision και Recall.

$$F1 = \frac{2 * recall * precision}{recall + precision}$$

- Receiver Operating Characteristic Curve, ROC.

  Η μετρική αυτή παρέχει μία εναλλακτική λύση για τις μετρικές Precision και Recall [24], και χρησιμοποιείται όταν δε χρειάζεται να εκτιμηθεί η ακριβής τιμή της πρόβλεψης, αλλά μόνο το άν ένα αντικείμενο είναι αρεστό ή όχι από τον χρήστη. Επιτρέπει τη γραφική αναπαράσταση της συμπεριφοράς του συστήματος όταν αυτό διαχωρίζει αρεστά ή μη αρεστά στο χρήστη αντικείμενα. Για την εφαρμογή αυτής της μετρικής είναι απαραίτητη η τροποποίηση της κλίμακας των αξιολογήσεων σε δυαδική μορφή, ωστε να κατηγοριοποιούνται τα αντικείμενα σε χρήσιμα και μή χρήσιμα.

- Half-life utility.

  Με τη μετρική αυτή εκτιμάται η χρησιμότητα μιας ταξινομημένης λίστας προτεινόμενων αντικειμένων. Λάθη στην κορυφή της λίστας θεωρούνται πολύ πιό σημαντικά από λάθη προς το τέλος της λίστας, διότι η πιθανότητα ο χρήστης να δει τις καταχωρήσεις που βρίσκονται στην αρχή της λίστας είναι μεγαλύτερη.

  $$R_\alpha = \frac{\sum_{j=1}^{N} \max(r_{ij} - d, 0)}{2^{(j-1)/(\alpha-1)}}.$$

  Όπου d είναι μια ουδέτερη αξιολόγηση, α είναι η μέση ζωή θέασης ενός αντικειμένου, δηλαδή η θέση στη λίστα όπου το αντικείμενο έχει 50% πιθανότητα να θεαθεί από το χρήστη, και r_{ij} είναι η πραγματική αξιολόγηση του χρήστη i για το αντικείμενο j.

Υπάρχουν περιπτώσεις στις οποίες οι παραπάνω μετρικές δεν επαρκούν. Αυτό συμβαίνει όταν χρειάζεται να εκτιμηθούν διαφορετικές ιδιότητες του συστήματος συστάσεων, όπως η ικανότητά του να προτείνει αντικείμενα άγνωστα αλλά αρεστά στο χρήστη ή το πόσο γρήγορα μπορεί να είναι αποτελεσματικό στο να παράγει χρήσιμες συστάσεις στο χρήστη. Επίσης, η ικανοποίηση του χρήστη από το σύστημα συστάσεων είναι δύσκολο να μετρηθεί διότι απαιτεί από τον χρήστη να παρέχει στο σύστημα περισσότερες πληροφορίες. Έτσι, προτείνονται συνεχώς νέες μετρικές όπως η GIM και η GPIM που είναι το μέσο σφάλμα των χρήσιμων αντικειμένων και το μέσο σφάλμα των χρήσιμων προτεινόμενων αντικειμένων αντίστοιχα [17].

2.4 Προβλήματα και προκλήσεις των συστημάτων συνεργατικής διήθησης.

Ένα από τα κυριότερα προβλήματα των συστημάτων συνεργατικής διήθησης είναι η κλιμάκωση (scalability). Οι υπολογισμοί των αλγορίθμων εξαρτώνται από τον όγκο των δεδομένων, ο οποίος συνεχώς αυξάνεται. Έτσι η εύρεση αποδοτικών και γρήγορων αλγορίθμων, παρά τον όγκο δεδομένων που χρειάζεται να επεξεργασθούν, είναι αναγκαία.

Η σποραδικότητα του πίνακα των αξιολογήσεων μπορεί να αποτελέσει πρόβλημα, επηρρεάζοντας τις προβλέψεις και οδηγώντας σε παραγωγή λανθασμένων συστάσεων. Για να παράγει ένα σύστημα συστάσεων προβλέψεις για ένα χρήστη, απαιτείται να το έχει τροφοδοτήσει προηγουμένως αυτός με αξιολογήσεις. Πρέπει λοιπόν να λαμβάνονται μέτρα για τις περιπτώσεις νέων χρηστών, ώστε να μπορούν να παραχθούν προβλέψεις. Το ίδιο συμβαίνει και όταν ένα νέο αντικείμενο εισάγεται στο σύστημα και δεν μπορεί να συμμετέχει στις συστάσεις πριν να αξιολογηθεί από κάποιον χρήστη. Σε αυτές τις περιπτώσεις η εγγύτητα μεταξύ χρηστών ή αντικειμένων δεν μπορεί να υπολογισθεί λόγω έλλειψης πληροφοριών [49].

Η συνωνυμία μεταξύ αντικειμένων αποτελεί πρόβλημα των συστημάτων συστάσεων διότι δεν είναι δυνατή η εύρεση λανθανουσών σχέσεων μεταξύ των αντικειμένων, τα οποία παρά το διαφορετικό όνομα που έχουν, αναφέρονται σε παραπλήσιο περιεχόμενο. Πρέπει να λαμβάνονται μέτρα ώστε τα συστήματα συστάσεων να μπορούν να εντοπίζουν και να χειρίζονται τις λανθάνουσες σχέσεις μεταξύ των αντικειμένων, ώστε να παράγουν καλύτερες συστάσεις [15].

Η παραγωγή ποιοτικών συστάσεων είναι σημαντική διότι αυξάνει την εμπιστοσύνη των χρηστών προς το σύστημα [45]. Άν ένα σύστημα προτείνει στους χρήστες αντικείμενα που δεν παρουσιάζουν ενδιαφέρον για αυτους, τότε είναι σχεδόν σίγουρο ότι θα σταματήσουν να το χρησιμοποιούν.

Σημαντικός είναι και ο τρόπος αντιμετώπισης του συστήματος συστάσεων προς ασυνήθιστους χρήστες, διότι αυτοί μπορεί να μή μπορούν να ταυτιστούν εύκολα με τις ομάδες χρηστών, ή να επιλέγουν αντικείμενα που να μή σχετίζονται μεταξύ τους ή να είναι σπάνια. Έτσι αυτοί οι χρήστες ευνοούν την εισαγωγή αντικειμένων στις συστάσεις που δεν είναι πολύ διαδεδομένα και επομένως μπορεί να είναι άγνωστα στους άλλους χρήστες [48].

Η προστασία των προσωπικών δεδομένων είναι επίσης σημαντική. Τα συστήματα συστάσεων αξιοποιούν τις πληροφορίες που εισάγουν σε αυτά οι χρήστες, για να παράγουν εξατομικευμένες συστάσεις. Η συλλογή περισσότερων πληροφοριών οδηγεί σε καλύτερες συστάσεις, αλλά μπορεί να έχει αρνητική επίδραση στους χρήστες, οι οποίοι μπορεί να αισθάνονται ότι το σύστημα γνωρίζει πολλές πληροφορίες για τις προτιμήσεις τους. Συνεπώς, υπάρχει ανάγκη σχεδιασμού λύσεων, ώστε να προστατεύονται τα προσωπικά δεδομένα των χρηστών [41].

Στο [42] παρουσιάζονται συνοπτικά οι προκλήσεις των συστημάτων συστάσεων. Οι χρήστες επιλέγουν αντικείμενα έχοντας κάθε φορά διαφορετική διάθεση ή διαφορετικό σκοπό για τον οποίο και τα επιλέγουν. Η διαφάνεια (Transparency) ενός συστήματος μπορεί να αφορά την παροχή βοήθειας στο χρήστη, ώστε να κάνει μία καλή επιλογή ή να αποφασίσει αν μια συγκεκριμένη σύσταση εναρμονίζεται με τη διάθεσή του.

Ανάλογα με τη φύση των αντικειμένων, οι σχέσεις μεταξύ των χρηστών και των αντικειμένων ενός συστήματος συστάσεων, μπορούν να αλλάζουν με γρήγορο ή αργό ρυθμό. Η ενσωμάτωση χρονικών πλαισίων στα συστήματα συστάσεων και ο διαχωρισμός των μακροπρόθεσμων ή βραχυπρόθεσμων αλλαγών στις προτιμήσεις ενός χρήστη, αποτελούν μία ακόμα πρόκληση στην έρευνα των συστημάτων συστάσεων.

Μία ακόμα πρόκληση είναι η καθοδηγούμενη πλοήγηση (Guided Navigation), όπου το σύστημα αλληλεπιδρά με τον χρήστη. Η ερμηνεία των δράσεων του χρήστη και η άντληση πληροφορίας από αυτές, και η αξιολόγηση των συστημάτων συστάσεων, είναι επίσης τομείς που αποτελούν πρόκληση στο σχεδιασμό των συστημάτων συστάσεων.

Κεφάλαιο 3

Παράλληλη Επεξεργασία.

3.1 Βασικές έννοιες παράλληλης επεξεργασίας.

Με τον όρο Παράλληλη Επεξεργασία περιγράφεται η ταυτόχρονη χρήση πολλαπλών υπολογιστικών πόρων για την επίλυση ενός προβλήματος. Το πρόβλημα διασπάται σε διακριτά τμήματα τα οποία μπορούν να επιλυθούν ταυτόχρονα και κάθε τμήμα εκτελείται σε διαφορετικές CPU.

Οι κυριότεροι λόγοι για τους οποίους χρησιμοποιείται η παράλληλη επεξεργασία είναι η ελάττωση του χρόνου εκτέλεσης και η επίλυση μεγαλύτερων προβλημάτων. Με την παράλληλη επεξεργασία στο ίδιο χρονικό διάστημα μπορούν να επεξεργασθούν περισσότερα δεδομένα και να παραχθούν περισσότεροι υπολογισμοί. Επίσης είναι δυνατή η κατανομή δεδομένων σε πολλούς υπολογιστές και με αυτό τον τρόπο περιορίζονται τα προβλήματα μεγέθους μνήμης.

Οι παράλληλοι υπολογιστές διακρίνονται σύμφωνα με την ταξινόμηση Flynn με βάση την πολλαπλότητα εντολών και δεδομένων. Ορίζονται τέσσερις πιθανές καταστάσεις. Η πρώτη είναι η SISD (Single instruction, single data), που είναι ο τυπικός σειριακός υπολογιστής. Η δεύτερη, η SIMD (Single instruction, multiple data), στην οποία υπάρχει συνήθως μια κεντρική μονάδα ελέγχου και πολλαπλές μονάδες επεξεργασίας, που εκτελούν την ίδια εντολή σε κάθε χρονική στιγμή. Στην κατηγορία αυτή περιλαμβάνονται οι λεγόμενοι vector ή array processors, που είναι πολύ ισχυροί εξειδικευμένοι υπερυπολογιστές και δεν είναι ευρείας κυκλοφορίας. Επιπλέον, οι περισσότεροι σύγχρονοι υπολογιστές και κυρίως αυτοί που περιλαμβάνουν μονάδες επεξεργασίας γραφικών GPU, χρησιμοποιούν οδηγίες και μονάδες εκτέλεσης της κατηγορίας αυτής [6]. Η επόμενη κατάσταση είναι η MISD (Multiple instruction, single data), όπου μια ροή δεδομένων τροφοδοτεί πολλαπλές μονάδες επεξεργασίας, και τέλος, η MIMD (Multiple instruction, multiple data), στην οποία υπάρχουν πολλαπλοί επεξεργαστές συνδεδεμένοι μεταξύ τους, και κάθε μονάδα επεξεργασίας μπορεί να εκτελεί διαφορετική ροή εντολών ή διαφορετική εντολή από την ίδια ροή εντολών και να επεξεργάζεται διαφορετική ροή δεδομένων ή διαφορετικό δεδομένο από την ίδια ροή.

Στην κατηγορία MIMD διακρίνονται δύο επιμέρους υποκατηγορίες, οι οποίες διαχωρίζονται βάσει κατανομής της μνήμης σε μοιραζόμενη ή κατανεμημένη. Στην κατηγορία μοιραζόμενης μνήμης (Shared Memory) όλοι οι υπολογιστές έχουν άμεση και ισότιμη πρόσβαση σε κοινή φυσική μνήμη, και στην κατηγορία κατανεμημένης μνήμης (Distributed Memory) κάθε υπολογιστής έχει άμεση πρόσβαση σε ιδιωτική τοπική μνήμη, ενώ η πρόσβαση στη μνήμη άλλων

υπολογιστών επιτυγχάνεται συνήθως μέσω δικτύου διασύνδεσης. Οι σύγχρονοι παράλληλοι υπολογιστές ανήκουν στην κατηγορία MIMD.

Συνήθως οι παράλληλες διεργασίες χρειάζεται να ανταλλάξουν δεδομένα. Η επικοινωνία μεταξύ των διεργασιών γίνεται είτε έμμεσα, χρησιμοποιώντας κοινές θέσης μνήμης, είτε άμεσα, μεταβιβάζοντας τα δεδομένα μέσω δικτύου διασύνδεσης. Πολλές φορές απαιτείται και συγχρονισμός των παράλληλων εργασιών, όπου απαιτείται επικοινωνία με άλλες εργασίες πριν συνεχιστεί η εκτέλεση σε μία εργασία και κατά συνέπεια κάποιος επεξεργαστής αναγκάζεται να παραμένει σε αναμονή πρίν συνεχίσει την εκτέλεση των εντολών.

3.1.1 Αρχιτεκτονικές παράλληλων υπολογιστών.

3.1.1.1 Υπολογιστές Μοιραζόμενης Μνήμης.

Στους παράλληλους υπολογιστές μοιραζόμενης μνήμης, όλοι οι επεξεργαστές έχουν πρόσβαση σε ένα καθολικό χώρο φυσικών διευθύνσεων, τον οποίο μοιράζονται παρά το γεγονός ότι εργάζονται ανεξάρτητα. Αν από έναν επεξεργαστή πραγματοποιηθούν τροποποιήσεις σε περιοχές μνήμης, οι αλλαγές αυτές θα είναι ορατές και στους υπόλοιπους επεξεργαστές.

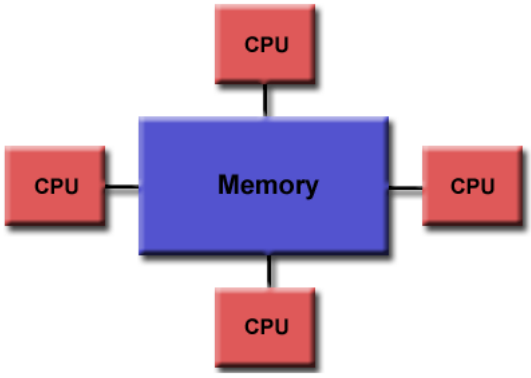
Τα πλεονεκτήματα της αρχιτεκτονικής αυτής είναι ότι ο καθολικός χώρος φυσικών διευθύνσεων επιτρέπει ένα μοντέλο προγραμματισμού φιλικό προς το χρήστη, αφού είναι παρόμοιο με αυτό του ακολουθιακού προγραμματισμού, και η κοινή χρήση των δεδομένων μεταξύ των επεξεργαστών είναι γρήγορη και ομοιόμορφη. Από την άλλη πλευρά, το κύριο μειονέκτημα είναι η έλλειψη κλιμάκωσης στον αριθμό επεξεργαστών. Προσθέτοντας επεξεργαστές αυξάνεται η κυκλοφορία στο δίαυλο της μνήμης, καθώς επίσης και η κίνηση που σχετίζεται με τη συνοχή κρυφής μνήμης. Ένα ακόμα μειονέκτημα της αρχιτεκτονικής μοιραζόμενης μνήμης είναι ότι έγκειται στην ευθύνη του προγραμματιστή ο συγχρονισμός των επεξεργαστών για τη διασφάλιση της σωστής πρόσβασης στη μνήμη.

Οι υπολογιστές μοιραζόμενης μνήμης μπορούν να διακριθούν σε δύο κατηγορίες, σύμφωνα με τον τρόπο προσπέλασης της μνήμης. Στην κατηγορία ομοιόμορφης προσπέλασης μνήμης (Uniform Memory Access, UMA) και στην κατηγορία ανομοιόμορφης προσπέλασης μνήμης (Non-Uniform Memory Access, NUMA).

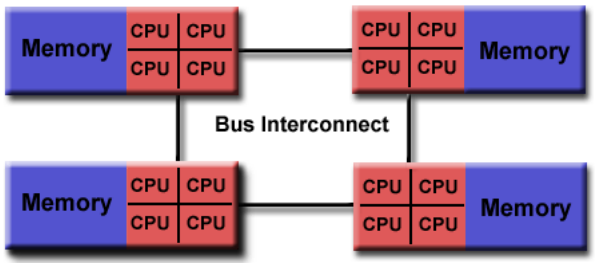
Η κατηγορία ομοιόμορφης προσπέλασης μνήμης αντιπροσωπεύεται από Συμμετρικούς Πολυ-Επεξεργαστές (Symmetric Multi-Processors, SMPs), οι οποίοι έχουν ίσους χρόνους προσπέλασης στη μοιραζόμενη μνήμη.

Στην κατηγορία ανομοιόμορφης προσπέλασης μνήμης δύο ή περισσότεροι SMPs συνδέονται με ειδικό δίαυλο ή διασυνδετικό δίκτυο, και καθένας από αυτούς έχει δυνατότητα προσπέλασης στη μνήμη του άλλου, αλλά ο χρόνος προσπέλασης της απομακρυσμένης μνήμης δεν είναι ίσος με αυτό της τοπικής του μνήμης.

Στα σχήματα 3.1 και 3.2 απεικονίζονται οι δύο κατηγορίες των υπολογιστών μοιραζόμενης μνήμης.



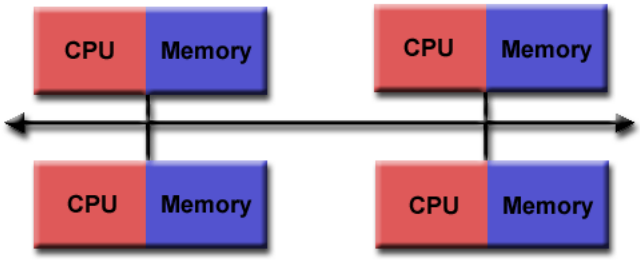
Σχήμα 3.1: Ομοιόμορφη Προσπέλαση Μοιραζόμενης Μνήμης.



Σχήμα 3.2: Ανομοιόμορφη Προσπέλαση Μοιραζόμενης Μνήμης.

3.1.1.2 Υπολογιστές Κατανεμημένης Μνήμης.

Στις αρχιτεκτονικές κατανεμημένης μνήμης, κάθε επεξεργαστής έχει τοπική μνήμη και η επικοινωνία με απομακρυσμένη μνήμη, δηλαδή την τοπική μνήμη των άλλων επεξεργαστών, επιτυγχάνεται μέσω δικτύου και συνήθως με τη ρητή συμμετοχή του απομακρυσμένου επεξεργαστή. Ο κάθε επεξεργαστής έχει τον ιδιωτικό του χώρο διευθύνσεων, έτσι δεν υπάρχει η έννοια ενός κοινού για όλους τους επεξεργαστές καθολικού χώρου διευθύνσεων. Η αρχιτεκτονική αυτή απεικονίζεται στο σχήμα 3.3.



Σχήμα 3.3: Αρχιτεκτονική Κατανεμημένης Μνήμης.

Κάθε επεξεργαστής έχει τη δική του τοπική μνήμη, οπότε, πιθανές τροποποιήσεις σε θέσεις μνήμης ενός επεξεργαστή δεν επηρεάζουν τη μνήμη άλλων επεξεργαστών. Η κρυφή μνήμη λειτουργεί ακριβώς όπως και στα συμβατικά συστήματα και δεν τίθεται ζήτημα συνοχής της. Όμως στην περίπτωση προσπέλασης δεδομένων απομακρυσμένης μνήμης, συνήθως ο προγραμματιστής πρέπει ρητά να ορίσει πότε και ποιά δεδομένα θα προσπελαστούν και να συγχρονίσει τις παράλληλες εργασίες σε διαφορετικούς επεξεργαστές.

Οι τεχνολογίες δικτύου που χρησιμοποιούνται κυμαίνονται από απλό Ethernet μέχρι ειδικά

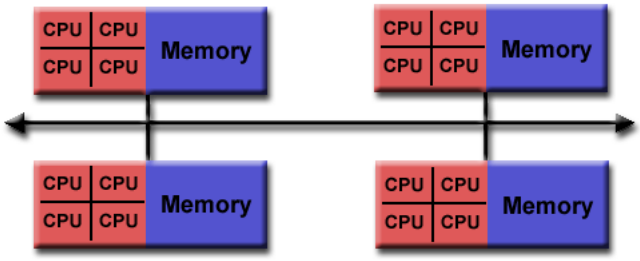
υπερταχέα δίκτυα. Επίσης η επικοινωνία μπορεί να είναι καθολική ή σημειακή, δηλαδή να υφίσταται κάποια τοπολογία, όπως γραμμή, δακτύλιος ή πλέγμα. Επιπλέον, το κόστος κατασκευής, συντήρησης και αναβάθμισης του παράλληλου συστήματος μπορεί να περιοριστεί σημαντικά, με τη χρήση συμβατικών υπολογιστών και τεχνολογιών δικτύου. Η αναβάθμιση της μνήμης επιτυγχάνεται εύκολα, με την προσθήκη περισσότερων επεξεργαστών.

Στους υπολογιστές κατανεμημένης μνήμης χρησιμοποιείται το προγραμματιστικό μοντέλο μεταβίβασης μηνυμάτων, το οποίο απαιτεί αρκετό κόπο ώστε να το κατανοήσει πλήρως ο προγραμματιστής. Άλλα μειονεκτήματα των συστημάτων κατανεμημένης μνήμης είναι ότι η απεικόνιση σύνθετων δομών δεδομένων είναι δύσκολη και μπορεί να χρειαστούν σημαντικές μετακινήσεις δεδομένων, και επιπλέον, οι χρόνοι προσπέλασης στη μνήμη δεν είναι ομοιόμορφοι.

3.1.1.3 Υβριδικοί Παράλληλοι Υπολογιστές.

Οι μεγαλύτεροι και ταχύτεροι σύγχρονοι υπολογιστές συνδυάζουν τις αρχιτεκτονικές μοιραζόμενης και κατανεμημένης μνήμης. Στην απλούστερη εκδοχή, πολυ-πύρηνοι επεξεργαστές και κάποιες φορές και μονάδες επεξεργασίας γραφικών (GPU), συνδέονται μέσω ενός γρήγορου τοπικού δικτύου.

Ο κάθε κόμβος είναι ένα σύστημα μοιραζόμενης μνήμης ομοιόμορφης προσπέλασης. Οι επεξεργαστές έχουν πρόσβαση στη μοιραζόμενη τοπική μνήμη του κόμβου, με συνοχή κρυφής μνήμης. Οι κόμβοι είναι συνδεδεμένοι μεταξύ τους με δίκτυο, έτσι ώστε ο κάθε κόμβος να έχει πρόσβαση στη μνήμη των υπολοίπων κόμβων με τη λογική της κατανεμημένης μνήμης ανομοιόμορφης προσπέλασης. Στο σχήμα 3.4 απεικονίζονται οι συνδεδεμένοι κόμβοι της υβριδικής αρχιτεκτονικής.



Σχήμα 3.4: Υβριδική Αρχιτεκτονική.

Οι υβριδικές αρχιτεκτονικές αποτελούν την προφανή λύση για την αξιοποίηση όσο το δυνατόν περισσότερων πόρων, αλλά ο προγραμματισμός ενός τέτοιου συστήματος είναι αρκετά σύνθετος απαιτώντας ταυτόχρονα συγχρονικό παραλληλισμό και ρητή επικοινωνία. Ωστόσο, οι τρέχουσες τάσεις της τεχνολογίας δείχνουν οτι το μέλλον βρίσκεται στις υβριδικές παράλληλες αρχιτεκτονικές [31, 6].

3.2 Μοντέλα Παράλληλου Προγραμματισμού.

Υπάρχουν αρκετά μοντέλα παράλληλου προγραμματισμού, τα επικρατέστερα στους σύγχρονους υπολογιστές είναι το Μοντέλο Νημάτων και το Μοντέλο Μεταβίβασης Μηνυμάτων. Άλ-

λα γνωστά μοντέλα παράλληλου προγραμματισμού είναι το Μοντέλο Μοιραζόμενης Μνήμης και το Υβριδικό Μοντέλο. Ο Παραλληλισμός Δεδομένων και ο Παραλληλισμός Λειτουργιών είναι μοντέλα παράλληλου προγραμματισμού υψηλότερου επιπέδου, τα οποία έχουν αναφορές στα προηγούμενα μοντέλα. Τα μοντέλα προγραμματισμού δε συνδέοναι άμεσα με συγκεκριμένες αρχιτεκτονικές, αλλά αποτελούν ένα εργαλείο που διευκολύνει τη σχεδίαση και συγγραφή όσο το δυνατό πιο ευέλικτου και μεταφερτού κώδικα.

Μοντέλο Μοιραζόμενης Μνήμης.

Σε αυτό το μοντέλο οι εργασίες έχουν κοινό χώρο διευθύνσεων, όπου διαβάζουν και γράφουν ασύγχρονα. Για τον έλεγχο πρόσβασης στη μοιραζόμενη μνήμη χρησιμοποιούνται μηχανισμοί διαμοιρασμού πόρων, όπως κλειδαριές. Η επικοινωνία και ο συγχρονισμός υλοποιούνται άμεσα, όμως η απλότητα των μηχανισμών συνήθως οδηγεί σε δύσκολο προγραμματισμό και ακόμη δυσκολότερη αποσφαλμάτωση και έλεγχο της εκτέλεσης.

Μοντέλο Νημάτων.

Το μοντέλο νημάτων είναι ένας τύπος προγραμματισμού μοιραζόμενης μνήμης, όπου κάθε διεργασία μπορεί να έχει πολλά συνδρομικά νήματα. Τα νήματα αυτά είναι ροές εντολών που εκτελούνται έχοντας κοινό χώρο λογικών διευθύνσεων αλλά ιδιωτική στοίβα κλήσεων και κατάσταση επεξεργαστή. Τα νήματα συνήθως υλοποιούνται ως ένα σύνολο συναρτήσεων βιβλιοθήκης ή οδηγιών προς το μεταγλωττιστή, και ο προγραμματιστής είναι υπεύθυνος για τον καθορισμό του παραλληλισμού. Οι προσπάθειες προτυποποίησης οδήγησαν σε δύο διαφορετικές αλλά σχετιζόμενες υλοποιήσεις νημάτων, τις POSIX Threads και OpenMP. Η δεύτερη περιγράφεται στην ενότητα 3.4.

Μοντέλο Μεταβίβασης Μηνυμάτων.

Στο μοντέλο μεταβίβασης μηνυμάτων υπάρχει ένα σύνολο εργασιών, που η κάθε μια έχει ιδιωτικό χώρο λογικών διευθύνσεων. Οι χώροι διευθύνσεων μπορεί να αντιστοιχούν είτε σε τοπικές φυσικές μνήμες είτε να μοιράζονται κοινή φυσική μνήμη. Για την επικοινωνία και το συγχρονισμό των εργασιών, που πραγματοποιούνται μέσω αποστολής και παραλαβής μηνυμάτων, απαιτούνται ζεύγη συνεργαζόμενων λειτουργιών που υλοποιούνται σε διαφορετικές εργασίες. Το μοντέλο μεταβίβασης μηνυμάτων συνήθως υλοποιείται ως ένα σύνολο συναρτήσεων βιβλιοθήκης. Ο προγραμματιστής καθορίζει τον παραλληλισμό με κλήσεις των συναρτήσεων μέσα στο πρόγραμμα. Η πρότυπη Διεπαφή Μεταβίβασης Μηνυμάτων (Message Passing Interface, MPI) περιγράφεται στην ενότητα 3.5.

Υβριδικό Μοντέλο.

Σε αυτό το μοντέλο συνδυάζονται δύο ή περισσότερα μοντέλα παράλληλου προγραμματισμού. Το πιό κοινό παράδειγμα είναι ο συνδυασμός του μοντέλου της μεταβίβασης μηνυμάτων (MPI) με το μοντέλο νημάτων (OpenMP). Το μοντέλο αυτό περιγράφεται στην ενότητα 3.6 Ένα ακόμα δημοφιλές παράδειγμα του υβριδικού μοντέλου είναι η χρήση MPI και GPU.

Μοντέλο Παραλληλισμού Δεδομένων.

Στο μοντέλο παραλληλισμού δεδομένων, η παράλληλη επεξεργασία εστιάζεται στην εκτέλεση λειτουργιών σε ένα σύνολο δεδομένων. Το σύνολο δεδομένων είναι μεγάλο και είναι οργανωμένο σε μια δομή όπως ένας πίνακας διαφόρων διαστάσεων ή ένας κύβος. Κάθε εργασία δουλεύει σε διαφορετικό τμήμα της δομής. Όταν το μοντέλο αυτό υλοποιείται προγραμματιστικά σε αρχιτεκτονικές μοιραζόμενης μνήμης, οι εργασίες προσπελαύνουν διαφορετικό

τμήμα της κοινής μνήμης. Ενώ όταν υλοποιείται σε αρχιτεκτονικές κατανεμημένης μνήμης, κάθε εργασία έχει στη τοπική της μνήμη διαφορετικό τμήμα των δεδομένων. Σε ένα σύστημα πολυεπεξεργαστών στο οποίο εκτελείται ένα σύνολο οδηγιών SIMD, ο παραλληλισμός δεδομένων επιτυγχάνεται όταν κάθε επεξεργαστής εκτελεί τις ίδιες λειτουργίες στα διαφορετικά μέρη των κατανεμημένων δεδομένων [54].

Μοντέλο Παραλληλισμού Λειτουργιών.

Στο μοντέλο αυτό, ο προγραμματιστής οφείλει να χωρίσει την εργασία σε επιμέρους λειτουργίες οι οποίες αντιστοιχούνται αυτομάτως στα φυσικά νήματα που προγραμματίζονται από το σύστημα [2]. Ο παραλληλισμός λειτουργιών μπορεί να εφαρμοσθεί με κλήσεις σε κατάλληλες βιβλιοθήκες. Σε ένα σύστημα πολυεπεξεργαστών, ο παραλληλισμός λειτουργιών επιτυγχάνεται όταν κάθε επεξεργαστής εκτελεί διαφορετικές λειτουργίες στα ίδια ή και σε διαφορετικά δεδομένα [57].

3.3 Ανάπτυξη Παράλληλων Εφαρμογών.

Ο σχεδιασμός και η ανάπτυξη παράλληλων προγραμμάτων είναι σύνθετη διαδικασία. Συνήθως ο προγραμματιστής είναι υπεύθυνος για να αντιληφθεί, να σχεδιάσει και να υλοποιήσει τον παραλληλισμό. Υπάρχουν διάφορα εργαλεία υποβοήθησης του προγραμματιστή για την αυτόματη μετατροπή ορισμένου τύπου ακολουθιακών προγραμμάτων σε παράλληλα, αλλά η χρήση τους μπορεί να δημιουργήσει πολλά προβλήματα, όπως την αδυναμία εντοπισμού των αιτιών εσφαλμένων αποτελεσμάτων, και τη δύσκολη τροποποίηση και συντήρηση του κώδικα. Στη συνέχεια περιγράφονται τα στάδια του σχεδιασμού παράλληλων προγραμμάτων.

Πρώτη και βασική μέριμνα είναι η κατανόηση του προβλήματος και όταν υπάρχει ακολουθιακός κώδικας, η κατανόηση αυτού. Στη συνέχεια πρέπει να προσδιοριστεί αν είναι εφικτή η παραλληλοποίηση του προβλήματος ή όχι. Η εύρεση των πιό χρονοβόρων τμημάτων του προγράμματος και η μελέτη ύπαρξης σημείων συμφόρησης σε αυτά, καθώς και η μελέτη ύπαρξης εμποδίων παραλληλισμού, όπως η εξάρτηση δεδομένων, μπορούν να χρησιμοποιηθούν ως κριτήρια για τον αν ένας ακολουθιακός κώδικας είναι κατάλληλος για παραλληλοποίηση.

Όταν ένα πρόβλημα κριθεί κατάλληλο για παραλληλοποίηση, το επόμενο βήμα είναι ο επιμερισμός των δεδομένων και των λειτουργιών. Οι δομές δεδομένων του προβλήματος χωρίζονται σε τμήματα, με οποιοδήποτε τρόπο θεωρηθεί καταλληλότερος, και κάθε παράλληλη διεργασία επεξεργάζεται ένα ή περισσότερα τμήματα. Ο επιμερισμός λειτουργιών αφορά τα λογικά τμήματα του προγράμματος. Μια εργασία αποτελείται από ένα ή περισσότερα υποπρογράμματα, και πολλές φορές είναι απαραίτητη η επικοινωνία μεταξύ των εργασιών. Συχνά απαιτείται ο συνδυασμός των δύο μεθόδων επιμερισμού.

Η επικοινωνία που χρειάζεται να εισαχθεί σε ένα παράλληλο πρόγραμμα εξαρτάται από τον τύπο του προβλήματος. Αρκετά προβλήματα χρειάζονται ελάχιστο διαμοιρασμό δεδομένων ή και ελάχιστη επικοινωνία μεταξύ των εργασιών για να παραλληλοποιηθούν. Πιό σύνθετοι τύποι προβλημάτων, που περιέχουν κάποια μορφή εξάρτησης δεδομένων ή συγχρονισμού, απαιτούν περισσότερη επικοινωνία. Υπάρχουν αρκετοί παράγοντες που πρέπει να ληφθούν υπ΄ όψη κατά το σχεδιασμό της επικοινωνίας μεταξύ εργασιών, όπως το κόστος επικοινωνίας, η ορατότητα της επικοινωνίας, δηλαδή το άν θα παρεμβαίνει ο προγραμματιστής στο πρόγραμμα για να ρυθμίσει την επικοινωνία ή όχι, το εύρος και η απόδοση της επικοινωνίας και ο

συγχρονισμός της.

Ο συγχρονισμός αφορά συνήθως τα συστήματα κατανεμημένης μνήμης και αναφέρεται στις εργασίες που εκτελούν την επικοινωνία. Ο συγχρονισμός των παράλληλων εργασιών του προγράμματος μπορεί να επιτευχθεί με τη χρήση φραγμάτων. Όταν κάποια εργασία φτάσει σε ένα φράγμα, αναστέλλεται η λειτουργία της μέχρι όλες οι εργασίες να φτάσουν το φράγμα. Επίσης, μπορούν να εφαρμοστούν κλειδώματα για την προστασία της πρόσβασης των εργασιών σε μοιραζόμενα δεδομένα.

Η ορθότητα εκτέλεσης των εφαρμογών στον παράλληλο προγραμματισμό επηρεάζεται από τις εξαρτήσεις των δεδομένων. Μπορεί να υπάρχει σειριακή εξάρτηση μεταξύ των εντολών ενός προγράμματος, όταν η σειρά εκτέλεσης των εντολών επηρεάζει το αποτέλεσμα του προγράμματος. Επίσης, από την πολλαπλή χρήση των ίδιων μεταβλητών από διαφορετικές εργασίες, μπορεί να προκύψει εξάρτηση δεδομένων. Η εξάρτηση δεδομένων μπορεί να οδηγήσει σε αδιέξοδα και σε συνθήκες ανταγωνισμού (Data Race Conditions), τα οποία είναι τα βασικά προβλήματα των συστημάτων μοιραζόμενης μνήμης. Η εξάρτηση δεδομένων αντιμετωπίζεται με το συγχρονισμό και την επικοινωνία ή το κλείδωμα των δεδομένων.

Η εξισορρόπηση φορτίου έχει σαν στόχο την όσο το δυνατό ισομερή κατανομή υπολογιστικού φόρτου μεταξύ των εργασιών. Έτσι, όλες οι εργασίες εκτελούν χρήσιμους υπολογισμούς για όσο το δυνατόν μεγαλύτερη διάρκεια, και ελαχιστοποιείται ο χρόνος κατά τον οποίο παραμένουν άεργες. Η εξισορρόπηση φορτίου είναι σημαντική διότι βελτιώνει την απόδοση των παράλληλων εφαρμογών, και μπορεί να επιτευχθεί είτε με στατική, είτε με δυναμική κατανομή φορτίου.

Ένας ακόμα παράγοντας που απαιτεί ρυθμίσεις κατά την ανάπτυξη παράλληλων εφαρμογών είναι ο χειρισμός των λειτουργιών εισόδου και εξόδου I/O. Οι λειτουργίες I/O προκαλούν καθυστερήσεις, οπότε είναι θεμιτό να μειώνονται όσο το δυνατό περισσότερο και να περιορίζονται σε συγκεκριμένα, σειριακά τμήματα του προγράμματος.

3.4 OpenMP.

Το OpenMP (Open Specifications for Multi Processing) είναι μια Διεπαφή Προγραμματισμού Εφαρμογών, που αποτελείται από οδηγίες προς το μεταγλωτιστή, βιβλιοθήκη συναρτήσεων και μεταβλητές περιβάλλοντος, με σκοπό την επίτευξη παραλληλισμού από το μεταγλωτιστή στο πρόγραμμα.

Είναι ένα πρότυπο παράλληλου προγραμματισμού, που παρέχει τη δυνατότητα ανάπτυξης παράλληλων προγραμμάτων για συστήματα μοιραζόμενης μνήμης [11]. Το OpenMP μπορεί να χρησιμοποιηθεί στις γλώσσες C/C++ και FORTRAN. Το μοντέλο προγραμματισμού του OpenMP είναι βασισμένο στο πολυνηματικό μοντέλο παραλληλισμού. Οι εφαρμογές ξεκινούν με ένα νήμα (master thread) και στα σημεία που ο προγραμματιστής έχει ορίσει να εκτελεστούν παράλληλα (parallel regions) δημιουργούνται νέα νήματα. Μετά την παράλληλη περιοχή συνεχίζεται η εκτέλεση του κώδικα με ένα νήμα, το master thread. Ο προγραμματιστής είναι υπεύθυνος για τον ορισμό και το χειρισμό παράλληλων περιοχών στο πρόγραμμα, καθώς και για το συγχρονισμό των νημάτων, όπου αυτός είναι απαραίτητος.

Για να αποφεύγεται η συνεχής προσπέλαση στη μνήμη, κάθε νήμα μπορεί να έχει μια προσωρινή άποψη (temporary view) της μνήμης και ιδιωτική μνήμη (thread-private memory)

στην οποία έχει πρόσβαση μόνο αυτό. Σε μια οδηγία παραλληλισμού μπορεί να ορίζονται μοιραζόμενες και ιδιωτικές μεταβλητές. Κάθε αναφορά σε μοιραζόμενη μεταβλητή είναι μία αναφορά στην ίδια την μεταβλητή, ενώ μια αναφορά σε ιδιωτική μεταβλητή είναι μια αναφορά σε ένα τοπικό αντίγραφο στην ιδιωτική μνήμη του νήματος. Για την προσπέλαση των μοιραζόμενων μεταβλητών από διαφορετικά νήματα απαιτείται συγχρονισμός. Σε περίπτωση εμφωλευμένων οδηγιών παραλληλισμού μία μεταβλητή που είναι ιδιωτική στην εξωτερική παράλληλη περιοχή, μπορεί να είναι μοιραζόμενη στην εσωτερική παράλληλη περιοχή, εκτός αν έχει οριστεί στην εσωτερική οδηγία ως ιδιωτική.

Στον πίνακα 3.1 περιγράφονται οι οδηγίες του OpenMP. Οι οδηγίες for, sections και single είναι οδηγίες διαμοιρασμού εργασίας και οι οδηγίες Barrier, master, critical, atomic, flush, ordered και threadprivate είναι οδηγίες συγχρονισμού.

Σε συνδυασμό με τις οδηγίες διαμοιρασμού εργασίας χρησιμοποιούνται οι φράσεις εμβέλειας δηλώσεων δεδομένων, οι οποίες καθορίζουν την εμβέλεια και το διαμοιρασμό των ματαβλητών. Οι φράσεις αυτές έχουν ισχύ μόνο στις οδηγίες που χρησιμοποιούνται και ορίζουν αν οι μεταβλητές θα είναι ορατές σε όλα τα νήματα ή σε συγκεκριμένο και άν θα μεταφέρουν τις τιμές τους από και πρός την παράλληλη περιοχή και με ποιό τρόπο [8, 31].

Στον πίνακα 3.2 περιγράφεται η λειτουργία αυτών των φράσεων.

ΟΔΗΓΙΑ	**ΠΕΡΙΓΡΑΦΗ**
parallel	Ορίζει την περιοχή του κώδικα που θα εκτελεστεί παράλληλα.
for	Διαμοιράζει τις επαναλήψεις ενός βρόγχου for στα νήματα της τρέχουσας παράλληλης περιοχής.
sections	Διαμοιράζει την εργασία σε διακριτά blocks. Κάθε block εκτελείται από ένα νήμα.
single	Σειριοποιεί ένα τμήμα του κώδικα ώστε να εκτελεστεί υποχρεωτικά από ένα νήμα.
barrier	Συγχρονίζει όλα τα νήματα της ομάδας.
master	Ορίζει ένα τμήμα κώδικα το οποίο θα εκτελεστεί από το master thread μόνο.
critical	Καθορίζει μια περιοχή η οποία πρέπει να εκτελεστεί μόνο από ένα νήμα κάθε φορά.
atomic	Καθορίζει ότι η συγκεκριμένη θέση μνήμης πρέπει να ενημερώνεται ατομικά (atomic action), μην επιτρέποντας πολλά νήματα να ενημερώσουν ταυτόχρονα τη συγκεκριμένη θέση μνήμης.
flush	Καθορίζει ένα σημείο συγχρονισμού στο οποίο η υλοποίηση του κώδικα πρέπει να παρέχει ένα συνεπές στιγμιότυπο της μνήμης και η τρέχουσα τιμή μιας μοιραζόμενης μεταβλητής εγγράφεται άμεσα στη μνήμη από τη cache (write back).
ordered	Καθορίζει ότι οι επαναλήψεις του εσωκλειώμενου βρόγχου θα εκτελεστούν με τη σειρά όπως θα εκτελούνταν σειριακά.
threadprivate	Καθορίζει ότι καθολικά αντικείμενα (ή μεταβλητές) μπορούν να γίνουν προσωρινά ιδιωτικά για κάποιο νήμα.

Πίνακας 3.1: Οδηγίες του OpenMP.

ΦΡΑΣΗ	ΠΕΡΙΓΡΑΦΗ
private	Ορίζει μεταβλητές ιδιωτικές για κάθε νήμα.
shared	Ορίζει μεταβλητές ως διαμοιραζόμενες σε όλα τα νήματα. Αυτό σημαίνει ότι αν ένα νήμα αλλάξει την τιμή μιας τέτοιας μεταβλητής, η αλλαγή θα είναι ορατή σε όλα τα νήματα.
firstprivate	Ορίζει μεταβλητές ιδιωτικές για κάθε νήμα και τα αντίγραφα των μεταβλητών αρχικοποιούνται στη δημιουργία των νημάτων.
lastprivate	Διατηρεί, κατά την έξοδο απο τη παράλληλη περιοχή, την τελυταία τιμή που απέκτησε κάποιο από τα αντίγραφά της.
default	Ορίζει ένα προκαθορισμένο τύπο πρόσβασης για όλες τις μεταβλητές σε μια παράλληλη περιοχή.
reduction	Εκτελεί μία πράξη αναγωγής σε κάποιες μοιραζόμενες μεταβλητές.
copyin	Χρησιμοποιείται για να δώσει αρχική τιμή σε threadprivate μεταβλητές.
copyprivate	Χρησιμοποιείται για την διανομή τιμών μεταβλητής από ένα νήμα σε όλα τα αντίγραφα της μεταβλητής στα υπόλοιπα νήματα.

Πίνακας 3.2: Φράσεις Εμβέλειας Δηλώσεων Δεδομένων του OpenMP.

3.5 Η Διεπαφή Μεταβίβασης Μηνυμάτων MPI.

Η Διεπαφή Μεταβίβασης Μηνυμάτων (Message Passing Interface - MPI) είναι ένα πρότυπο σύστημα μεταβίβασης μηνυμάτων στατικής σύνδεσης βασισμένη στην προδιαγραφή του MPI Forum. Το πρότυπο MPI ορίζει το συντακτικό και τη σημασιολογία ρουτίνων χρήσιμων σε ένα ευρύ φάσμα χρηστών για τη συγγραφή παράλληλων προγραμμάτων που βασίζονται στη μεταβίβαση μηνυμάτων. Οι προδιαγραφές της διεπαφής έχουν ήδη οριστεί για τις γλώσσες προγραμματισμού C/C++ και Fortran.

Είναι ένα πρωτόκολλο επικοινωνίας ανεξάρτητο από γλώσσες που χρησιμοποιείται για τον προγραμματισμό παράλληλων υπολογιστών. Υποστηρίζει σημειακή και συλλογική επικοινωνία και περιλαμβάνει ρουτίνες για τη διαχείριση περιβάλλοντος, για διαχείριση ομάδων διεργασιών και εικονικών τοπολογιών. Οι στόχοι του MPI είναι υψηλή απόδοση, επεκτασιμότητα και φορητότητα.

Παραμένει το κυρίαρχο μοντέλο που χρησιμοποιείται σε υπολογιστές υψηλών επιδόσεων, και είναι πρότυπο για την επικοινωνία μεταξύ των διεργασιών που εκτελούν ένα παράλληλο πρόγραμμα σε ένα σύστημα κατανεμημένης μνήμης. Τα προγράμματα που χρησιμοποιούν τη διεπαφή MPI εκτελούνται και σε συστήματα μοιραζόμενης μνήμης.

Τα προγράμματα που χρησιμοποιούν MPI λειτουργούν με διεργασίες. Συνήθως οι διεργασίες αντιστοιχίζονται με τους επεξεργαστές. Για επίτευξη μεγαλύτερων επιδόσεων σε κάθε CPU ή σε κάθε πυρήνα σε πολυπύρηνους υπολογιστές θα ανατεθεί μόνο μία διεργασία. Αυτή η ανάθεση γίνεται στην αρχή του MPI προγράμματος όταν καλείται η εντολή mpiexec (ή mpirun). Η διεπαφή MPI έχει ώς στόχο να παρέχει συγχρονισμό και λειτουργικότητα στην επικοινωνία μεταξύ ενός συνόλου διεργασιών [31, 55, 7] .

Στους πίνακες 3.3, 3.4 και 3.5 περιγράφονται συνοπτικά μερικές από τις κυριότερες ρουτίνες διαχείρησης περιβάλλοντος, σημειακής και συλλογικής επικοινωνίας του MPI.

ΡΟΥΤΙΝΑ	**ΠΕΡΙΓΡΑΦΗ**
MPI_Init	Αρχικοποιεί το περιβάλλον εκτέλεσης του MPI.
MPI_Comm_size	Προσδιορίζει τον αριθμό των διεργασιών στο group το οποίο σχετίζεται με έναν communicator.
MPI_Comm_rank	Προσδιορίζει το αναγνωριστικό της καλούμενης διεργασίας μέσα στον communicator.
MPI_Abort	Τερματίζει όλες τις διεργασίες του MPI που σχετίζονται με τον communicator.
MPI_Get_processor_name	Επιστρέφει το όνομα της διεργασίας.
MPI_Initialized	Δείχνει αν έχει κληθεί η MPI_Init, επιστρέφει λογικό true ή false.
MPI_Wtime	Επιστρέφει τον χρόνο που έχει περάσει σε δευτερόλεπτα.
MPI_Finalize	Τερματίζει το περιβάλλον εκτέλεσης του MPI.

Πίνακας 3.3: Ρουτίνες διαχείρησης περιβάλλοντος του MPI.

ΡΟΥΤΙΝΑ	**ΠΕΡΙΓΡΑΦΗ**
MPI_Send	Βασική λειτουργία αποστολής μηνύματος.
MPI_Recv	Βασική λειτουργία παραλαβής μηνύματος.
MPI_Ssend	Σύγχρονη ανασταλτική αποστολή: στέλνει ένα μήνυμα μέχρι ο buffer της εφαρμογής να αδειάσει και να είναι έτοιμος να ξαναχρησιμοποιηθεί και μέχρι η διεργασία παραλαβής να επιβεβαιώσει την παραλαβή του μηνύματος.
MPI_Sendrecv	Αποστολή ενός μηνύματος και παραλαβή ενός άλλου.
MPI_Wait	Προκαλεί αναστολή μέχρι να ολοκληρωθεί μια μη-ανασταλτική λειτουργία.
MPI_Probe	Διεξάγει έναν έλεγχο αναστολής για ένα μήνυμα.

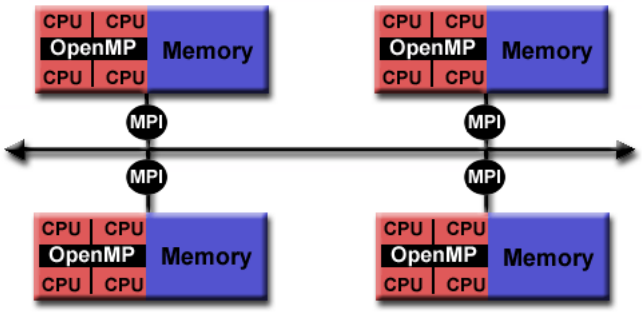
Πίνακας 3.4: Ρουτίνες σημειακής επικοινωνίας του MPI.

ΡΟΥΤΙΝΑ	**ΠΕΡΙΓΡΑΦΗ**
MPI_Barrier	Δημιουργεί ένα όριο συγχρονισμού σε μια ομάδα.
MPI_Bcast	Διαδίδει ένα μήνυμα από την διεργασία-πηγή στις υπόλοιπες διεργασίες της ομάδας.
MPI_Scatter	Διανέμει τμήματα ενός μηνύματος στις υπόλοιπες διεργασίες της ομάδας
MPI_Gather	Συλλέγει διαφορετικά μηνύματα από κάθε διεργασία σε μία διεργασία-προορισμό.
MPI_Allgather	Όλες οι διεργασίες της ομάδας συλλέγουν τα μυνήματα όλων των διεργασιών
MPI_Reduce	Μια διεργασία εφαρμόζει μία συγκεκριμένη πράξη στα βαθμωτά δεδομένα που συλλέγει από όλες τις διεργασίες και τοποθετεί το αποτέλεσμα σε μία μεταβλητη.

Πίνακας 3.5: Ρουτίνες συλλογικής επικοινωνίας του MPI.

3.6 Υβριδικό μοντέλο με χρήση MPI και OpenMP.

Το υβριδικό μοντέλο παράλληλου προγραμματισμού με χρήση MPI και OpenMP χρησιμοποιεί MPI για τη μεταφορά μηνυμάτων μεταξύ κόμβων και OpenMP σε κάθε συμμετρικό πολυ-επεξεργαστή (SMP) των κόμβων (Σχήμα 3.5). Τα κυριότερα κίνητρα για την εφαρμογή ενός υβριδικού μοντέλου είναι η ελαχιστοποίηση παραγόντων όπως το κόστος επικοινωνίας, υπολογισμών και συγχρονισμού, και η εξισορρόπηση φορτίου μεταξύ των κόμβων του συστήματος.



Σχήμα 3.5: Υβριδικό Μοντέλο.

Οι μεγαλύτεροι και ταχύτεροι υπολογιστές χρησιμοποιούν αρχιτεκτονική που περιλαμβάνει και μοιραζόμενη και κατανεμημένη μνήμη. Παρά το γεγονός ότι το υβριδικό μοντέλο προγραμματισμού εκμεταλλεύεται καί τους δύο τύπους μνήμης του συστήματος, δεν είναι πάντα ευεργετική η εφαρμογή του γιατί εξαρτάται από τη δομή του προβλήματος, τους υπολογισμούς και την επικοινωνία που απαιτείται. Ωστόσο, αν η εφαρμογή υβριδικού μοντέλου προγραμματισμού σε ένα πρόβλημα είναι επιτυχημένη, μπορεί να αποδώσει σημαντική επιτάχυνση (speedup) και κλιμάκωση (scalability).

3.7 Μετρικές αξιολόγησης παράλληλων προγραμμάτων.

Στην παρούσα ενότητα περιγράφονται συνοπτικά οι μετρικές που χρησιμοποιούνται για την αξιολόγηση της απόδοσης των παράλληλων εφαρμογών [31, 22, 9]. Με T_1 συμβολίζεται ο σειριακός χρόνος εκτέλεσης, με T_p ο παράλληλος χρόνος εκτέλεσης, και με p το πλήθος των επεξεργαστών που χρησιμοποιούνται. T_c είναι το κόστος εκτέλεσης κάθε λειτουργίας, T_o είναι η συνολική επιβάρυνση, δηλαδή ο χρόνος εκτέλεσης λειτουργιών που δεν εφαρμόζονται στον ακολουθιακό αλγόριθμο και W το μέγεθος του προβλήματος.

Επιτάχυνση (Speedup)

Επιτάχυνση ενός παραλληλοποιημένου προγράμματος είναι ο λόγος του χρόνου της ακολουθιακής εκτέλεσης προς το χρόνο της παράλληλης εκτελεσης, για εκτέλεση μιας παράλληλης εφαρμογής με συγκεκριμένο αριθμό επεξεργαστών και ίδιο αλγόριθμο και δεδομένα όπως η ακολουθιακή. Όταν η επιτάχυνση αυξάνεται ανάλογα με τον αριθμό των χρησιμοποιούμενων επεξεργαστών ονομάζεται γραμμική.

$$S = \frac{T_1}{T_p}$$

Το πλήθος των επεξεργαστών που χρησιμοποιούνται αποτελεί ένα άνω όριο στην επιτάχυνση που μπορεί να επιτευχθεί σε ένα παράλληλο σύστημα. Η επιτάχυνση για τη χρήση ενός επεξεργαστή ορίζεται ίση με ένα. Όταν χρησιμοποιούνται περισσότεροι επεξεργαστές επιτυγχάνεται επιτάχυνση που συνήθως είναι μικρότερη από το πλήθος των επεξεργαστών. Συχνά χρησιμοποιείται ο νόμος του Amdahl για τον υπολογισμό της μέγιστης αναμενόμενης βελτίωσης της επιτάχυνσης ενός προγράμματος από τον παραλληλισμό διαφόρων τμημάτων του. Σύμφωνα με το νόμο αυτό, άν P είναι το ποσοστό του προγράμματος που παραλληλοποιείται και $1 - P$ είναι αυτό που παραμένει ακολουθιακό, τότε η μέγιστη επιτάχυνση που μπορεί να επιτευχθεί με τη χρήση N επεξεργαστών είναι:

$$S_N = \frac{N}{P} + \frac{1}{1-P}$$

Όσο το $N \to \infty$ η μέγιστη επιτάχυνση $S_\infty \to \frac{1}{1-P}$.

Κλιμάκωση (Scalability)

Είναι η δυνατότητα μιας παράλληλης εφαρμογής να επιδείξει γραμμική επιτάχυνση με τη πρόσθεση περισσότερων επεξεργαστών ή με την αύξηση του μεγέθους του επιλυόμενου προβλήματος. Η μετρική αυτή σχετίζεται με την ισοαπόδοση που περιγράφεται παρακάτω.

Αποδοτικότητα (Efficiency)

Αποδοτικότητα ενός παράλληλου συστήματος ορίζεται ώς ο λόγος της επιτάχυνσης που επιτυγχάνεται προς το πλήθος των επεξεργαστών που χρησιμοποιούνται.

$$E = \frac{S}{p}$$

Επιβάρυνση (Overhead)

Είναι ο χρόνος που δαπανάται στο συντονισμό των παράλληλων εργασιών σε αντίθεση με το χρόνο ωφέλιμου υπολογισμού. Εξαρτάται από τη δημιουργία και τη διανομή εργασιών, από το συγχρονισμό και την επικοινωνία κ.τ.λ

Κοκκότητα (Granularity)

Συνήθως η επικοινωνία είναι πολύ πιο δαπανηρή από τον υπολογισμό, κυρίως στα συστήματα κατανεμημένης μνήμης. Η κοκκότητα είναι ο λόγος του χρόνου υπολογισμών προς το χρόνο επικοινωνίας. Η κοκκότητα χαρακτηρίζεται αδρομερής (coarse) όταν ο υπολογιστικός φόρτος μιας παράλληλης εργασίας είναι πολύ μεγαλύτερος από την επικοινωνία της, και λεπτομερής (fine) όταν ο υπολογιστικός φόρτος μια παράλληλης εργασίας είναι συγκρίσιμος με την επικοινωνία της.

Κόστος και Ωφελιμότητα (Cost and Utilization)

Κόστος είναι ο χρόνος εκτέλεσης επί τον αριθμό επεξεργαστών που χρησιμοποιούνται. Ωφελιμότητα είναι ο ποσοστιαίος λόγος του ακολουθιακού προς το παράλληλο κόστος. Ωφελιμότητα 100% σημαίνει μηδενική επιβάρυνση, δηλαδή ιδανική παραλληλοποίηση.

Ισοαπόδοση (Isoefficiency)

Είναι το ποσοστό αύξησης του προβλήματος που απαιτείται, ώστε αυξάνοντας το πλήθος p των επεξεργαστών που χρησιμοποιούνται, η Αποδοτικότητα E να παραμένει σταθερή.

$$W = KT_o$$

Όπου K είναι μία σταθερά που εξαρτάται από την αποδοτικότητα.

$$K = \frac{E}{T_c(1-E)}$$

Ισοαπόδοση σε ετερογενή συστήματα (H-Isoefficiency)

Η ισοαπόδοση (Isoefficiency) εξαρτάται μόνο από το πλήθος των επεξεργαστών, θεωρώντας ότι οι επεξεργαστές έχουν όλοι την ίδια υπολογιστική ισχύ. Στην περίπτωση όμως των ετερογενών συστημάτων δε συμβαίνει αυτό. Η H-Isoefficiency ορίζεται ώς,

$$W = KT_oP_T$$

Η υπολογιστική ισχύς ενός ετερογενούς συστήματος, P_T ορίζεται ώς το άθροισμα της υπολογιστικής ισχύος κάθε επεξεργαστή P_i.

$$P_T = \sum_{i=1}^{p} P_i$$

Το K είναι και σε αυτή την περίπτωση μία σταθερά που εξαρτάται από την Αποδοτικότητα του συστήματος.

3.8 Παράδειγμα παραλληλοποίησης προγράμματος υπολογισμού του π.

3.8.1 Σειριακός υπολογισμός του π.

Στο παρακάτω παράδειγμα υπολογίζεται το π με χρήση αριθμητικής ολοκλήρωσης. Υπολογίζεται το τόξο εφαπτομένης $4\arctan(1)$, όπου $\arctan(1) = \int_0^1 \frac{1}{1+x^2}\,dx$. Στη συνέχεια παρατίθεται ο ακολουθιακός κώδικας (πλαίσιο 3.1).

```
# include <stdio.h>
# include <math.h>

double f( double );
double f( double a )
{
    return (4.0 / (1.0 + a*a));
}

int main( int argc, char *argv[])
{
```

```
    int done = 0, n, i;
    double PI25DT = 3.14159265358979323846264 3;
    double pi, h, sum, x;

    n = 0;
    while (!done)
    {
        printf("Enter␣the␣number␣of␣intervals:␣(0␣quits)␣");
        scanf("%d",&n);

    if (n == 0)
            done = 1;
        else
        {
            pi = 0.0;
            h  = 1.0 / (double) n;
                sum = 0.0;
                for (i = 0; i <= n; i++)
               {
                    x = h * ((double)i - 0.5);
                    sum += f(x);
                }
                   pi += h * sum;
        }
        printf("pi␣is␣approximately␣%.16f,␣Error␣is␣%.16f\n", pi, fabs(
           pi - PI25DT));

    }
    return 0;
}
```

Listing 3.1: Ακολουθιακός κώδικας υπολογισμού του π .

Με n συμβολίζεται ο αριθμός των διαστημάτων στα οποία χωρίζεται το διάστημα [0,1]. Ο αριθμός αυτός παρέχεται στο πρόγραμμα από τον χρήστη. Στην περίπτωση που ο χρήστης δώσει την τιμή 0, η εκτέλεση του προγράμματος τερματίζεται. Η μεταβλητή h είναι το πλάτος κάθε διαστήματος. Ο αλγόριθμος που χρησιμοποιείται στο παραπάνω πρόγραμμα εκτελεί n επαναλήψεις για τον υπολογισμό του π, δηλαδή όσες και το πλήθος των διαστημάτων που επέλεξε ο χρήστης. Έτσι, σε περίπτωση που ο χρήστης δώσει πολύ μεγάλη τιμή για το n, το χρονοβόρο τμήμα του προγράμματος θα είναι ο υπολογισμός που πραγματοποιείται μέσα στη for.

3.8.2 Παράλληλος υπολογισμός του π με OpenMP.

Στη συνέχεια παρατίθεται στο πλαίσιο 3.2 η υλοποίηση του υπολογισμού του π με χρήση του OpenMP.

```
1 # include <stdio.h>
2 # include <math.h>
3 # include <omp.h>
4
5 int main ( int argc, char *argv[] );
6 double pi_est_omp ( int n );
```

```
7
8  int main ( int argc, char *argv[] )
9  {
10   int n;
11
12   printf ( "␣␣Number␣of␣processors␣available␣=␣%d\n", omp_get_num_procs
         ( ) );
13   printf ( "␣␣Number␣of␣threads␣=␣␣␣␣␣␣␣␣␣␣␣␣␣␣%d\n",
         omp_get_max_threads ( ) );
14   printf ( "␣␣Give␣number␣of␣points␣=␣␣␣␣␣␣␣␣␣␣");
15   scanf  ( "%d", &n);
16   printf ( "␣␣N␣␣␣␣␣␣␣␣␣␣MODE␣␣␣␣␣␣␣␣␣␣␣␣ESTIMATE␣␣␣␣␣␣␣ERROR␣␣␣␣\n");
17
18 pi_est_omp ( n );
19
20   return 0;
21 }
22
23 double pi_est_omp ( int n )
24 {
25   double pi = 3.141592653589793;
26   double h;
27   double estimate;
28   int i;
29   double sum2,sum1;
30   double x;
31
32   h = 1.0 / ( double ) ( n );
33   sum2 = 0.0;
34
35 # pragma omp parallel \
36   shared ( h, n ) \
37   private ( i, x,sum1 ) \
38   reduction ( +: sum2 )
39 {
40 sum1=0.0;
41 # pragma omp for
42   for ( i = 1; i <= n; i++ )
43   {
44     x = h * ( double ) (  i - 0.5 );
45     sum1 = sum1 + 1.0 / ( 1.0 + x * x );
46   }
47  sum2=sum2+sum1;
48 }
49
50   estimate = 4.0 * sum2 / ( double ) ( n );
51   printf ( "%11d␣␣␣OMP␣%14f␣␣␣%14g␣␣␣\n", n, estimate, fabs(estimate-pi) );
52
53   return 0;
54 }
```

Listing 3.2: Παράλληλος κώδικας υπολογισμού του π με χρήση OpenMP .

Στον ακολουθιακό κώδικα χρησιμοποιήθηκαν οι μεταβλητές, $n, i, PI25DT, pi, h, sum$ και x. Στον παράλληλο χρησιμοποιήθηκαν οι $n, i, pi, estimate, h, sum1, sum2$ και x. Οι μεταβλητές $PI25DT$ και pi του ακολουθιακού και του παράλληλου κώδικα αντίστοιχα, αναφέρονται στη γνωστή εκ των πρωτέρων τιμή του π, που χρησιμοποιείται για τον υπολογισμό του

σφάλματος της προσέγγισής του από το πρόγραμμα. Η μεταβλητή pi του ακολουθιακού και η μεταβλητή $estimate$ του παράλληλου προγράμματος αναφέρονται στην τιμή που υπολογίζει το πρόγραμμα για το π.

Στη συνέχεια περιγράφονται οι αλλαγές που χρειάστηκε να πραγματοποιηθούν για να παραλληλοποιηθεί ο σειριακός κώδικας. Στη σειρά 3 προστείθεται η βιβλιοθήκη omp στο πρόγραμμα. Στη σειρά 6 δηλώνεται η συνάρτηση pi_est_omp που υπολογίζει την τιμή του π, ενώ στο σειριακό η συνάρτηση f πραγματοποιούσε μόνο το μέρος του υπολογισμού του π που αφορά τη συνάρτηση κατανομής Cauchy, $f(x) = \frac{1}{1+x^2}$. Στις σειρές 12 και 13 καλούνται οι συναρτήσεις omp_get_ num_procs, που επιστρέφει το πλήθος των επεξεργαστών του συστήματος που είναι διαθέσιμοι και η omp_get_max_threads, που επιστρέφει το μέγιστο αριθμό νημάτων που μπορούν να δημιουργηθούν.

Στη σειρά 35 ξεκινάει η παράλληλη περιοχή, δηλαδή το τμήμα του κώδικα που θα εκτελεστεί από όλα τα νήματα που δημιουργούνται τη στιγμή που η εκτέλεση του προγράμματος φτάσει σε αυτό το σημείο. Στη συνέχεια δηλώνονται οι μεταβλητές της παράλληλης περιοχής και η φράση reduction που εκτελεί μια πράξη αναγωγής στη μεταβλητή $sum2$ στο τέλος της παράλληλης περιοχής. Στη σειρά 41 εφαρμόζεται η οδηγία for που διαμοιράζει τις επαναλήψεις του βρόχου στα νήματα της παράλληλης περιοχής. Η σειρά 50 θα μπορούσε να συμπεριληφθεί στην παράλληλη εκτέλεση της οδηγίας for, αλλά σε αυτή την περίπτωση θα έπρεπε στο τέλος της παράλληλης περιοχής να γίνει αναγωγή στη μεταβλητή $estimate$ αντί για τη μεταβλητή $sum2$.

Και στα δύο προγράμματα χρησιμοποιούνται οι ίδιες μεταβλητές, αλλά για να επιτευχθεί σωστά η παραλληλοποίηση και να αποφευχθεί ο σχηματισμός συνθηκών ανταγωνισμού (Data Race Conditions)[1], εφαρμόζονται αλλαγές στον τρόπο χρήσης των μεταβλητών. Οι μεταβλητές h και n δηλώνονται μοιραζόμενες, δηλαδή κάθε νήμα μπορεί να πραγματοποιήσει αλλαγές σε αυτές, οι οποίες θα είναι ορατές από όλα τα νήματα, και οι μεταβλητές $i, sum1$ και x δηλώνονται ιδιωτικές, οπότε κάθε νήμα θα έχει δικό του αντίγραφο της μεταβλητής. Για κάθε νήμα δημιουργείται ένα ιδιωτικό αντίγραφο της μεταβλητής $sum2$. Στο τέλος των πράξεων, η τιμή αυτής της μεταβλητής που έχει κάθε νήμα, προστείθεται.

3.8.3 Παράλληλος υπολογισμός του π με MPI.

Στο πλαίσιο 3.3 φαίνεται η παραλληλοποίηση του υπολογισμού του π με τη χρήση του MPI. Στην πρώτη σειρά καλείται η βιβλιοθήκη του MPI. Στο πρόγραμμα αυτό δηλώνονται οι ίδιες μεταβλητές με το ακολουθιακό, $n, i, PI25DT, pi, h, sum$ και x, και επιπλέον οι $myid, numprocs$ και $mypi$. Η $myid$ χρησιμοποιείται ώς αναγνωριστικό για την κάθε διεργασία, και η $numprocs$ για το συνολικό αριθμό των διεργασιών που χρησιμοποιούνται.

Στη σειρά 13 αρχικοποιείται το περιβάλλον εκτέλεσης του MPI με τη ρουτίνα διαχείρησης περιβάλοντος MPI_Init, και αμέσως μετά προσδιορίζεται ο συνολικός αριθμός των διεργασιών με τη ρουτίνα MPI_Comm_size και δίνεται σε κάθε διεργασία ένας μοναδικός αριθμός που χρησιμοποιείται σαν ταυτότητα, με τη ρουτίνα MPI_Comm_rank.

Στις σειρές 18 εώς 20, η διεργασία με προσδιοριστικό αριθμό 0 ζητάει από τον χρήστη το

[1] Οι συνθήκες ανταγωνισμού, Race Conditions, είναι πραγραμματιστικά λάθη που παράγουν απρόβλεπτη συμπεριφορά και αποτελέσματα λόγω μή συγχρονισμένων ταυτόχρονων εκτελέσεων. Η ανίχνευσή τους σε παράλληλα προγράμματα μοιραζόμενης μνήμης μπορεί να είναι πολύ δύσκολη.

πλήθος των διαστημάτων n. Ο αριθμός αυτός διαδίδεται στις υπόλοιπες διεργασίες με χρήση της ρουτίνας συλλογικής επικοινωνίας MPI_Bcast. Στη συνέχεια, όλες οι διεργασίες εφαρμόζουν υπολογισμούς. Η εντολή for στη σειρά 27 ρυθμίζει ποιές επαναλήψεις θα εκτελέσει κάθε διεργασία, χρησιμοποιώντας τον προσδιοριστικό αριθμό κάθε διεργασίας και το συνολικό αριθμό των διεργασιών που συμμετέχουν στον υπολογισμό του π. Κάθε διεργασία, μετά το τέλος των επαναλήψεων, εκχωρεί το αποτέλεσμα των υπολογισμών της στη μεταβλητή *mypi*.

```
1  #include "mpi.h"
2  #include <math.h>
3  # include <stdio.h>
4
5  int main(argc,argv)
6  int argc;
7  char *argv[];
8  {
9      int done = 0, n, myid, numprocs, i;
10     double PI25DT = 3.14159265358979323846264 3;
11     double mypi, pi, h, sum, x;
12
13     MPI_Init(&argc,&argv);
14     MPI_Comm_size(MPI_COMM_WORLD,&numprocs);
15     MPI_Comm_rank(MPI_COMM_WORLD,&myid);
16     while (!done)
17     {
18     if (myid == 0) {
19         printf("Enter␣the␣number␣of␣intervals:␣(0␣quits)␣");
20         scanf("%d",&n);
21     }
22     MPI_Bcast(&n, 1, MPI_INT, 0, MPI_COMM_WORLD);
23     if (n == 0) break;
24
25     h   = 1.0 / (double) n;
26     sum = 0.0;
27     for (i = myid + 1; i <= n; i += numprocs) {
28         x = h * ((double)i - 0.5);
29         sum += 4.0 / (1.0 + x*x);
30     }
31     mypi = h * sum;
32
33     MPI_Reduce(&mypi, &pi, 1, MPI_DOUBLE, MPI_SUM, 0,
34            MPI_COMM_WORLD);
35
36     if (myid == 0)
37         printf("pi␣is␣approximately␣%.16f,␣Error␣is␣%.16f\n",
38            pi, fabs(pi - PI25DT));
39     }
40     MPI_Finalize();
41     return 0;
42 }
```

Listing 3.3: Παράλληλος κώδικας υπολογισμού του π με χρήση MPI .

Στη σειρά 33, με τη ρουτίνα συλλογικής επικοινωνίας MPI_Reduce, η διεργασία με προσδιοριστικό αριθμό 0 συλλέγει από όλες τις διεργασίες την τιμή της μεταβλητής *mypi*, αθροίζει

όλα τα δεδομένα που συλλέγει και τοποθετεί το άθροισμα στη μεταβλητή *pi*. Στη σειρά 40, τερματίζεται το περιβάλλον εκτέλεσης του MPI.

3.8.4 Υβριδικός παράλληλος υπολογισμός του π με MPI και OpenMP.

Στο πλαίσιο 3.4 παρατίθεται ο κώδικας της υβριδικής υλοποίησης του υπολογισμού του π με MPI και OpenMP

```
1  #include <stdio.h>
2  #include <math.h>
3  #include <omp.h>
4  #include "mpi.h"
5
6  int main(int argc, char *argv[])
7  {
8
9    double mypi,sumpi,h,sum,x,sum1;
10   int n,myrank,numprocs,i;
11   double PI25DT=3.14159265358979323846264 3;
12
13   MPI_Init(&argc, &argv);
14   MPI_Comm_rank(MPI_COMM_WORLD, &myrank);
15   MPI_Comm_size(MPI_COMM_WORLD, &numprocs);
16
17   n=10000000;
18   h=1.0/n;
19   sum=0.0;
20
21  #pragma omp parallel default(shared) private(i,x,sum1) reduction(+:sum)
22  {
23
24   sum1=0.0;
25   #pragma omp for
26   for (i = myrank+1; i <= n; i+= numprocs) {
27     x = h * ( i - 0.5 ) ;
28     sum1 += 4.0 / ( 1.0 + x*x ) ;
29     }
30     sum=sum+sum1;
31  }
32   mypi = h * sum;
33
34   MPI_Reduce(&mypi,&sumpi,1,MPI_DOUBLE,MPI_SUM,0,MPI_COMM_WORLD) ;
35
36   if (myrank == 0)
37     printf("pi␣is␣approximately␣%.16f,␣Error␣is␣%.16f␣\n",sumpi,fabs(
         sumpi-PI25DT)) ;
38
39   MPI_Finalize();
40   return 0;
41 }
```

Listing 3.4: Παράλληλος υβριδικός κώδικας υπολογισμού του π .

Σε αυτό το πρόγραμμα καλούνται και οι δύο βιβλιοθήκες, MPI και OpenMP. Στις σειρές από 13 ώς 15 αρχικοποιείται το περιβάλλον εκτέλεσης του MPI, προσδιορίζεται ο συνολικός αριθμός των διεργασιών και αντιστοιχίζεται σε κάθε διεργασία ένας μοναδικός αριθμός. Στη σειρά 21 ξεκινάει η παράλληλη περιοχή μέσα στην οποία δημιουργούνται νήματα. Κάθε διεργασία, όταν θα εισέλθει στην παράλληλη περιοχή, θα δημιουργήσει νήματα για να εκτελέσουν τους υπολογισμούς μέχρι το τέλος της παράλληλης περιοχής, όπου όλα τα νήματα θα τερματιστούν και θα συνεχίσει μόνο ένα. Οι μεταβλητές i, x και $sum1$ είναι ιδιωτικές σε κάθε νήμα, και η φράση reduction εκτελεί μια πράξη αναγωγής στη μεταβλητή sum. Όλες οι διεργασίες υπολογίζουν το μέρος που τους αντιστοιχεί, και στη σειρά 34, με την εφαρμογή της ρουτίνας συλλογικής επικοινωνίας MPI_Reduce, συλλέγονται όλες οι τιμές της μεταβλητής $mypi$ από τις διεργασίες και εκχωρούνται στη μεταβλητή $sumpi$.

Κεφάλαιο 4

Προγραμματιστικά Πλαίσια για Παράλληλη Συνεργατική Διήθηση.

Τα προγραμματιστικά πλαίσια (Software Frameworks) είναι λογισμικά που παρέχουν γενική λειτουργικότητα και μπορούν να τροποποιηθούν ώστε να παρέχουν ειδικές λειτουργίες σε μία εφαρμοφή. Περιέχουν συλλογές βιβλιοθηκών και παρέχουν διεπιφάνεια εφαρμογών προγραμματισμού (API). Ο σκοπός των προγραμματιστικών πλαισίων είναι να βελτιώσουν την αποδοτικότητα της δημιουργίας νέου λογισμικού και την παραγωγικότητα του προγραμματιστή. Στοχεύουν επίσης στη βελτίωση της ποιότητας και της αξιοπιστίας του λογισμικού. Στο κεφάλαιο αυτό περιγράφονται τα πιό διαδεδομένα προγραμματιστικά πλαίσια που χρησιμοποιούνται για την ανάπτυξη παράλληλων προγραμμάτων συνεργατικής διήθησης.

4.1 MapReduce.

Το MapReduce [50, 14] είναι ένα προγραμματιστικό πλαίσιο που δημιουργήθηκε για να υποστηρίξει τη χρήση κατενεμημένων υπολογισμών σε μεγάλα σύνολα δεδομένων και σε συστοιχίες υπολογιστών. Οι βιβλιοθήκες του MapReduce έχουν αναπτυχθεί σε πολλές γλώσσες, όπως c++, c#, java, python κ.α.

Η λειτουργία του πλαισίου αυτού είναι η επεξεργασία κατανεμημένων προβλημάτων σε πολύ μεγάλα σύνολα δεδομένων χρησιμοποιώντας μεγάλες συστοιχίες υπολογιστών. Τα δεδομένα μπορούν να είναι αποθηκευμένα σε ένα σύστημα αρχείων ή σε βάση δεδομένων. Το πλαίσιο MapReduce μετατρέπει μια λίστα ζευγών κλειδιών-τιμών σε μία λίστα τιμών, χωρίζοντας τη λειτουργία του σε δυό βήματα.

Το πρώτο είναι το βήμα Map, κατά το οποίο ο κόμβος που έχει το ρόλο του συντονιστή λαμβάνει τα δεδομένα, τα διαχωρίζει σε μικρότερα υποπροβλήματα και τα κατανέμει στους υπόλοιπους κόμβους. Οι υπόλοιποι κόμβοι μπορούν επίσης να χωρίσουν τα δεδομένα που θα λάβουν σε υποπροβλήματα, και να δημιουργηθεί τελικά μια δομή πολυεπίπεδου δένδρου. Οι κόμβοι επεξεργάζονται τα δεδομένα και επιστρέφουν στον συντονιστή τις απαντήσεις. Η συνάρτηση Map που γράφεται από ένα χρήστη της βιβλιοθήκης MapReduce, λαμβάνει ένα ζεύγος κλειδιού-τιμής και παράγει ένα σύνολο ζευγών ενδιάμεσων κλειδιών-τιμών. Η βιβλιοθήκη MapReduce ομαδοποιεί όλα τα ζεύγη που αντιστοιχούν στο αρχικό ζέυγος κλειδιού-

τιμής και τα παρέχει στη συνάρτηση Reduce.

Η συλλογή των αποτελεσμάτων των υποπροβλημάτων, από τον κόμβο συντονιστή, γίνεται στο βήμα Reduce. Τα αποτελέσματα συνδυάζονται με κατάλληλο τρόπο, ώστε να παράγουν τη λύση του αρχικού προβλήματος. Η συνάρτηση Reduce λαμβάνει ένα ενδιάμεσο κλειδί και ένα σύνολο τιμών που αντιστοιχούν στο κλειδί, και συγχωνεύει τις τιμές για να σχηματίσει ένα μικρότερο σύνολο τιμών.

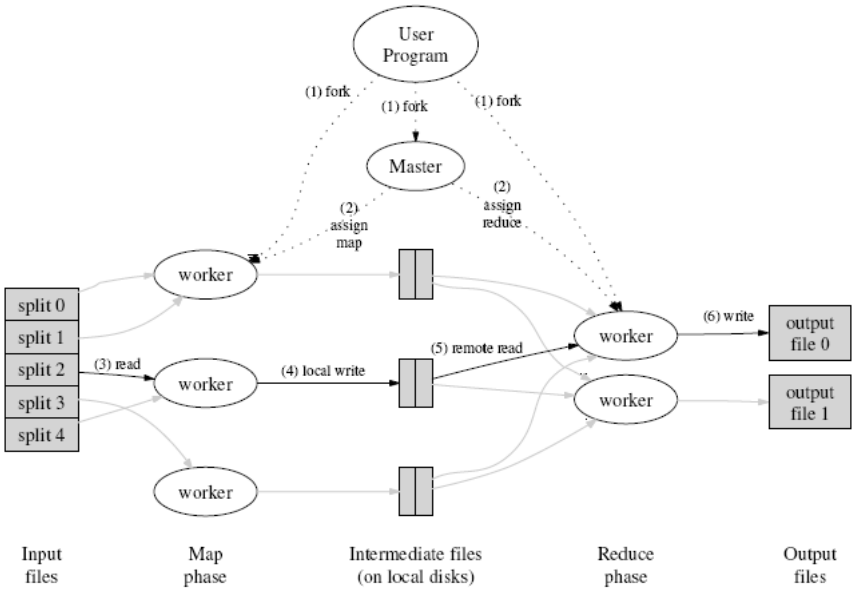
Το MapReduce επιτρέπει την εκτέλεση παράλληλων υπολογισμών σε διαφορετικούς επεξεργαστές, αναλαμβάνοντας τις λεπτομέρειες του παραλληλισμού, όπως η κατανομή των δεδομένων, η εξισορρόπηση φορτίου και η ανοχή σφαλμάτων.

Στην εικόνα 4.1 φαίνεται η συνολική ροή της MapReduce λειτουργίας. Όταν ένα πρόγραμμα καλεί τη MapReduce συνάρτηση συμβαίνουν τα εξής:

1. Η MapReduce βιβλιοθήκη χωρίζει τα δεδομένα σε M κομμάτια, και στη συνέχεια ξεκινάει αντίγραφα του προγράμματος σε μία συστοιχία υπολογιστών.
2. Ένας επεξεργαστής αναλαμβάνει το ρόλο του συντονιστή και αναθέτει εργασία στους υπόλοιπους. Θα ανατεθούν M εργασίες τύπου Map και R εργασίες τύπου Reduce.
3. Ένας εργαζόμενος, στον οποίο ανατίθεται μία εργασία τύπου Map διαβάζει τα δεδομένα, τα αναλύει σε ζεύγη κλειδιού-τιμής και τα εισάγει στη Map συνάρτηση που παράγει τα ενδιάμεσα ζεύγη κλειδιού-τιμής και τα φορτώνει στη μνήμη.
4. Περιοδικά, τα ζεύγη που είναι στη μνήμη εγγράφονται στον τοπικό δίσκο, χωρίζονται σε P περιοχές και η τοποθεσία στην οποία εγγράφονται επιστρέφεται στο συντονιστή, ο οποίος είναι υπεύθυνος για το διαμοιρασμό τους στους εργαζομένους για τη Reduce φάση.
5. Όταν ένας εργαζόμενος της φάσης Reduce ενημερώνεται από το συντονιστή για την τοποθεσία των δεδομένων, διαβάζει τα δεδομένα από τον τοπικό δίσκο των εργαζομένων της Map φάσης, και όταν έχει διαβάσει όλα τα ενδιάμεσα δεδομένα, τα ταξινομεί σύμφωνα με τα ενδιάμεσα κλειδιά και ομαδοποιεί όλα τα περιστατικά με το ίδιο κλειδί.
6. Ο εργαζόμενος, για κάθε μοναδικό ενδιάμεσο κλειδί που βρίσκει, εισάγει το κλειδί και τις αντίστοιχες τιμές στη Reduce συνάρτηση, και το αποτέλεσμα προσαρτάται σε ένα τελικό αρχείο. Στο τέλος θα υπάρχουν R αρχεία με το αποτέλεσμα της MapReduce.
7. Όταν τελειώσουν όλες οι Map και Reduce εργασίες, ο συντονιστής επιστρέφει στο αρχικό πρόγραμμα, που κάλεσε τη MapReduce συνάρτηση, και συνεχίζει την εκτέλεσή του.

4.2 Hadoop.

Το Hadoop είναι ένα προγραμματιστικό πλαίσιο ανοιχτού κώδικα για παραγωγή αξιόπιστων και κλιμακώσιμων προγραμμάτων. Επιτρέπει την κατανεμημένη επεξεργασία μεγάλων συνόλων δεδομένων σε συστοιχίες υπολογιστών, χρησιμοποιώντας ένα απλό προγραμματιστικό μοντέλο. Είναι σχεδιασμένο ώστε να επιτυγχάνει καλή κλιμάκωση είτε χρησιμοποιώντας έναν υπολογιστή, είτε χιλιάδες.



Σχήμα 4.1: Η MapReduce λειτουργία.

Στο Hadoop περιλαμβάνονται πολλές εφαρμογές. Υλοποιεί μία εφαρμογή που ονομάζεται Hadoop MapReduce, που είναι προγραμματιστικό πλαίσιο για κατανεμημένη επεξεργασία σε συστοιχίες υπολογιστών. Η εφαρμογή χωρίζεται σε μικρά τμήματα εργασίας τα οποία εκτελούνται σε οποιοδήποτε κόμβο της συστοιχίας. Επίσης, παρέχει ένα κατανεμημένο σύστημα αρχείων, HDFS (Hadoop Distributed File System), που αποθηκεύει δεδομένα στους κόμβους αυξάνοντας το συνολικό εύρος ζώνης στη συστοιχία. Το Hadoop MapReduce και το HDFS είναι σχεδιασμένα με τέτοιο τρόπο ώστε οι περιπτώσεις όπου κάποιος κόμβος αποτυγχάνει να χειριστεί τα δεδομένα, να αντιμετωπίζονται αυτόματα. Αν κάποιοι κόμβοι αποτύχουν να ολοκληρώσουν την εργασία που τους έχει ανατεθεί, η εργασία αυτή ανατίθεται σε άλλους κόμβους. Επιπλέον, στο Hadoop MapReduce, όσο περισσότερες Map και Reduce εργασίες υλοποιούνται, τόσο καλύτερη εξισορρόπηση φορτίου επιτυγχάνεται.

Η εφαρμογή Hadoop Map/Reduce λειτουργεί όπως και το προγραμματιστικό πλαίσιο MapReduce που περιγράφηκε στην προηγούμενη ενότητα. Λειτουργεί σε συστοιχίες υπολογιστών εκτελώντας εργασίες τύπου Map/Reduce στους κόμβους, και εφαρμόζει κατανεμημένους υπολογισμούς σε σύνολα δεδομένων που περιέχουν ζεύγη της μορφής κλειδί-τιμή. Η αρχιτεκτονική που ακολουθεί είναι αυτή του συντονιστή-εργαζομένου. Χρησιμοποιείται ένας εξυπηρέτης συντονιστής, ο οποίος αλληλεπιδρά με τους χρήστες και με το υπόλοιπο σύστημα. Οι χρήστες υποβάλλουν εργασίες τύπου Map ή Reduce στο συντονιστή και αυτός τις τοποθετεί σε μία σειρά και εξυπηρετεί ξεκινώντας από την παλαιότερη. Ο συντονιστής αναθέτει τις εργασίες στους εργαζόμενους εξυπηρέτες που έχει στη διάθεσή του. Σε κάθε κόμβο της συστοιχίας αντιστοιχεί ένας εργαζόμενος εξυπηρέτης. Οι εργαζόμενοι εκτελούν τις εργασίες και διαχειρίζονται τη διακίνηση των δεδομένων μεταξύ των Map και Reduce φάσεων.

Το HDFS είναι σχεδιασμένο για να αποθηκεύει αξιόπιστα πολύ μεγάλα αρχεία σε υπολογιστές μίας μεγάλης συστοιχίας υπολογιστών. Αποθηκεύει κάθε αρχείο σαν μία ακολουθία από μικρότερα ισομεγέθη αρχεία, εκτός από το τελευταίο που είναι διαφορετικού μεγέθους. Κρατείται αντίγραφο από αυτά τα μικρότερα αρχεία για να επιτευχθεί η ανεκτικότητα του

συστήματος σε σφάλματα. Σε αυτή την εφαρμογή, όπως και προηγουμένως, ακολουθείται η αρχιτεκτονική συντονιστή-εργαζομένων. Ο συντονιστής αναλαμβάνει όλες τις λειτουργίες ανοίγματος, κλεισίματος, μετονομασίας και ορίζει ποιό από τα αρχεία θα δώσει σε κάθε εργαζόμενο. Οι εργαζόμενοι εξυπηρετούν τις εντολές ανάγνωσης και εγγραφής στα αρχεία, δημιουργούν νέα αρχεία, δημιουργούν αντίγραφα ή διαγράφουν υπάρχοντα αρχεία, σύμφωνα με οδηγίες που λαμβάνουν από το συντονιστή.

Μεταξύ άλλων εφαρμογών που περιλαμβάνονται στο Hadoop είναι το Mahout, που είναι μία βιβλιοθήκη για κλιμακώσιμη εκμάθηση μηχανών και εξόρυξη δεδομένων, το Pig, μία υψηλού επιπέδου γλώσσα ροής δεδομένων για παράλληλους υπολογισμούς, και το ZooKeeper, που είναι μία υψηλής απόδοσης υπηρεσία για κατανεμημένες εφαρμογές [23].

4.3 Mahout.

Το Mahout είναι μία εφαρμογή με στόχο την υλοποίηση κατανεμημένων και κλιμακώσιμων αλγορίθμων Μηχανικής Μάθησης (Machine Learning) στην πλατφόρμα Hadoop. Οι κυριότεροι αλγόριθμοι του Mahout για συσταδοποίηση, ταξινόμηση, και συνεργατική διήθηση υλοποιούνται στο Hadoop χρησιμοποιώντας το πλαίσιο MapReduce. Ωστόσο η συλλογή βιβλιοθηκών αυτή, δεν περιορίζεται μόνο σε εφαρμογές που βασίζονται στο Hadoop, αλλά μπορεί να συμπεριλάβει και προγράμματα που αξιοποιούν ένα μόνο κόμβο ή μία συστοιχία χωρίς Hadoop. Οι κυριότερες βιβλιοθήκες του Mahout είναι βελτιστοποιημένες ώστε να παρέχουν καλές επιδόσεις ακόμα και σε μή κατανεμημένους αλγόριθμους.

Οι αλγόριθμοι που συμπεριλαμβάνονται στο Mahout αφορούν κυρίως τέσσερις τομείς. Την παραγωγή συστάσεων, εξετάζοντας τη συμπεριφορά των χρηστών και προσπαθώντας να προσδιορίσουν από αυτή τα αντικείμενα που είναι πιθανό να αρέσουν στους χρήστες. Τη συσταδοποίηση, όπου τα συσχετιζόμενα αντικείμενα ομαδοποιούνται. Την ταξινόμηση, όπου σύμφωνα με τα ήδη κατηγοριοποιημένα αντικείμενα, ταξινομούνται και τα υπόλοιπα και τέλος, το συχνό έλεγχο των συνόλων των αντικειμένων, με σκοπό την εύρεση αντικειμένων που επιλέγονται μαζί.

Στο Mahout περιλαμβάνεται μία μηχανή παραγωγής συστάσεων που δεν βασίζεται στο Hadoop. Αυτό το σύστημα συστάσεων περιλαμβανόταν παλαιότερα στην εφαρμοφή Taste, η οποία ενσωματώθηκε στο Mahout. Λαμβάνονται οι προτιμήσεις των χρηστών για τα αντικείμενα και υπολογίζονται οι προτιμήσεις για άλλα αντικείμενα. Το Mahout παρέχει πολλές εφαρμογές που μπορούν να χρησιμοποιηθούν ώστε να κατεσκευάσει κανείς το σύστημα συστάσεων που αντιπροσωπεύει καλύτερα τις ανάγκες του. Μεταξύ των εφαρμογών αυτών περιλαμβάνονται διεπιφάνειες για τον υπολογισμό της εγγύτητας μεταξύ χρηστών ή αντικειμένων, για το σχηματισμό γειτονιών χρηστών κ.α. Χρησιμοποιώντας τις εφαρμογές του Mahout μπορούν να σχεδιαστούν συστήματα συστάσεων βάσει μνήμης ή βάσει αντικειμένων, αλλά προς το παρόν δεν υποστηρίζονται συστήματα συστάσεων βάσει μοντέλου.

Για την εφαρμογή συστημάτων συστάσεων συνεργατικής διήθησης βάσει αντικειμένων, στο Mahout περιλαμβάνονται δύο Map/Reduce υλοποιήσεις. Η πρώτη προσδιορίζει όλα τα αντικείμενα που παρουσιάζουν εγγύτητα μεταξύ τους, λαμβάνει ένα αρχείο που περιέχει τις προτιμήσεις των χρηστών και παράγει ζεύγη τιμών αντικειμένων και εγγύτητας μεταξύ τους. Η δεύτερη είναι ένα κατανεμημένο σύστημα συστάσεων που λαμβάνει ένα αρχείο με τις προτιμήσεις των χρηστών και παράγει λίστες χρηστών και τα προτεινόμενα αντικείμενα μαζί με

μία τιμή που εκφράζει το βαθμό προτίμησής τους από το χρήστη [30].

4.4 GraphLab.

Το GraphLab είναι ένα πολύ πρόσφατο, υπό ανάπτυξη προγραμματιστικό πλαίσιο ανοικτού κώδικα για τη διευκόλυνση παράλληλων υλοποίησεων αλγορίθμων Μηχανικής Μάθησης. Χρησιμοποιεί ένα μοντέλο δεδομένων τύπου γραφήματος, που εκτός από την αναπαράσταση των δεδομένων εκφράζει και τις υπολογιστικές εξαρτήσεις. Το Graphlab απαλλάσσει τους χρήστες από τη ρύθμιση παραγόντων όπως ο συγχρονισμός και η αποφυγή φαινομένων συνθηκών ανταγωνισμού (data race), παρέχοντας μία υψηλού επιπέδου αναπαράσταση των δεδομένων μέσω γραφημάτων.

Παρέχει λειτουργίες που μοιάζουν με αυτές του προγραμματιστικού πλαισίου MapReduce. Ανάλογη με τη Map λειτουργία, είναι μία λειτουργία ενημερώσεων (update function), η οποία μπορεί να διαβάσει και να τροποποιήσει επικαλυπτόμενα αρχεία δεδομένων, με τρόπο που ορίζει ο χρήστης που παρέχει στο σύστημα το γράφημα των δεδομένων. Οι διάφορες καταστάσεις του προγράμματος αναπαρίστανται με αυθαίρετα τμήματα μνήμης τα οποία συσχετίζονται με τις ακμες και τις κορυφές του γραφήματος. Οι λειτουργίες ενημερώσεων μπορούν να ενεργοποιηθούν αναδρομικά και να επιτυγχάνονται με αυτό τον τρόπο δυναμικοί επαναληπτικοί υπολογισμοί. Η λειτουργία συγχρονισμού (sync operation), είναι ανάλογη με τη Reduce λειτουργία, ορίζει την ολική συγκέντρωση των αποτελεσμάτων των υπολογισμών και μπορεί να εκτελείται σε περιβάλλοντα μεγάλων εξαρτήσεων. Επιπλέον, μπορεί να εφαρμόζεται ταυτόχρονα με τη λειτουργία των ενημερώσεων.

Το μοντέλο δεδομένων του GraphLab αποτελείται από δύο μέρη. Ένα γράφημα δεδομένων και ένα πίνακα. Το γράφημα, $G = (V, E)$, κωδικοποιεί την υπολογιστική δομή του προβλήματος και τις καταστάσεις του προγράμματος που μπορούν να υποστούν τροποποιήσεις. Ο χρήστης συσχετίζει αυθαίρετα μέρη δεδομένων ή παραμέτρους με κάθε κορυφή και με κάθε προσανατολισμένη ακμή του γραφήματος. Ο πίνακας είναι κοινόχρηστος (shared data table SDT), και περιέχει ζεύγη κλειδιών-τιμών που συσχετίζουν τιμές κλειδιών με τα αυθαίρετα τμήματα δεδομένων.

Εφόσον τα διάφορα αρχεία δεδομένων μπορεί να επικαλύπτονται, η ταυτόχρονη εκτέλεση των λειτουργιών ενημέρωσης μπορεί να οδηγήσει σε φαινόμενα συνθηκών ανταγωνισμού και να δημιουργήσει ασυνέπεια ή ακόμα και αλλοίωση των δεδομένων. Το GraphLab παρέχει τρία μοντέλα για την επίτευξη συνέπειας δεδομένων που ρυθμίζουν ποιά επικαλυπτόμενα μέρη δεδομένων μπορούν να τροποποιούνται ταυτόχρονα. Το μοντέλο πλήρους συνέπειας εξασφαλίζει ότι καμία άλλη λειτουργία δεν θα διαβάζει ή θα τροποποιεί τα δεδομένα που χρησιμοποιούνται από άλλη λειτουργία. Σε αυτό το μοντέλο ο παραλληλισμός περιορίζεται στις κορυφές που δέν έχουν κοινούς γείτονες. Το μοντέλο συνέπειας ακμών εξασφαλίζει ότι καμία άλλη λειτουργία δεν θα διαβάζει ή θα τροποποιεί τα δεδομένα που χρησιμοποιούνται από ακμές προσκείμενες μεταξύ τους, και το μοντέλο συνέπειας κορυφών που εξασφαλίζει ότι όσο εφαρμόζεται μία λειτουργία σε ένα αρχείο δεδομένων, καμία άλλη λειτουργία δεν θα εφαρμόζεται σε αυτό.

Στο GraphLab παρέχονται επίσης εργαλεία συγχρονισμού που καθορίζουν την προτεραιότητα εκτέλεσης των υπολογισμών και μπορεί να εξαρτώνται από τα δεδομένα. Οι κορυφές μπορούν να ανανεώνονται ταυτόχρονα ή σειριακά. Η σειρά με την οποία εφαρμόζονται

οι λειτουργίες ενημέρωσης στις κορυφές ορίζεται από μία παράλληλη δομή δεδομένων που καλείται δρομολογητής (scheduler), η οποία περιέχει μία δυναμική λίστα ζευγών κορυφών-λειτουργιών.

Οι συνθήκες τερματισμού που χρησιμοποιούνται σε πολλούς επαναληπτικούς αλγόριθμους εκμάθησης μηχανών απαιτούν τη γνώση της συνολικής κατάστασης των δεδομένων. Το GraphLab παρέχει δύο τρόπους για την ανίχνευση των συνθηκών τερματισμού. Ο πρώτος βασίζεται στον δρομολογητή, ο οποίος δίνει το σήμα τερματισμού όταν δεν υπάρχουν άλλες λειτουργίες προς εκτέλεση. Ο δεύτερος βασίζεται στην παροχή από τον χρήστη λειτουργιών που εξετάζουν τον κοινόχρηστο πίνακα και παράγουν εντολές τερματισμού όταν ο αλγόριθμος συγκλίνει [18, 28].

Κεφάλαιο 5

Επισκόπηση Παράλληλων Αλγορίθμων Συνεργατικής Διήθησης.

Ο πολύ μεγάλος όγκος δεδομένων και η ανάγκη για γρηγορότερη επεξεργασία τους, καθιστά αναγκαία την εφαρμογή μεθόδων ώστε να επιτευχθεί όσο το δυνατόν καλύτερη κλιμάκωση σε μικρότερο χρόνο εκτέλεσης των αλγορίθμων. Η εφαρμογή μεθόδων παράλληλου προγραμματισμού είναι φυσιολογικό να είναι πολύ διαδεδομένη στους αλγορίθμους συνεργατικής διήθησης, αφού αυτές οι μέθοδοι έχουν ώς στόχο να επιτύχουν καί τη βελτίωση της κλιμάκωσης και τη μείωση του χρόνου εκτέλεσης. Σε αυτό το κεφάλαιο θα περιγραφούν εφαρμογές αλγορίθμων συνεργατικής διήθησης που υλοποιούνται με μεθόδους παράλληλης επεξεργασίας.

5.1 ALS με -λ- κανονικοποίηση με βάρη.

Στο [64] παρουσιάζεται ένας παράλληλος αλγόριθμος της μεθόδου ALS με σταθμισμένη κατά λ κανονικοποίηση σε Matlab. Σε κάθε χρήστη και σε κάθε αντικείμενο αντιστοιχεί ένα διάνυσμα χαρακτηριστικών. Ο πίνακας U είναι ο πίνακας που περιέχει τα διανύσματα των χαρακτηριστικών των χρηστών, και ο πίνακας M περιέχει τα διανύσματα των χαρακτηριστικών των αντικειμένων. Η παραλληλοποίηση εφαρμόζεται στις ενημερώσεις των πινάκων U και M. Ο πίνακας των εκτιμήσεων R κατανέμεται στους υπολογιστές του συστήματος με δύο τρόπους. Μία κατανομή ανά σειρές του πίνακα R, δηλαδή ανά χρήστες και μία ανά στήλες, δηλαδή ανά αντικείμενα. Κάθε υπολογιστής υπολογίζει τις δικές του ενημερώσεις των πινάκων U και M για τα μέρη του πίνακα R που έχει στη διάθεσή του. Για τον υπολογισμό του πίνακα U, όλοι οι υπολογιστές χρειάζονται τον πίνακα M, επομένως εφαρμόζεται μία "gather" συνάρτηση. Αντίστροφα συμβαίνει για τον υπολογισμό του M. Για τις ενημερώσεις των πινάκων U και M κάθε υπολογιστής υπολογίζει το κομμάτι που του αντιστοιχεί και μετά εφαρμόζονται οι "gather" συναρτήσεις.

Η απόδοση του αλγόριθμου αυτού μετρήθηκε σε μία συστοιχία υπολογιστών με 30 επεξεργαστές με λειτουργικό σύστημα Linux, στο σύνολο δεδομένων Netflix, και το RMSE που επιτεύχθηκε είναι 0,8985. Ο αλγόριθμος ALS-WR χρειάστηκε 2,5 ώρες για να πραγματο-

ποιήσει την εκπαίδευση του μοντέλου για 30 επαναλήψεις του ALS.

5.2 Αλγόριθμος που βασίζεται στην τεχνική αποσύνθεσης εννοιών.

Ο αλγόριθμος που περιγράφεται στη συνέχεια έχει τέσσερα στάδια, τα οποία στην batch λειτουργία[1] εκτελούνται το ένα μετά το άλλο. Τα στάδια αυτά είναι ο σφαιρικός k-means υπολογισμός, η κατασκευή των διανυσμάτων, ο πολλαπλασιασμός των αραιών πινάκων και η παραγωγή των προβλέψεων. Κάθε στάδιο έχει σχεδιαστεί με μία δομή υπολογισμών που βασίζεται σε ένα από κάτω προς τα πάνω δέντρο. Κάθε κόμβος σε ένα επίπεδο του δέντρου μπορεί να εκτελεστεί ανεξάρτητα από τους άλλους κόμβους του ίδιου επιπέδου, αλλά εξαρτάται από την εκτέλεση ενός υποσυνόλου κόμβων του προηγούμενου επιπέδου του δέντρου. Έχουν ληφθεί μέτρα για την εξισορρόπηση φορτίου μεταξύ των κόμβων. Αντί για τη χρήση φράγματος μεταξύ των διαφορετικών επιπέδων του δέντρου, πραγματοποιείται έλεγχος εξάρτησης για τον κόμβο του επόμενου επιπέδου, ώστε να ξεκινήσουν οι υπολογισμοί χωρίς να έχουν τελειώσει τους υπολογισμούς όλοι οι κόμβοι του τρέχοντος επιπέδου.

Επιπλέον, όσο οι υπολογισμοί στο δέντρο κατευθύνονται προς τη ρίζα του, το πλήθος των κόμβων που εφαρμόζουν τις επαναλήψεις του k-means ελαττώνεται. Έτσι μειώνεται και η χρήση νημάτων. Για να βελτιωθεί η χρήση νημάτων εφαρμόζεται η τεχνική του διαχωρισμού των διαστάσεων (Dimension partitioning), όπου κάθε διάνυσμα χαρακτηριστικών χωρίζεται σε σταθερό αριθμό κομματιών. Για τον υπολογισμό του σφαιρικού μέσου κάθε κόμβου χρησιμοποιούνται πολλά νήματα και κάθε ένα από αυτά υπολογίζει ένα εύρος των διαστάσεων του διανύσματος χαρακτηριστικών. Στο τέλος εφαρμόζεται μία αναγωγική πράξη άθροισης των υπολογισμένων τιμών. Η τεχνική αυτή χρησιμοποιείται στα υψηλότερα επίπεδα του δέντρου όταν το διαθέσιμο πλήθος των νημάτων είναι μεγαλύτερο από το πλήθος των κόμβων στους οποίους εφαρμόζεται ο k-means.

Στην online λειτουργία[2] οι αλλαγές στα δεδομένα που είναι δυνατόν να συμβούν είναι η προσθήκη νέου διανύσματος χαρακτηριστικών, η προσθήκη νέας διάστασης, και ενημερώσεις στις τιμές των διαστάσεων των υπάρχοντων διανυσμάτων. Σε προκαθορισμένα χρονικά διαστήματα το σύνολο των αλλαγών που προκύπτουν αποστέλλονται για επεξεργασία στα τέσσερα στάδια του αλγορίθμου. Ο αλγόριθμος υπολογίζει το υπολογιστικό φορτίο κάθε σταδίου και αναλογικά εκχωρεί νήματα στα στάδια. Οι αλλαγές στους κόμβους του δέντρου σημειώνονται αυτόματα και οι επανυπολογισμοί προσδίδονται δυναμικά στα διαθέσιμα νήματα. Με αντίστοιχο τρόπο πραγματοποιούνται οι υπολογισμοί και στα υπόλοιπα στάδια του αλγορίθμου.

Η εφαρμογή του αλγόριθμου αυτού χρησιμοποιεί Posix Threads NPTL API σε συστοιχία υπολογιστών με 32 πυρήνες και λειτουργικό σύστημα Linux. Για το σύνολο δεδομένων Netflix, ο χρόνος παραγωγής 1.4M προβλέψεων είναι 4.5 sec, και RMSE=0.84. Ο χρόνος εκπαίδευσης του μοντέλου είναι 64 sec [36].

[1] Στη Batch λειτουργία ο αλγόριθμος εφαρμόζεται σε ένα ολοκληρωμένο σύνολο δεδομένων.

[2] Στην Online λειτουργία, όσο εφαρμόζεται ο αλγόριθμος, εισέρχονται νέα δεδομένα στο μοντέλο.

5.3 Αλγόριθμοι συσταδοποίησης.

5.3.1 Βελτιστοποιημένος κατανεμημένος αλγόριθμος με βάση τη συσταδοποίηση.

Στον αλγόριθμο αυτό χρησιμοποιούνται c νήματα ανά κόμβο, τα οποία στη συνέχεια αναφέρονται ώς $T_1, T_2, ..., T_c$. Ο αλγόριθμος χρειάζεται τον πίνακα A των εκτιμήσεων με m γραμμές και n στήλες, ένα μή μηδενικό πίνακα W, το πλήθος των συστάδων ανά σειρές l και το πλήθος των συστάδων ανά στήλες k, και υπολογίζει την τοπικά βέλτιστη συσταδοποίηση (ρ, γ) και τους μέσους των συστάδων ανά στήλες A_{ih}^{CC} ,των συστάδων ανά γραμμές A_{gj}^{RC} και των συσταδοποιήσεων A_{gh}^{COC}.

Κάθε κόμβος p λαμβάνει $m_p = \frac{m}{P_0}$ γραμμές και $n_p = \frac{n}{P_0}$ στήλες, όπου P_0 είναι το συνολικό πλήθος των κόμβων. Τα νήματα κάθε κόμβου p λαμβάνουν το καθένα από $m_{p'}$ γραμμες και $n_{p'}$ στήλες, όπου $m_{p'} = \frac{m_p}{c}$ και $n_{p'} = \frac{n_p}{c}$.

Κάθε νήμα T_i αρχικοποιεί τυχαία μια συσταδοποίηση (ρ_i^p, γ_i^p). Το νήμα T_1 συλλέγει όλα τα αθροίσματα και τα βάρη $(S_i^R, S_j^C, W_i^R, W_j^C \forall i, j)$ από τα άλλα νήματα χρησιμοποιώντας τη ρουτίνα MPI_Allgather. Όλα τα νήματα υπολογίζουν τους μέσους όλων των γραμμών και στηλών A_i^R και A_j^C. Το νήμα T_1 συγκεντρώνει όλα τα (ρ^p) και τα συνενώνει για να δημιουργήσει το (ρ), δηλαδή το βαθμό συμμετοχής των σειρών στη συστάδα.

Όσο η τιμή του RMSE δεν συγκλίνει στο επιτρεπτό εύρος σφάλματος, τα νήματα στους κόμβους υπολογίζουν τα εξής:

- Τα νήματα $T_2, ..., T_c$ υπολογίζουν τους μέσους της μερικής συνεισφοράς στην κατά γραμμές συσταδοποίηση A_{gj}^{RC} και το νήμα T_1 χρησιμοποιώντας MPI_Allgather υπολογίζει το βαθμό συμμετοχής των στηλών στη συστάδα, συνενώνοντας τα (γ^p).

- Τα νήματα $T_2, ..., T_c$ υπολογίζουν τους μέσους της μερικής συνεισφοράς στην κατά στήλες συσταδοποίηση A_{ih}^{CC} και το νήμα T_1 χρησιμοποιώντας MPI_Allreduce υπολογίζει τους συνολικούς μέσους της συσταδοποίησης κατά γραμμές A_{gh}^{RC}.

- Τα νήματα $T_2, ..., T_c$ υπολογίζουν τη συνεισφορά των τοπικών γραμμών και στηλών στα αθροίσματα και τα βάρη της συσταδοποίησης. (S_p^{COC}, W_p^{COC}). Το νήμα T_1 υπολογίζει τους μέσους της ολικής συσταδοποίησης A_{ih}^{RC}.

- Το νήμα T_1 υπολογίζει τα αθροίσματα και τα βάρη της συσταδοποίησης (S^{COC}, W^{COC}) και υπολογίζει το A^{COC}. Τα υπόλοιπα νήματα υπολογίζουν τα $\hat{A}^R(i, j, g)$ και $\hat{A}^C(i, j, h)$.

- Όλα τα νήματα ενημερώνουν τα ρ^p ενημερώνοντας πρώτα τον $\hat{A}^R(i, j, g)$ με τους μέσους της συσταδοποίησης και παράγουν τον πίνακα $\hat{A}_{ij}$.

- Το νήμα T_1 συγκεντρώνει όλα τα (ρ^p) και τα συνενώνει για να δημιουργήσει το (ρ), δηλαδή το βαθμό συμμετοχής των σειρών στη συστάδα. Τα υπόλοιπα νήματα ενημερώνουν τα γ^p ενημερώνοντας πρώτα τον $\hat{A}^C(i, j, h)$ με τους μέσους της συσταδοποίησης και παράγουν τον πίνακα $\hat{A}_{ij}$.

Ο αλγόριθμος αυτός δοκιμάστηκε σε ένα σύστημα Blue Gene/P με 1024 κόμβους για το σύνολο δεδομένων Netflix και χρειάστηκε χρόνο εκπαιδεύσεως του μοντέλου περίπου 6 δευτερόλεπτα, παρήγαγε 1.4M προβλέψεις σε 2.5 sec και RMSE περίπου 0.87 [37].

5.3.2 Συσταδοποίηση με χρήση MPI.

Στο [20] περιγράφεται ένας αλγόριθμος που ταυτόχρονα δημιουργεί τις γειτονιές χρηστών και αντικειμένων και παράγει προβλέψεις βασιζόμενος στις τιμές των μέσων εκτιμήσεων των γειτονιών συσταδοποίησης, λαμβάνοντας υπόψην και τα ξεχωριστά χαρακτηριστικά των χρηστών και των αντικειμένων.

Ο αλγόριθμος πραγματοποιείται σε τρία στάδια. Το πρώτο είναι η στατική εκπαίδευση κατά την οποία γίνεται η συσταδοποίηση και ο υπολογισμός κάποιων στατιστικών στοιχείων που χρησιμοποιούνται στις προβλέψεις. Το δεύτερο είναι οι προβλέψεις, που προσεγγίζουν την άγνωστη αξιολόγηση χρησιμοποιώντας τα στατιστικά που έχουν υπολογισθεί, και τρίτο, η σταδιακή εκπαίδευση κατά την οποία ενημερώνονται τα στατιστικά βάσει των εισερχόμενων εκτιμήσεων.

Σε κάθε επανάληψη της συσταδοποίησης πραγματοποιούνται τρείς ενέργειες. Ο υπολογισμός των μέσων τιμών των πινάκων, ο σχηματισμός των συστάδων ανά γραμμές και ο σχηματισμός των συστάδων ανά στήλες. Η παραλληλοποίηση του σταδίου της συσταδοποίησησης γίνεται μοιράζοντας τις γραμμές και τις στήλες στους επεξεργαστές έτσι ώστε ο σχηματισμός των συστάδων ανά γραμμές και ανά στήλες να γίνεται ταυτόχρονα. Για τον υπολογισμό των μέσων των πινάκων, οι διάφοροι επεξεργαστές υπολογίζουν ένα μέρος των υπολογισμών και τα αποτελέσματα συνενώνονται.

Οι πίνακες και τα στατιστικά που έχουν υπολογιστεί κατανέμονται στους διάφορους επεξεργαστές για αποθήκευση. Όταν ζητείται νέα εκτίμηση ενός αντικειμένου p_j για ένα χρήστη u_i, ο επεξεργαστής με το ρόλο του συντονιστή προσδιορίζει πρώτα τις συστάδες στις οποίες ανήκουν το αντικείμενο και ο χρήστης και αποδίδει το αίτημα της εκτίμησης του αντικειμένου σε έναν εργαζόμενο επεξεργαστή που έχει στη διάθεσή του τις κατάλληλες συστάδες. Σε περίπτωση νέου χρήστη ή νέου αντικειμένου, ο συντονιστής τους αντιστοιχεί σε κατάλληλη συστάδα και ορίζει τον κατάλληλο εργαζόμενο για να εκτελέσει τους υπολογισμούς.

Για την αξιολόγηση του αλγορίθμου αυτού χρησιμοποιήθηκαν τα σύνολα δεδομένων της MovieLens και του BookCrossing [39]. Ο αλγόριθμος υλοποιήθηκε στη γλώσσα C++ με χρήση της διεπαφής MPI και εκτελέσθηκε σε συστοιχία υπολογιστών με 256 επεξεργαστές και λειτουργικό συτημα Linux. Η ακρίβεια των προβλέψεων μετρήθηκε με χρήση του MAE.

5.3.3 Συσταδοποίηση με χρήση της βιβλιοθήκης DataRush.

Ένας ακόμα παράλληλος αλγόριθμος που βασίζεται στη συσταδοποίηση για να παράγει προβλέψεις των αξιολογήσεων των αντικειμένων περιγράφεται στο [13]. Στην υλοποίηση αυτή δημιουργούνται διαγράμματα ροής δεδομένων που εκτελούνται παράλληλα. Ένα διάγραμμα ροής δεδομένων χρησιμοποιείται για τη δημιουργία των γειτονιών συσταδοποίησης. Τα δεδομένα χωρίζονται σε τόσα μέρη όσοι και οι διαθέσιμοι πυρήνες του συστήματος. Κάθε πυρήνας επεξεργάζεται τα δεδομένα που του αντιστοιχούν και υπολογίζει τα απαραίτητα στατιστικά

στοιχεία. Σε κάθε επανάληψη συγκροτείται και εκτελείται νέο διάγραμμα ροής δεδομένων και οι επαναλήψεις συνεχίζονται μέχρι η αλλαγή στο RMSE να είναι μικρότερη ή ίση με το κατώφλι 1×10^{-8}.

Για την παραγωγή των προβλέψεων δημιουργείται άλλο διάγραμμα ροής δεδομένων. Κάθε πυρήνας υπολογίζει τις προβλέψεις ανάλογα με το αν ο χρήστης και το αντικείμενο είναι νέα στο σύστημα.

Ο αλγόριθμος αυτός για την παραλληλοποίηση χρησιμοποιεί τη βιβλιοθήκη και μηχανή ροής δεδομένων DataRush που κατασκευάζει και εκτελεί διαγράμματα ροής δεδομένων σε Java. Σε σύστημα δύο τετραπύρηνων επεξεργαστών με λειτουργικό σύστημα Windows, υπολογίσθηκε ακρίβεια προβλέψεων $RMSE = 0.88846$.

5.4 Αλγόριθμοι βασισμένοι σε προγραμματιστικά πλαίσια.

Πρόσφατα παρατηρείται έντονη τάση ανάπτυξης εφαρμογών εκμάθησης μηχανών με τη χρήση προγραμματιστικών πλαισίων όπως το Hadoop και το Graphlab, που διευκολύνουν την παραλληλοποίηση και στοχεύουν στην επεξεργασία μεγάλων συνόλων δεδομένων. Στη συνέχεια της ενότητας αυτής περιγράφονται τέτοιες εφαρμογές παραλληλοποίησης αλγορίθμων συνεργατικής διήθησης.

5.4.1 Αλγόριθμος συστάσεων βάσει χρηστών στο προγραμματιστικό πλαίσιο Hadoop.

Το Hadoop [23] είναι ένα ανοιχτού κώδικα λογισμικό που επιτρέπει την κατανεμημένη επεξεργασία μεγάλων συνόλων δεδομένων μέσω συστοιχίων υπολογιστών χρησιμοποιώντας ένα απλό προγραμματιστικό μοντέλο. Μεταξύ άλλων περιλαμβάνει ένα κατανεμημένο σύστημα αρχείων (Hadoop Distributed File System HDFS), το πλαίσιο MapReduce για κατανεμημένη επεξεργασία μεγάλων συνόλων δεδομένων σε συστοιχίες υπολογιστών, τη βιβλιοθήκη Mahout για κλιμακωτές εφαρμογές εκμάθησης μηχανών (machine learning) και εξόρυξης δεδομένων (data mining) και το πλαίσιο Pig που είναι μια υψηλού επιπέδου γλώσσα ροής δεδομένων για παράλληλους υπολογισμούς.

Το πλαίσιο MapReduce υλοποιεί τους υπολογισμούς σε δύο φάσεις, τη φάση Map και τη φάση Reduce. Στην πρώτη λαμβάνει ένα σύνολο δεδομένων με τη μορφή ζεύγων κλειδί/τιμή και παράγει μία έξοδο ζεύγων με την ίδια μορφή. Όλες οι τιμές που σχετίζονται με το ίδιο κλειδί ομαδοποιούνται. Στη φάση Reduce εισέρχονται οι τιμές που συσχετίζονται με ένα κλειδί, και το κλειδί. Οι τιμές συγχωνεύονται και σχηματίζουν ένα μικρότερο σύνολο τιμών.

Η πρώτη φάση του αλγορίθμου συνεργατικής διήθησης είναι ο διαμοιρασμός των δεδομένων. Αποθηκεύεται ο αριθμός του χρήστη userID σε διάφορα αρχεία, όπου κάθε σειρά αντιστοιχεί σε άλλο χρήστη. Αυτά τα αρχεία χρησιμοποιούνται σαν δεδομένα στη φάση Map. Η πλατφόρμα Hadoop υπολογίζει τις απαιτήσεις του αλγορίθμου σε μνήμη και αν διαθέτει τους απαραίτητους πόρους ξεκινάει έναν mapper. Ο mapper διαβάζει το αρχείο ανά σειρά. Λαμβάνει τον αριθμό σειράς ώς κλειδί και το userID που αντιστοιχεί σε αυτή τη σειρά σαν τιμή.

Υπολογίζει την εγγύτητα του χρήστη με άλλους χρήστες και προσδιορίζει τους εγγύτερούς του. Υπολογίζει στη συνέχεια τις προβλέψεις των αξιολογήσεων αντικειμένων. Η τιμή userID και οι προβλέψεις είναι τα ζεύγη κλειδιού/τιμής που θα χρησιμοποιηθούν στην επόμενη φάση. Στη φάση Reduce ταξινομούνται οι λίστες συστάσεων ανά χρήστη και αποθηκεύονται στο σύστημα αρχείων.

Στο [63] υλοποιείται ο παραπάνω αλγόριθμος συνεργατικής διήθησης βάσει χρηστών στο Hadoop. Η υλοποίηση του αλγορίθμου πραγματοποιήθηκε σε 9 διπύρηνους υπολογιστές με λειτουργικό σύστημα Linux, χρησιμοποιώντας το σύνολο δεδομένων Netflix. Μετρήθηκε η επιτάχυνση για διάφορα μεγέθη του συνόλου δεδομένων.

5.4.2 Αλγόριθμος συστάσεων βάσει αντικειμένων στο προγραμματιστικό πλαίσιο Hadoop.

Ο αλγόριθμος συνεργατικής διήθησης βάσει αντικειμένων που περιγράφεται στη συνέχεια χρησιμοποιεί την μετρική εγγύτητας Pearson και παράγει τις προβλέψεις υπολογίζοντας τη μέση εκτίμηση του χρήστη με χρήση βαρών.

Στον αλγόριθμο αυτό τα τρία μέρη με το μεγαλύτερο υπολογιστικό κόστος διαχωρίζονται σε τέσσερις MapReduce φάσεις. Ο υπολογισμός της μέσης εκτίμησης για κάθε αντικείμενο υπολογίζεται στην πρώτη MapReduce φάση, ο υπολογισμός της εγγύτητας μεταξύ ζευγών αντικειμένων υπολογίζεται στη δεύτερη MapReduce φάση και ο υπολογισμός των προβλεπόμενων αξιολογήσεων για ένα χρήστη πραγματοποιείται σε δύο MapReduce στάδια. Στο πρώτο δημιουργείται ο πίνακας που περιέχει τις τιμές εγγύτητας και στο δεύτερο υπολογίζονται οι προβλέψεις. Τα δεδομένα με ίδιο userID ή ίδιο itemID αναθέτονται στον ίδιο mapper και reducer.

Τα πειράματα εκτελούνται σε μία συστοιχία τριών υπολογιστών όπου ο ένας αναλαμβάνει το ρόλο του συντονιστή. Για το σύνολο δεδομένων MovieLens μετράται η επιτάχυνση του αλγορίθμου. Σε κάθε κόμβο εκτός του συντονιστή υποστηρίζονται εώς και πέντε MapReduce φάσεις [25].

5.4.3 Βιβλιοθήκη ανοιχτού κώδικα για συνεργατική διήθηση στο GraphLab.

Το GraphLab είναι ένα παράλληλο προγραμματιστικό πλαίσιο ανοιχτού κώδικα για τον κλάδο της Μηχανικής Μάθησης. Έχει ώς στόχο το σχεδιασμό και την υλοποίηση αποδοτικών παράλληλων αλγορίθμων. Παρέχει λειτουργίες παρόμοιες με τη MapReduce και επιτυγχάνει υψηλού βαθμού επιδόσεις σε παράλληλους αλγόριθμους [18, 28].

Στο [58] περιγράφονται δώδεκα αλγόριθμοι συνεργατικής διήθησης και παρουσιάζεται η απόδοσή τους, χρησιμοποιώντας το παράλληλο προγραμματιστικό πλαίσιο GraphLab. Από τους αλγόριθμους αυτούς, οι ALS, wALS, SVD++, PMF, BPTF και SGD συμπεριλαμβάνονται στην ανοιχτού κώδικα βιβλιοθήκη συνεργατικής διήθησης του GraphLab, που είναι διαθέσιμη στο [18].

Για τη ρύθμιση των παραμέτρων των αλγορίθμων και τη μέτρηση της απόδοσής τους χρησιμο-

ποιήθηκε μια συστοιχία τεσσάρων οκταπύρηνων υπολογιστών καθώς και ο υπερυπολογιστής BlackLight [5]. Μετρήθηκε το RMSE και η επιτάχυνση της παράλληλης υλοποίησής τους.

5.4.4 Παράλληλες προσεγγίσεις του αλγόριθμου SGD.

5.4.4.1 Ο αλγόριθμος SGD σε σταθερό σύνολο δεδομένων.

Ο αλγόριθμος SGD μπορεί να παραλληλοποιηθεί κατανέμοντας το φόρτο εργασίας σε d επεξεργαστές. Ο πίνακας των εκτιμήσεων R χωρίζεται σε υποπίνακες και αυτοί κατανέμονται στους επεξεργαστές. Δημιουργούνται διαφορετικά σύνολα υποπινάκων έτσι ώστε κάθε υποπίνακας ενός συνόλου να μην επικαλύπτει τις γραμμές ή τις στήλες των άλλων υποπινάκων του συνόλου. Με αυτόν τον τρόπο όταν οι διεργασίες εργάζονται σε ένα σύνολο υποπινάκων δεν πραγματοποιούνται αντικρουόμενες ενημερώσεις του αλγορίθμου. Τα σύνολα υποπινάκων που δημιουργούνται πρέπει να καλύπτουν όλο τον πίνακα R και να είναι ξένα μεταξύ τους. Ένας επεξεργαστής αναλαμβάνει το ρόλο του συντονιστή και οι υπόλοιποι την επεξεργασία των δεδομένων.

Ο αλγόριθμος υλοποιήθηκε με χρήση MapReduce σε μία Hadoop συστοιχία 40 κόμβων, όπου καθένας από αυτούς διαθέτει δύο τετραπύρηνους επεξεργαστές, για το σύνολο δεδομένων του διαγωνισμού Netflix και μετρήθηκε η κλιμάκωση και η επιτάχυνσή του. [19]

5.4.4.2 Ο αλγόριθμος SGD για συνεχή ροή δεδομένων.

Στο [34] περιγράφεται η παράλληλη υλοποίηση του αλγόριθμου SGD για συνεχή ροή δεδομένων. Για να επιτευχθεί η επεξεργασία των δεδομένων καθώς αυτά καταφθάνουν στο σύστημα, προσαρμόζεται ο συντονιστής έτσι ώστε να αναθέτει δυναμικά στους εργαζόμενους τους υποπίνακες και όχι βασιζόμενος σε προκαθορισμένα σύνολα υποπινάκων. Κάθε εργαζόμενος αποθηκεύει μόνο τα δεδομένα τα οποία θα επεξεργαστεί, και η ευθύνη του διαμοιρασμού ώστε να μή πραγματοποιούνται αντικρουόμενες ανανεώσεις, ανήκει αποκλειστικά στο συντονιστή. Σε αυτή την υλοποίηση δεν υπάρχει συνθήκη τερματισμού, αφού ο αλγόριθμος συνεχίζεται όσο εισέρχονται δεδομένα στο σύστημα συστάσεων.

Ο αλγόριθμος υλοποιήθηκε με το Hadoop MapReduce πλαίσιο παραλληλοποίησης και με το πλαίσιο Storm που πραγματοποιεί κατανεμημένους υπολογισμούς σε ροές δεδομένων. Μετρήθηκε το RMSE για το σύνολο δεδομένων της MovieLens.

5.5 Συνοπτική παρουσίαση των παράλληλων αλγορίθμων συνεργατικής διήθησης.

Παραπάνω περιγράφηκαν παράλληλες υλοποιήσεις αλγορίθμων συνεργατικής διήθησης. Στις υλοποιήσεις αυτές χρησιμοποιήθηκαν διαφορετικές τεχνολογίες για την παραλληλοποίηση και διαφορετικά σύνολα δεδομένων. Επίσης, η απόδοση των αλγορίθμων εκτιμάται με χρήση διαφορετικών μετρικών. Στον πίνακα 5.1 παρουσιάζονται οι τεχνολογίες και οι μετρικές που χρησιμοποιήθηκαν σε κάθε αλγόριθμο. Τα σύνολα δεδομένων που χρησιμοποιήθηκαν σε

κάθε υλοποίηση φαίνονται στον πίνακα 5.2. Αυτά που χρησιμοποιούνται περισσότερο είναι το σύνολο δεδομένων Netflix και το σύνολο δεδομένων MovieLens.

Αλγόριθμος	Τεχνολογία	Μετρικες
ALS weighted-λ- regularization	Matlab	RMSE,Χρόνος
ALS,wALS,SVD++, PMF,BPTF,SGD	GraphLab	RMSE,Επιτάχυνση
Concept Decomposition	Pthreads	RMSE,Χρόνος
Co-clustering 1	MPI,OpenMP	RMSE,Χρόνος
Item Based	Hadoop	Επιτάχυνση
User Based	Hadoop	Επιτάχυνση
Co-clustering 2	Pervasive DataRush, Hotspot JVM	RMSE
SGD data stream	Hadoop,Storm	RMSE
Co-clustering 3	MPI,C++	MAE
SGD batch mode	Hadoop	Κλιμάκωση,Επιτάχυνση

Πίνακας 5.1: Παράλληλοι αλγόριθμοι συνεργατικής διήθησης.

Αλγόριθμος	Σύνολο Δεδομένων
ALS weighted-λ- regularization	Netflix
ALS,wALS,SVD++, PMF,BPTF,SGD	Yahoo Music
Concept Decomposition	Netflix
Co-clustering 1	Netflix
Item Based	MovieLens
User Based	Netflix
Co-clustering 2	Netflix
SGD data stream	MovieLens
Co-clustering 3	MovieLens, Bookcrossing
SGD batch mode	Netflix

Πίνακας 5.2: Σύνολα δεδομένων που χρησιμοποιήθηκαν.

Κεφάλαιο 6

Ο Αλγόριθμος **Slope One**

Μία από τις πιό πολυχρησιμοποιημένες οικογένειες αλγορίθμων για συνεργατική διήθηση είναι οι slope one αλγόριθμοι. Οι αλγόριθμοι αυτοί χρησιμοποιούν μεθόδους συνεργατικής διήθησης για την παραγωγή συστάσεων, βασιζόμενοι στην αντικείμενο προς αντικείμενο προσέγγιση. Οι slope one αλγόριθμοι αποτέλεσαν πεδίο μελέτης και χρησιμοποιήθηκαν ώς μέσο για τη βελτίωση άλλων αλγορίθμων, όχι μόνο εξαιτίας της απλότητάς τους, αλλά καί της αποδοτικότητας και της ακρίβειάς τους, η οποία συγκρίνεται με αυτή πιό περίπλοκων και υπολογιστικά χρονοβόρων αλγορίθμων.

Τα βασικότερα πλεονεκτήματα των slope one μεθόδων είναι ότι η εφαρμογή τους είναι εύκολη, η ενημέρωση των δεδομένων τους είναι εφικτό να γίνεται δυναμικά, είναι αποτελεσαματικοί, παράγοντας γρήγορα συστάσεις, και δεν απαιτούν πολλά από τους νέους χρήστες του συστήματος συστάσεων. Ακόμα και ένας χρήστης με λίγες εκτιμήσεις μπορεί να λάβει έγκυρες συστάσεις. Επιπλέον, η απόδοσή τους είναι συγκρίσιμη με αυτή άλλων μεθόδων. Έτσι, καθίσταται απαραίτητη η περεταίρω μελέτη τους.

Σε αυτό το κεφάλαιο θα περιγραφεί ο αλγόριθμος slope one και οι βασικές υλοποιήσεις του. Θα γίνει ανασκόπηση των πρόσφατων υλοποιήσεών του και θα παρουσιασθούν και θα συγκριθούν δύο παράλληλες υλοποιήσεις του. Μία με χρήση του openMP και μιά με ταυτόχρονη χρήση openMP και MPI. Τέλος, θα παρουσιασθούν και θα σχολιασθούν τα αποτελέσματα των υλοποιήσεων αυτών.

6.1 Περιγραφή του αλγορίθμου και των διάφορων εκδοχών του.

Ο αλγόριθμος slope one ορίζει ανά ζεύγη πόσο πιό αρεστό είναι ένα αντικείμενο σε σχέση με ένα άλλο. Για να υπολογιστεί αυτή η διαφορά ένας τρόπος είναι να αφαιρεθεί η μέση εκτίμηση των δύο αντικειμένων και στη συνέχεια να χρησιμοποιηθεί για την πρόβλεψη των εκτιμήσεων ενός άλλου χρήστη. Για παράδειγμα, όπως φαίνεται στον πίνακα 6.1, δεδομένων δύο χρηστών A και B και δύο αντικειμένων i και j, όπου ο χρήστης A έχει εκτιμήσει καί τα δύο αντικείμενα ενώ ο χρήστης B μόνο το i, η πρόβλεψη της εκτίμησης του χρήστη B για το αντικείμενο j γίνεται υπολογίζοντας τη διαφορά των εκτιμήσεων για τα δύο αντικείμενα του χρήστη A, $dif_{A_{j-i}} = r_{A_j} - r_{A_i}$ και προσθέτοντάς την στην εκτίμηση του χρήστη B για

το αντικείμενο i. Οπότε, $pred(B, j) = r_{B_i} + dif_{A_{j-i}}$

Items / Users	...	i	...	j	...
⋮		⋮		⋮	
A	...	r_{A_i}	...	r_{A_j}	...
⋮		⋮		⋮	
B	...	r_{B_i}	...	?	...
⋮		⋮		⋮	

Πίνακας 6.1: Παράδειγμα για τον αλγόριθμο Slope One.

Έτσι, ο αλγόριθμος slope one για να παράγει πρόβλεψη της εκτίμησης ενός αντικειμένου από ένα χρήστη λαμβάνει υπόψην και πληροφορίες από άλλους χρήστες που έχουν εκτιμήσει το ίδιο αντικείμενο, όπως και η μέθοδος προσαρμοσμένου συνημιτόνου βάσει αντικειμένων, αλλά και τις υπόλοιπες εκτιμήσεις του χρήστη για άλλα αντικείμενα, όπως ο αλγόριθμος βάσει μέσης πρόβλεψης χρήστη. Στις προβλέψεις εισάγονται μόνο οι εκτιμήσεις των χρηστών που έχουν εκτιμήσει κάποια κοινά αντικείμενα με τον χρήστη για τον οποίο παράγεται η πρόβλεψη και μόνο οι εκτιμήσεις που έχει κάνει ο χρήστης αυτός.

Δεδομένου ενός συνόλου δεδομένων εκπαίδευσης χ και δύο αντικειμένων j και i με εκτιμήσεις αντίστοιχα u_j και u_i στο σύνολο των εκτιμήσεων ενός χρήστη $u \in S_{j,i}(\chi)$, η μέση απόκλιση της εκτίμησης του αντικειμένου i σε σχέση με την εκτίμηση του αντικειμένου j δίνεται από τον παρακάτω τύπο, στον οποίο $card(S_{j,i}(\chi))$ είναι το πλήθος των στοιχείων του συνόλου $S_{j,i}(\chi)$.

$$dev_{j,i} = \sum_{i \in S_{j,i}(\chi)} \frac{u_j - u_i}{card(S_{j,i}(\chi))}.$$

Με αυτόν τον τρόπο υπολογίζεται μόνο μία φορά ο συμμετρικός πίνακας αποκλίσεων, ο οποίος είναι πολύ εύκολο να ενημερωθεί με την εισαγωγή νέων δεδομένων. Εφόσον $dev_{j,i} + u_i$ είναι μια πρόβλεψη για το u_j δεδομένου του u_i , ένα καλό κριτήριο για την πρόβλεψη εκτιμήσεων είναι να χρησιμοποιηθεί η μέση τιμή όλων αυτών των προβλέψεων:

$$pred(u, j) = \frac{1}{card(R_j)} \sum_{i \in R_j} dev_{j,i},$$

όπου $R_j = \{i | j \in S(u), i \neq j, card(S_{j,i}(\chi)) > 0\}$ είναι το σύνολο όλων των σχετικών αντικειμένων. Η πρόβλεψη για αρκετά πυκνά σύνολα δεδομένων, στα οποία ισχύει ότι $card(S_{j,i}(\chi)) > 0$ για όλα σχεδόν τα i και j, δηλαδή το πλήθος των στοιχείων του συνόλου S των εκτιμήσεων χρηστών που περιέχουν ταυτόχρονα εκτιμήσεις καί για τα δύο αντικείμενα i και j, για όλα σχεδόν τα i και j απλοποιείται στον παρακάτω τύπο :

$$pred(u, j) = \bar{u} + \frac{1}{card(R_j)} \sum_{i \in R_j} dev_{j,i}.$$

Η πρόβλεψη αυτή δεν εξαρτάται από το πώς ο χρήστης έχει εκτιμήσει διάφορα αντικείμενα, αλλά από τη μέση εκτίμηση του χρήστη και από τα αντικείμενα τα οποία έχει εκτιμήσει.

Μία άλλη υλοποίηση αυτού του αλγόριθμου γίνεται λαμβάνοντας υπόψην, εκτός από τα αντικείμενα που έχει εκτιμήσει ο χρήστης, και το πλήθος των παρατηρούμενων εκτιμήσεων για κάθε αντικείμενο $c_{j,i} = card(S_{j,i}(\chi))$,οπότε η πρόβλεψη για την weighted slope one μορφή δίνεται από τον εξής τύπο:

$$pred(u, j) = \frac{\sum_{i \in S(u)-\{j\}} (dev_{j,i} + u_i) c_{j,i}}{\sum_{i \in S(u)-\{j\}} c_{j,i}}$$

Έτσι, αν ένα ζεύγος αντικειμένων έχει εκτιμηθεί από περισσότερους χρήστες, τότε θα έχει και πιό σημαντικό ρόλο στις προβλέψεις από ένα ζεύγος αντικειμένων που έχει εκτιμηθεί από λιγότερους χρήστες.

Επιπλέον, είναι δυνατό να διαχωριστεί το σύνολο των εκτιμήσεων ενός χρήστη σε αυτές για τα αντικείμενα για τα οποία ο χρηστης έχει δώσει εκτίμηση μεγαλύτερη από μια συγκεκριμένη τιμή, δηλαδή ένα σύνολο με τις εκτιμήσεις για τα αντικείμενα που αρέσουν στο χρήστη, και ένα για τις υπόλοιπες εκτιμήσεις για τα αντικείμενα που δεν του αρέσουν. Έτσι ορίζεται το δίπολο (bi-polar) σχήμα του αλγορίθμου, το οποίο όμως προβλέπει μόνο αν ένα αντικείμενο θα αρέσει ή όχι στο χρήστη, χωρίς να δίνει ένδειξη για το πόσο θα του αρέσει.

Οι προβλέψεις χωρίζονται σε δύο μέρη και χρησιμοποιείται ώς κατώφλι ένας αριθμός, όπως το μέσο της βαθμολογικής κλίμακας των αξιολογήσεων. Τα αντικείμενα που η αξιολόγησή τους είναι μεγαλύτερη από το κατώφλι χαρακτηρίζονται ώς αρεστά, ενώ τα υπόλοιπα ώς μή αρεστά. Η μέθοδος αυτή θα παρήγαγε καλά αποτελέσματα αν οι αξιολογήσεις των χρηστών ήταν κατανεμημένες ομοιόμορφα στο εύρος της κλίμακας των αξιολογήσεων. Για να μπορούν να υποστηριχθούν όλες οι κατηγορίες χρηστών, εφαρμόζεται σαν κατώφλι η μέση τιμή των αξιολογήσεων κάθε χρήστη.

Ο διαχωρισμός του κάθε χρήστη σε δύο, σύμφωνα με τις προτιμήσεις του, επιφέρει το διπλασιασμό του συνόλου των χρηστών του συστήματος. Ωστόσο, οι περιορισμοί που εφαρμόζονται από το δίπολο σχήμα, μειώνουν το συνολικό πλήθος των εκτιμήσεων που χρειάζεται να συμπεριληφθούν στον υπολογισμό των προβλέψεων [27].

6.2 Ψευδοκώδικας και χρόνος εκτέλεσης.

Algorithm 1 Ψευδοκώδικας του Slope One.

Require: $R = \{r_{m_n}\}$ { The set of ratings}
  for $k = 1 \rightarrow m$ **do**
    for $i = 1 \rightarrow n$ **do**
      for $j = i + 1 \rightarrow n$ **do**
        $dif_{j,i} \leftarrow r_j - r_i$
        $count_{j,i} \leftarrow$ {number of users who rated both items}
        $dev_{j,i} \leftarrow \sum \frac{dif_{j,i}}{count_{j,i}}$
      end for
    end for
  end for
  return P Prediction

Παρουσιάζεται εδώ ο ψευδοκώδικας, για n αντικείμενα, m χρήστες και N εκτιμήσεις. Για να ξεκινήσει ο αλγόριθμος προαπαιτείται η τροφοδότησή του με τον πίνακα R, ο οποίος περιέχει τις αξιολογήσεις που έχουν δώσει οι χρήστες για τα αντικείμενα του συστήματος συστάσεων. Για κάθε χρήστη και για κάθε αντικείμενο, υπολογίζει για όλα τα υπόλοιπα αντικείμενα τη διαφορά των αξιολογήσεών τους, μετράει πόσοι είναι οι χρήστες που έχουν αξιολογήσει και τα δύο αντικείμενα, και υπολογίζει τη μέση απόκλιση των διαφορών των αξιολογήσεών τους. Στο τέλος, χρησιμοποιεί όσα έχει υπολογίσει για να παράγει την πρόβλεψη για την αξιολόγηση του ζητούμενου αντικειμένου από συγκεκριμένο χρήστη.

Ο υπολογισμός των διαφορών των μέσων εκτιμήσεων για κάθε ζεύγος αντικειμένων απαιτεί το πολύ mn^2 βήματα. Η ανανέωση των αποτελεσμάτων όταν ένας χρήστης έχει εισάγει χ εκτιμήσεις και εισάγει μία νέα απαιτεί χ βήματα. Ο χρόνος εκτέλεσης του αλγορίθμου είναι λοιπόν mn^2 στη χειρότερη περίπτωση [56].

6.3 Ανασκόπηση σχετικών εργασιών.

Τα τελευταία χρόνια παρατηρείται έντονα η χρήση της οικογένειας των αλγορίθμων slope one σε συνδυασμό με άλλες μεθόδους, καθώς και με τεχνικές εξόρυξης δεδομένων, με σκοπό την επίτευξη γρηγορότερων και αποδοτικότερων συστημάτων συστάσεων.

Ο D.Zhang [62] παρουσίασε το 2009 μια μέθοδο που χρησιμοποιεί slope one για να παράγει τις προβλέψεις των εκτιμήσεων αντικειμένων ενός χρήστη, και στη συνέχεια χρησιμοποιεί την μετρική εγγύτητας Pearson correlation για να υπολογίσει τα γειτονικά αντικείμενα του αντικειμένου για το οποίο έγινε η πρόβλεψη και τέλος παράγει τις συστάσεις. Η ποιότητα των προβλέψεων σε αυτή τη μέθοδο επηρρεάζεται από το μέγεθος του συνόλου των γειτονικών αντικειμένων. Ο αλγόριθμος αυτός είναι πιό ακριβής από τον παραδοσιακό αλγόριθμο συνεργατικής διήθησης.

Μία άλλη μέθοδος, χωρίζει πρωτα το σύνολο των αντικειμένων σε υποσύνολα, βασιζόμενη στο είδος αντικειμένων που έχει ζητήσει ο χρήστης. Με αυτόν τον τρόπο μειώνει τη διάσταση του συνόλου των αντικειμένων, των εκτιμήσεων και πολλές φορές και των χρηστών. Η συσταδοποίηση των δεδομένων επιτυγχάνεται με τον αλγόριθμο K-means, λαμβάνοντας σαν παραμέτρους δημογραφικά στοιχεία, όπως η ηλικία του χρήστη και το είδος του αντικειμένου, το οποίο πρέπει ο χρήστης να έχει προσδιορίσει. Στη συνέχεια χρησιμοποιείται ο slope one αλγόριθμος στο μικρότερο σύνολο δεδομένων για να παράγει τις προβλέψεις. Έτσι μειώνεται ο χρόνος υπολογισμού και αυξάνεται η ακρίβεια των προβλέψεων, αφού τα δεδομένα είναι λιγότερα και από αυτά έχουν αφαιρεθεί τα αντικείμενα μή συναφούς περιεχομένου [35].

Ο αλγόριθμος slope one συνδυάστηκε με τον αλγόριθμο userrank, ο οποίος βασίζεται στον Pagerank και δίνει βάρη σε κάθε χρήστη, ανάλογα με το πόσα συσχετιζόμενα αντικείμενα έχει. Τα βάρη αυτά χρησιμοποιούνται στον υπολογισμό των διαφορών. Στο [32] αποδεικνύεται ότι η προσεγγιση στον slope one ενσωματώνοντας τον userrank βελτιώνει τα αποτελέσματα των συστάσεων. Ένας ακόμα αλγόριθμος που συνδυάζει την συνεργατική διήθηση βάσει αντικειμένων με τη συνεργατική διήθηση βάσει χρηστών προτάθηκε το 2009 και χρησιμοποιεί τον slope one για να γεμίσει τις κενές θεσεις του πίνακα των εκτιμήσεων και στη συνέχεια στον πυκνό πια πίνακα εφαρμόζει τεχνικές συνεργατικής διήθησης βάσει χρηστών [38]. Ο αλγόριθμος slope one έχει επιλεχθεί και από τους Zilei Sun et al [65] γιατί είναι πιό αποδοτικός στον υπολογισμό της εγγύτητας μεταξύ αντικειμένων από άλλους αλγόριθμους

που βασίζονται στα αντικείμενα για να παράγουν προβλέψεις.

6.4 Μελέτη ακολουθιακού κώδικα του αλγόριθμου slope one.

Στη συνέχεια παρατίθενται και σχολιάζονται αποσπάσματα του ακολουθιακού κώδικα που υλοποιεί ένα σύστημα παραγωγής προβλέψεων χρησιμοποιώντας τον αλγόριθμο slope one. Στην αρχή του προγράμματος δεσμεύεται χώρος στη μνήμη για τους πίνακες ratings, numerator, denominator και deviation.

Ο πίνακας ratings είναι διαστάσεων users ×ITEMS, δηλαδή όσοι είναι οι χρήστες του συστήματος επί το πλήθος των αντικειμένων του συστήματος, και περιέχει τις αξιολογήσεις των χρηστών για τα αντικείμενα. Κάθε σειρά του πίνακα αυτού αντιστοιχεί σε ένα χρήστη, και κάθε στήλη σε ένα αντικείμενο. Στις θέσεις που αντιστοιχούν σε αντικείμενα που οι χρήστες δεν έχουν εκτιμήσει ο πίνακας έχει την τιμή 0.

Ο πίνακας numerator είναι διαστάσεων users ×ITEMS ×ITEMS και τοποθετούνται σε αυτόν οι διαφορές των εκτιμήσεων των αντικειμένων που έχει αξιολογήσει κάθε χρήστης. Για κάθε χρήστη, ο πίνακας που σχηματίζεται είναι αντισυμμετρικός, διαστάσεων ITEMS ×ITEMS, και στην κύρια διαγώνιό του έχει 0.

Ο πίνακας denominator είναι ένας πίνακας συχνοτήτων, διαστάσεων ITEMS ×ITEMS. Σε αυτόν αποθηκεύεται η συχνότητα με την οποία τα αντικείμενα αξιολογούνται ταυτόχρονα. Κάθε φορά που ένας χρήστης έχει εκτιμήσει τα αντικείμενα i και j, η τιμή της θέσης της i γραμμής και της j στήλης του πίνακα αυξάνεται κατά ένα. Ο πίνακας αυτός είναι συμμετρικός και τα στοιχεία της κύριας διαγωνίου του εκφράζουν το πλήθος των εκτιμήσεων που έχουν εισαχθεί στο σύστημα για το κάθε αντικείμενο.

Ο πίνακας deviation είναι και αυτός ίσων διαστάσεων με τον denominator και περιέχει στη θέση της i γραμμής και της j στήλης του, τη μέση τιμή της διαφοράς των εκτιμήσεων των αντικειμένων i και j. Στην πραγματικότητα για όλους τους παραπάνω πίνακες δεσμεύεται χώρος στη μνήμη με την εντολή calloc και γίνεται αναφορά στις διάφορες θέσεις μνήμης με χρήση κατάλληλων δεικτών.

Στον κώδικα αυτό χρησιμοποιείται και ο πίνακας buffer ο οποίος είναι δύο διαστάσεων. Η μία διάσταση είναι ίση με το πλήθος των εγγραφών του αρχείου των εκτιμήσεων των χρηστών, και η άλλη είναι ίση με τρία, για να αποθηκεύεται ο χρήστης, το αντικείμενο και η τιμή της αξιολόγησής του απο τον χρήστη. Στη συνέχεια αυτός ο πίνακας χρησιμοποιείται για το σχηματισμό του πίνακα ratings. Στο πλαίσιο 6.1 φαίνεται ο τρόπος με τον οποίο δημιουργείται ο πίνακας ratings. Η μεταβλητή testuser χρησιμοποιείται για τον έλεγχο αλλαγής χρήστη. Κάθε φορά που αλλάζει ο χρήστης ρυθμίζονται εκ νέου οι μεταβλητές i, j και k, ώστε να τοποθετούνται οι σωστές τιμές στον πίνακα ratings.

```
/* ###### Set Ratings ###### */

int testuser;
testuser=buffer[0][0];
int k=0;
```

```
int w=0;
int m=0;
for(i=0;i<records;i++) {
  if(buffer[i][0]==testuser){
      if(buffer[i][1]==j) {
        ratings[k]=buffer[i][2];
        }
      else {
        ratings[k]=0;
        i--;
        }
      j++;
    }
  else {
    testuser++;
    i--;
    j=1;
    k--;
    }
  k++;
}
```

Listing 6.1: Υπολογισμός του πίνακα ratings στον ακολουθιακό κώδικα.

Οι υπολογισμοί που μπορούν να πραγματοποιηθούν offline ολοκληρώνονται με τον υπολογισμό των πινάκων numerator, denominator και deviation, που απεικονίζεται στο πλαίσιο 6.2. Για τον πίνακα numerator υπολογίζονται πρώτα τα στοιχεία της κύριας διαγωνίου του υποπίνακα που αντιστοιχεί σε κάθε χρήστη. Στην περίπτωση που ο χρήστης δεν έχει δώσει εκτίμηση για κάποιο αντικείμενο, στην αντίστοιχη θέση του πίνακα numerator τοποθετείται η τιμή 10. Η τιμή αυτή είναι αδύνατο να προκύψει από τις διαφορές των εκτιμήσεων δύο αντικειμένων. Έτσι θα μπορεί αργότερα να γίνεται διάκριση μεταξύ των περιπτώσεων που ο χρήστης δεν έχει δώσει αξιολόγηση και των περιπτώσεων που η διαφορά μεταξύ των αξιολογήσεων δύο αντικειμένων είναι 0.

Στις υπόλοιπες περιπτώσεις, που ο χρήστης έχει δώσει εκτίμηση για κάποιο αντικείμενο, στην κύρια διαγώνιο του πίνακα numerator εισάγεται η τιμή 0. Υπολογίζονται στη συνέχεια και οι υπόλοιπες διαφορές, τοποθετώντας πάντοτε την τιμή 10 στις περιπτώσεις όπου δέν έχει παραχωρηθεί αξιολόγηση για κάποιο αντικείμενο. Στη συνέχεια, για τις θέσεις του πίνακα numerator όπου η τιμή είναι διάφορη του 10, υπολογίζεται ο πίνακας deviation, όπου στη θέση της σειράς i και της στήλης j τοποθετείται το άθροισμα των διαφορών των αξιολογήσεων των αντικειμένων i και j. Έπειτα υπολογίζονται ο πίνακας συχνοτήτων denominator και ο τελικός πίνακας deviation. Ο πίνακας deviation λαμβάνει την τιμή 0 στις ίδιες θέσεις στις οποίες ο πίνακας denominator έχει 0, και στις υπόλοιπες θέσεις λαμβάνει το αποτέλεσμα του λόγου της τιμής που είχε υπολογισθεί προηγουμένως για τον πίνακα deviation προς την τιμή που περιέχει ο πίνακας denominator.

```
/* ###### Set numerator(differences matrix) - denominator(frequencies
   matrix) ###### */
/* ###### Set deviation ###### */
k=0;
int l=0;
testuser=buffer[0][0];
int x=testuser-2;
```

```
for(testuser=buffer[0][0];testuser<=users;testuser++) {
  w=0;
  m=0;
  for(i=1;i<=ITEMS;i++) {
    for(j=1;j<=ITEMS;j++) {
    if(i==j){
      if( (ratings[x+i]==0) || (ratings[x+j]==0) ) {
      numerator[k]=10;
      }
      else {
      numerator[k]=0;
      }
      k++;
      }
    else {
      if( (ratings[x+i]==0) || (ratings[x+j]==0) ) {
      numerator[k]=10;
      }
      else {
      numerator[k]=ratings[x+i]-ratings[x+j];
      }
      k++;
      }
      if(numerator[l]!=10){
      deviation[m]=deviation[m]+numerator[l];
      }
      m++;
      l++;
    }
  if( ratings[x+i]!=0 ) {
        w=w+i-1;
        denominator[w]=denominator[w]+1;
        w++;
        for(j=i+1;j<=ITEMS;j++) {
          if( ratings[x+j]!=0 ) {
            denominator[w]=denominator[w]+1;
          }
        w++;
        }
      }
      else if (ratings[x+i]==0){
        w=w+i-1;
        w++;
        for(j=i+1;j<=ITEMS;j++) {
        w++;
        }
      }
  }
x=x+ITEMS;
}
// calculate symmetric values of denominator
int z=0;
for(i=1;i<ITEMS;i++){
  z=i*ITEMS;
  denominator[z]=denominator[i];
  for(k=0;k<=i-1;k++){
    z++;
    denominator[z]=denominator[z-(i-1)*(ITEMS-1)+(k*(ITEMS-1))];
```

```
  }
}
/* ###### Calculate Deviation ###### */

k=0;
  for(i=1;i<=ITEMS;i++) {
    for(j=1;j<=ITEMS;j++) {
        //calculate deviation
        if(denominator[k]==0){
        deviation[k]=0;
        }
        else{
        deviation[k]=deviation[k]/denominator[k];
        }
        k++;
        }
      }
```

Listing 6.2: Offline υπολογισμοί στον ακολουθιακό κώδικα.

Οι προβλέψεις και οι προβλέψεις με χρήση βαρών παράγονται με κλήση των αντίστοιχων συναρτήσεων. Η συνάρτηση για τον υπολογισμό των προβλέψεων φαίνεται στο πλαίσιο 6.3 και η συνάρτηση για τον υπολογισμό των προβλέψεων με βάρη στο 6.4.

```
/* ###### Function Predictions ###### */

float predictions(int *ratings,int *denominator,float *deviation,int
   user,int item){

float prediction;
float u;
int count=0;
float sum=0;
int i=0;
int cardRi=0;
float Sdev=0;
int item2;
for(i=((user*ITEMS)-ITEMS);i<user*ITEMS;i++){
  if(ratings[i]!=0){
    count++;
    sum=sum+ratings[i];
    u=sum/count;
    }
  }
if(ratings[(user*ITEMS)+item-(ITEMS+1)]==0){
  cardRi=count;
  }
  else{
  cardRi=count-1;
  }
for(item2=1;item2<=ITEMS;item2++){
  if(item2!=item && ratings[user*ITEMS+item2-(ITEMS+1)]!=0 &&
     denominator[(item-1)*ITEMS+item2-1]!=0){
      Sdev=Sdev+deviation[(item-1)*ITEMS+item2-1];
    }
  }
if(cardRi==0 || Sdev==0){
```

```
printf("Prediction␣not␣available␣for␣user␣%i␣and␣item␣%i␣\n",user,item);
}
else{
prediction=u+Sdev/cardRi;
}
return prediction;
}
```

Listing 6.3: Συνάρτηση υπολογισμού των προβλέψεων .

```
/* ###### Function Weighted Predictions ###### */

float weightedpredictions(int *ratings,int *denominator,float *deviation
    ,int user,int item){

float prediction;
float S1=0;
int S2=0;
int item2;
for(item2=1;item2<=ITEMS;item2++){
  if(item2!=item && ratings[user*ITEMS+item2-(ITEMS+1)]!=0 ){
      S1=S1+(deviation[(item-1)*ITEMS+item2-1]+ratings[user*ITEMS+item2
         -(ITEMS+1)])*denominator[(item-1)*ITEMS+item2-1];
      S2=S2+denominator[(item-1)*ITEMS+item2-1];
    }
  }
if(S2==0){
printf("Prediction␣not␣available␣for␣user␣%i␣and␣item␣%i␣\n",user,item);
}
else{
prediction=S1/S2;
}
return prediction;
}
```

Listing 6.4: Συνάρτηση υπολογισμού των προβλέψεων με χρήση βαρών .

Στη συνάρτηση για την παραγωγή των προβλέψεων υπολογίζεται πρώτα η μέση τιμή των εκτιμήσεων του χρήστη. Στη συνέχεια υπολογίζεται το πλήθος των αντικειμένων του συνόλου R_i. Αν το αντικείμενο για το οποίο γίνεται η πρόβλεψη έχει τιμή αξιολόγησης ίση με μηδέν, δηλαδή αν δεν έχει αξιολόγηση, τότε το πλήθος των αντικειμένων του συνόλου R_i είναι ίσο με την τιμή της μεταβλητής count που υπολογίσθηκε κατά τον υπολογισμό της μέσης εκτίμησης του χρήστη. Αν έχει τιμή αξιολόγησης διαφορετική του μηδενός, τότε είναι ίσο με την τιμή της μεταβλητής count μείον ένα, έτσι ώστε να μη συνυπολογίζεται το αντικείμενο αυτό στο πλήθος των αντικειμένων του συνόλου R_i, ενώ προηγουμένως, στη μεταβλητή count είχε υπολογισθεί και αυτό, αφού η τιμή της αξιολόγησής του είναι διαφορετική του μηδενός.

Η μείωση της μεταβλητής count κατά ένα είναι φαινομενικά ένας άσκοπος υπολογισμός, αφού δεν χρειάζεται να γίνει πρόβλεψη για την αξιολόγηση ενός αντικειμένου αν ήδη ο χρήστης το έχει εκτιμήσει. Στην πραγματικότητα όμως έχει συμπεριληφθεί στον κώδικα ώς ένα μέτρο για να μπορεί να υπολογισθεί η πρόβλεψη της εκτίμησης ενός αντικειμένου ακόμα και αν το αντικείμενο έχει ήδη εκτιμηθεί, ώστε να μπορεί να εφαρμοσθεί έλεγχος της απόδοσης του αλγορίθμου αν κριθεί απαραίτητο.

Έπειτα υπολογίζεται το άθροισμα των αποκλίσεων των εκτιμήσεων του αντικειμένου για το οποίο παράγεται η πρόβλεψη από τα υπόλοιπα αντικείμενα που έχει εκτιμήσει ο χρήστης, και στο τέλος παράγεται η πρόβλεψη για το ζητούμενο αντικείμενο.

Για την παραγωγή των προβλέψεων με χρήση βαρών, για όλα τα αντικείμενα που έχει αξιολογήσει ο χρήστης, αθροίζεται η αξιολόγηση που τους έχει δώσει με την τιμή της κατάλληλλης θέσης του πίνακα deviation, και πολλαπλασιάζεται με το πλήθος των χρηστών που έχουν αξιολογήσει και τα δύο αντικείμενα $c_{j,i}$, δηλαδή με την τιμή του πίνακα deviation. Το άθροισμα αυτό διαιρείται με το άθροισμα όλων των επιμέρους $c_{j,i}$ και το αποτέλεσμα είναι η πρόβλεψη για το αντικείμενο j (μεταβλητή item).

Ο κώδικας αυτός έχει μεγάλες απαιτήσεις σε μνήμη, αφού συνολικά χρειάζεται $users*items+items^2(users+2)+(records*3)$ θέσεις μνήμης. Παρά το γεγονός ότι ο ακολουθιακός κώδικας μπορεί να βελτιωθεί περαιτέρω, το νούμερο αυτό είναι απαγορευτικό για την υλοποίησή του σε πραγματικά σύνολα δεδομένων που αποτελούνται από εκατομμύρια εγγραφές και χρήστες, παρά τη μεγάλη διαθεσιμότητα υπολογιστικών πόρων. Συνεπώς, η ανάγκη επίλυσης του προβλήματος μνήμης και η ανάγκη παραγωγής προβλέψεων σε πραγματικό χρόνο ώθησαν στην παραλληλοποίησή του.

6.5 Πειραματική μελέτη παράλληλων εφαρμογών του αλγόριθμου slope one.

6.5.1 Εφαρμογή με χρήση της διεπαφής OpenMp.

Σε αυτή την ενότητα παρουσιάζεται και αναλύεται η υλοποίηση του slope one αλγορίθμου με χρήση της διεπαφής OpenMP. Πρώτα περιγράφεται το πείραμα και ο κώδικας της υλοποίησης, σε γλώσσα C και στη συνέχεια παρουσιάζονται τα αποτελέσματά του.

6.5.1.1 Περιγραφή του πειράματος.

Το σύστημα εκτέλεσης του πειράματος περιέχει δύο επεξεργαστές AMD opteron (tm) Processor 6128 HE όπου κάθε ένας περιέχει οκτώ πυρήνες. Η συχνότητα ρολογιού είναι στα 800 Mhz και το μέγεθος της cache μνήμης είναι 512 KB. Η μνήμη RAM του συστήματος είναι χωρητικότητος 16 GB. Το λειτουργικό σύστημα, στο οποίο εκτελέστηκαν τα πειράματα, είναι το Ubuntu Linux 10.04. Οι εφαρμογές υλοποιήθηκαν με γλώσσα προγραμματισμού τη C και χρησιμοποιήθηκε η έκδοση 3.0 της διεπαφής OpenMP. Για τη μέτρηση του χρόνου, χρησιμοποιήθηκε η συνάρτηση omp_get_wtime(), του OpenMP αντί της συνάρτησης clock() της C, λόγω της καλύτερης ανάλυσης χρόνου που προσφέρει.

Για τα αποτελέσματα επιτάχυνσης (speed up) έγιναν μετρήσεις για το σύνολο δεδομένων u.data της MovieLens, που περιλαμβάνει 943 χρήστες, 1682 αντικείμενα και 100000 εκτιμήσεις, με χρήση ενός, τεσσάρων, οκτώ, δώδεκα και δεκαέξι νημάτων. Ενώ για τα αποτελέσματα κλιμάκωσης (scale up) έγιναν μετρήσεις σε διάφορα σύνολα δεδομένων της MovieLens, όπως αυτά περιγράφονται στο παράρτημα Α΄

Η εφαρμογή παραλληλοποίησης του αλγορίθμου slope one πραγματοποιήθηκε σε υπολογι-

στές μοιραζόμενης μνήμης, ακολουθώντας το μοντέλο νημάτων, με χρήση του OpenMP. Τα χρονοβόρα τμήματα του ακολουθιακού κώδικα είναι ο υπολογισμός των πινάκων ratings, numerator, denominator και deviation. Επιλέχθηκε να γίνει επιμερισμός των δεδομένων μεταξύ των νημάτων, και όχι των λειτουργιών, διότι σε πολλά στάδια των υπολογισμών μερικών από τους παραπάνω πίνακες απαιτείται η χρήση των υπολοίπων πινάκων. Έτσι, ο επιμερισμός των λειτουργιών, δηλαδή ο υπολογισμός του κάθε πίνακα από διαφορετικό νήμα, θα οδηγούσε σε σχεδόν σειριοποίηση της εκτέλεσης του προγράμματος. Ο έλεγχος πρόσβασης στη μοιραζόμενη μνήμη επιτυγχάνεται μόνο με το χαρακτηρισμό μεταβλητών ώς ιδιωτικές, χωρίς χρήση κλειδαριών.

Στη συνέχεια περιγράφονται τα παράλληλοποιημένα σημεία του προγράμματος. Δημιουργείται πρώτα ο πίνακας ratings,που περιλαμβάνει τις εκτιμήσεις κάθε χρήστη. Σε κάθε σειρά του πίνακα περιέχονται οι εκτιμήσεις ενός χρήστη, και στις θέσεις που αντιστοιχούν στα αντικείμενα για τα οποία ο χρήστης δεν έχει δώσει εκτιμήσεις καταχωρείται η τιμή 0. Έτσι ο πίνακας αυτός είναι πολύ αραιός. Στο πλαίσιο 6.5 φαίνεται ο παραλληλοποιημένος κώδικας σχηματισμού του πίνακα αυτού. Χρησιμοποιείται η οδηγία parallel που ορίζει μια παράλληλη περιοχή του κώδικα, στην οποία οι μεταβλητές i, j, k και $testuser$ είναι ιδιωτικές, και η οδηγία for για το διαμοιρασμό των επαναλήψεων του βρόγχου.

```
#pragma omp parallel private(i,j,k,testuser)
{
#pragma omp for
for(i=0;i<records;i++) {
  testuser=buffer[i][0];
  j=buffer[i][1];
  k=(testuser*ITEMS)-ITEMS+j-1;
  ratings[k]=buffer[i][2];
  }
} /* end of parallel section */
```

Listing 6.5: Δημιουργία του πίνακα ratings.

Στο πλαίσιο 6.6 υπολογίζονται οι πίνακες numerator, denominator και ο αρχικός πίνακας deviation. Ο numerator είναι ο πίνακας των διαφορών μεταξύ των εκτιμήσεων των αντικειμένων και ο deviation περιέχει το άθροισμα των διαφορών αυτών για κάθε ζεύγος αντικειμένων. Για να αποφευχθούν προβλήματα χώρου ο πίνακας numerator έγινε ίσων διαστάσεων με τον πίνακα deviation και μηδενίζεται πρίν από κάθε αλλαγή χρήστη, έτσι ώστε να υπολογισθούν οι διαφορές χωρίς να χρειάζεται χώρος αποθήκευσης ίσος με Users ×Items ×Items. Με αυτό τον τρόπο υπερνικούνται σε μεγάλο βαθμό τα προβλήματα μνήμης που παρουσιάζει ο ακολουθιακός κώδικας, και η υλοποίηση αυτή μπορεί να εφαρμοσθεί σε μεγαλύτερα σύνολα δεδομένων. Ο πίνακας denominator είναι ένας πίνακας συχνοτήτων μεγέθους Items ×Items στον οποίο αποθηκεύεται πόσες φορές συναντάται εκτίμηση για τα αντικείμενα i και j ταυτόχρονα.

Στο πλαίσιο 6.6 ορίζεται η παράλληλη περιοχή με ιδιωτικές μεταβλητές τις $i, j, k, testuser$, x, w, m και z και τον πίνακα numerator δηλωμένο ώς firstprivate, ώστε να συνδυάζεται η λειτουργία της φράσης private με τη δυνατότητα αρχικοποίησης του πίνακα κατά την είσοδο στην παράλληλη περιοχή. Η παράλληλη περιοχή που ορίζεται εδώ, ισχύει μέχρι το τέλος του πλαισίου 6.7. Στο πλαίσιο 6.6 χρησιμοποιείται η οδηγία for που διαμοιράζει την εργασία, αναθέτοντας διαφορετικούς χρήστες σε κάθε νήμα. Έτσι, κάθε νήμα υπολογίζει τον πίνακα numerator για κάθε ένα χρήστη από αυτούς που του αναλογούν, τα αντίστοιχα μέρη του

πίνακα denominator, και αρχικοποιεί κατάλληλα τον πίνακα deviation.

```
#pragma omp parallel private(i,j,k,testuser,x,w,m,z) firstprivate(
   numerator)
{
#pragma omp for
for(testuser=1;testuser<=users;testuser++) {
x=(testuser-1)*ITEMS;
  k=0;
  w=0;
  m=0;
  for(i=1;i<=ITEMS;i++) {
    for(j=1;j<=ITEMS;j++) {
    if(i==j){
      if( (ratings[x+i-1]==0) || (ratings[x+j-1]==0) ) {
      numerator[k]=10;
      }
      else {
      numerator[k]=0;
      }
      k++;
      }
    else {
      if( (ratings[x+i-1]==0) || (ratings[x+j-1]==0) ) {
      numerator[k]=10;
      }
      else {
      numerator[k]=ratings[x+i-1]-ratings[x+j-1];
      }
      k++;
      }
      //calculate deviation's numerator
      if(numerator[k-1]!=10){
      deviation[m]=deviation[m]+numerator[k-1];
      }
      m++;

    } //end for j
  if( ratings[x+i-1]!=0 ) {
        w=w+i-1;
        denominator[w]=denominator[w]+1;
        w++;
        for(j=i+1;j<=ITEMS;j++) {
          if( ratings[x+j-1]!=0 ) {
            denominator[w]=denominator[w]+1;
          }
        w++;
        }
      }
      else if (ratings[x+i-1]==0){
        w=w+i-1;
        w++;
        for(j=i+1;j<=ITEMS;j++) {
        w++;
        }
      }
  }
```

```
  for(z=0;z<ITEMS*ITEMS;z++)
  {
    numerator[z]=0;
  }
}
```

Listing 6.6: Δημιουργία των πινάκων numerator,denominator, και deviation.

Οι συμμετρικές τιμές του πίνακα denominator υπολογίζονται στο πλαίσιο 6.7 με τη χρήση μιας νέας οδηγίας for.

```
#pragma omp for
for(i=1;i<ITEMS;i++){
  z=i*ITEMS;
  denominator[z]=denominator[i];
  for(k=0;k<=i-1;k++){
    z++;
    denominator[z]=denominator[z-(i-1)*(ITEMS-1)+(k*(ITEMS-1))];
  }
}
```

Listing 6.7: Συμμετρικές τιμές του πίνακα denominator.

Ο τελικός πίνακας deviation στον οποίο περιέχονται οι διαφορές μεταξύ των αντικειμένων, διαιρεμένες με την αντίστοιχη τιμή του πίνακα συχνοτήτων denominator, επίσης υπολογίζεται με μία ξεχωριστή οδηγία for (Πλαίσιο 6.8).

```
#pragma omp for
  for(i=1;i<=ITEMS;i++) {
  k=(i-1)*ITEMS;
    for(j=1;j<=ITEMS;j++) {
        //calculate deviation
        if(denominator[k]==0){
        deviation[k]=0;
        }
        else{
        deviation[k]=deviation[k]/denominator[k];
        }
        k++;
        }
      }
```

Listing 6.8: Ο πίνακας deviation.

Οι προβλέψεις και οι προβλέψεις με χρήση βαρών γίνονται επίσης παράλληλα. Η αντίστοιχη συνάρτηση predictions ή weighted predictions καλείται, και το αποτέλεσμά της εγγράφεται σε αρχείο ταυτόχρονα από διάφορα νήματα. Κάθε ένα από τα νήματα αναλαμβάνει να καλέσει τις συναρτήσεις για να προβλέψει τις εκτιμήσεις διαφορετικών χρηστών. Παρατίθεται ο κώδικας για την κλήση της συνάρτησης Predictions (Πλαίσιο 6.9). Παρόμοιος είναι και ο κώδικας για την κλήση της συνάρτησης Weighted predictions. Ορίζεται μια παράλληλη περιοχή με ιδιωτικές μεταβλητές i, $user$ και $item$. Κάθε νήμα αναλαμβάνει να παράγει προβλέψεις για όλα τα αντικείμενα που δεν έχουν αξιολογηθεί, εξετάζοντας το κομμάτι του πίνακα ratings

που του αναλογεί, και πραγματοποιεί εγγραφές των αποτελεσμάτων του σε ένα αρχείο.

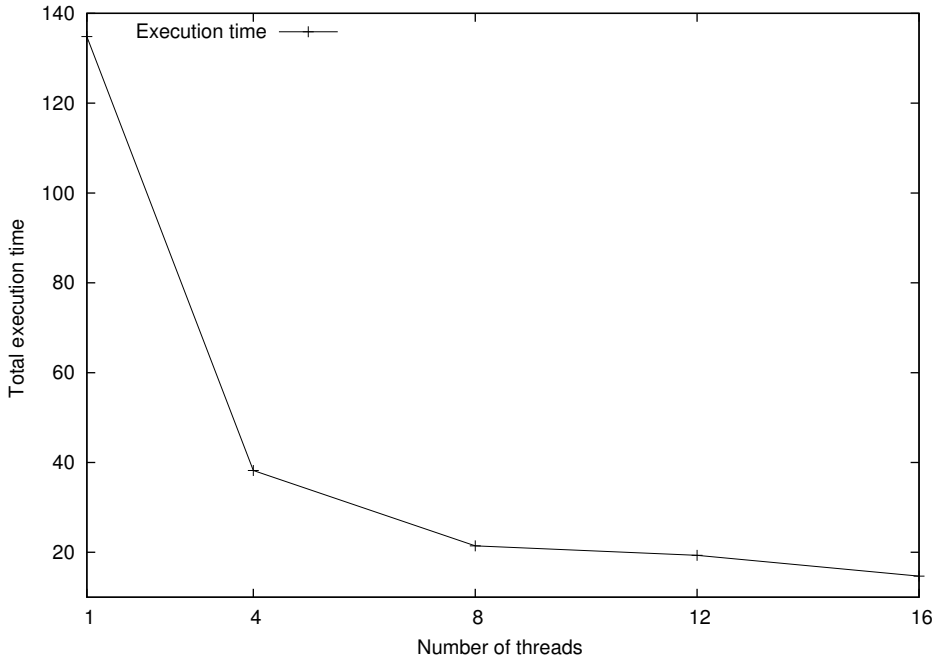
```
#pragma omp parallel private(i,user,item)
{
#pragma omp for
for(i=0;i<users*ITEMS;i++)
{
  if(ratings[i]==0)
  {
    user=i/ITEMS+1;
    item=i%ITEMS+1;
    pred=predictions(ratings,denominator,deviation,user,item);
//     fprintf(fp,"%d\t%d\t%f\n",user,item,pred);
  }
}
} /* end of parallel section */
```

Listing 6.9: Κλήση της συνάρτησης πρόβλεψης.

6.5.1.2 Παρουσίαση των αποτελεσμάτων.

Αποτελέσματα επιτάχυνσης.

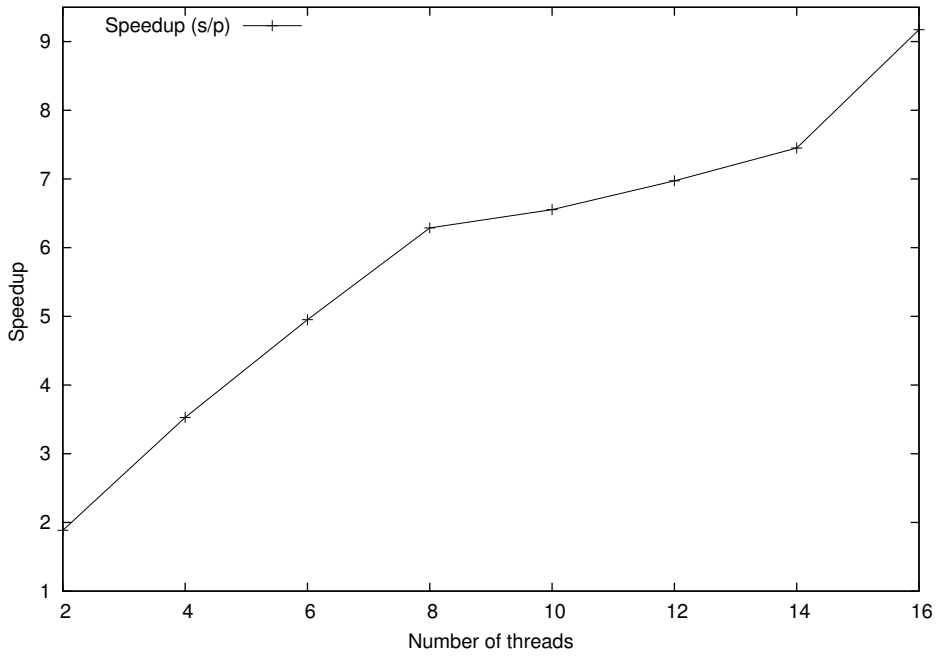
Στην εικόνα 6.1 φαίνεται το διάγραμμα που απεικονίζει το συνολικό χρόνο εκτέλεσης του προγράμματος σε συνάρτηση με το πλήθος των νημάτων.



Σχήμα 6.1: Συνολικός χρόνος εκτέλεσης.

Ο χρόνος εκτέλεσης με χρήση ενός νήματος αντιστοιχεί στο σειριακό χρόνο εκτέλεσης του προγράμματος, δηλαδή με χρήση ενός μόνο νήματος, που επιτυγχάνεται εφαρμόζοντας τη μεταβλητή περιβάλλοντος OMP_NUM_THREADS. Παρατηρείται μείωση του χρόνου εκτέλεσης του προγράμματος όσο αυξάνεται το πλήθος των χρησιμοποιούμενων νημάτων. Ο

χρόνος εκτέλεσης μειώνεται κατά 9 φορές έναντι του ακολουθιακού χρόνου, με τη χρήση 16 νημάτων. Στην εικόνα 6.2 απεικονίζεται ο λόγος του σειριακού χρόνου εκτέλεσης προς τον παράλληλο χρόνο εκτέλεσης για το σύνολο δεδομένων MovieLens 100k.

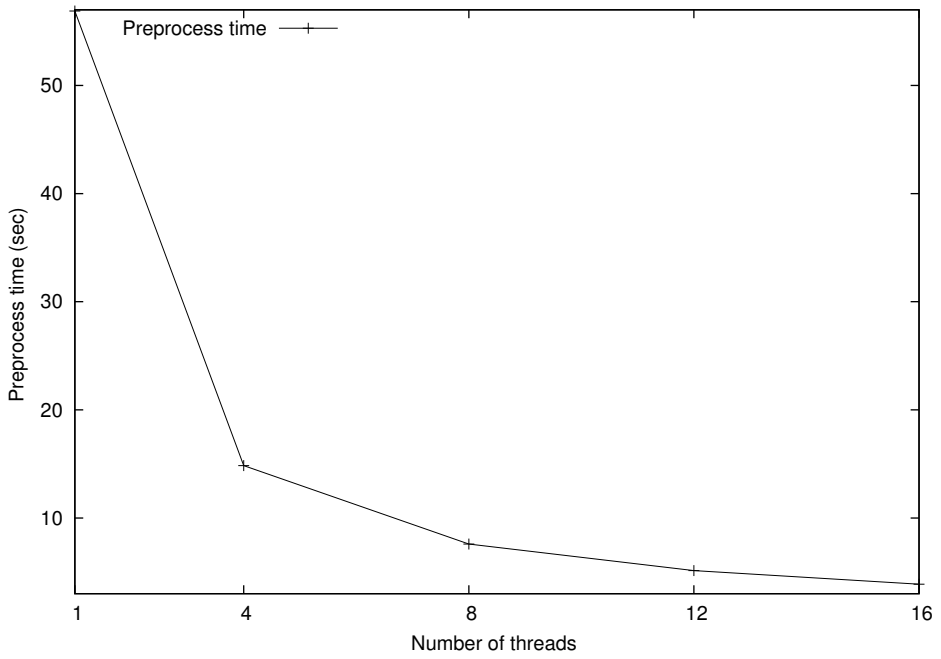


Σχήμα 6.2: Σειριακός χρόνος εκτέλεσης προς παράλληλο για το σύνολο δεδομένων MovieLens 100k.

Αξίζει εδώ να σημειωθεί ότι ο συνολικός χρόνος εκτέλεσης του προγράμματος αφορά τον απαιτούμενο χρόνο υπολογισμών και τον χρόνο προβλέψεων για απλές προβλέψεις, αλλά και για προβλέψεις με βάρη. Στο συνολικό χρόνο συμπεριλαμβάνεται και ο χρόνος εγγραφής των προβλέψεων σε αρχεία. Επομένως, ο χρόνος υπολογισμών, που χρειάζεται να υπολογιστεί μία φορά και αυτό μπορεί να γίνει offline, είναι πολύ μικρότερος. Με σειριακή εκτέλεση του προγράμματος ο χρόνος της φάσης προεπεξεργασίας δεδομένων είναι περίπου 56,88 sec ενώ με τη χρήση 16 νημάτων είναι 3,87 sec, δηλαδή 14×γρηγορότερος. Ο χρόνος αυτός σε συνάρτηση με το πλήθος νημάτων φαίνεται στην εικόνα 6.3 .

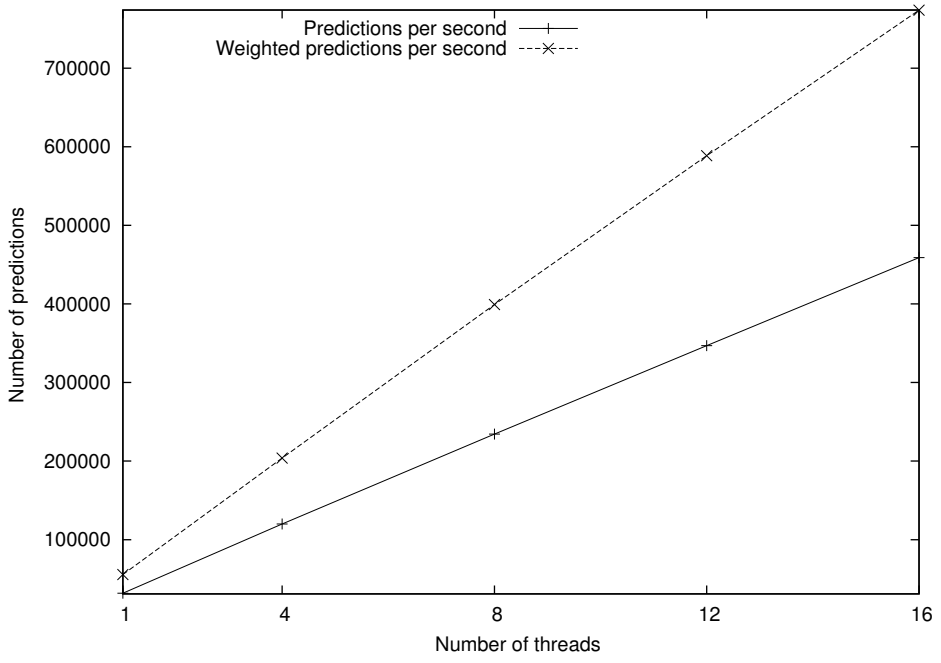
Οι προβλέψεις επίσης παράγονται γρηγορότερα. Ενώ για το σειριακό πρόγραμμα χρειάστηκαν περίπου 47 sec για να παραχθούν όλες οι προβλέψεις και περίπου 27 sec για να παραχθούν όλες οι προβλέψεις με βάρη, με τη χρήση 16 νημάτων χρειάστηκαν μόνο 3,2 sec και 1,9 sec αντίστοιχα. Σημειώνεται εδώ ότι οι χρόνοι που μετρήθηκαν αποτελούν όλο το χρόνο που χρειάστηκε για να παραχθούν προβλέψεις για όλα τα αντικείμενα για τα οποία οι χρήστες δεν είχαν δώσει εκτιμήσεις, και για όλους τους χρήστες του συνόλου δεδομένων. Δεν συμπεριλαμβάνεται στο χρόνο των προβλέψεων ο χρόνος αποθήκευσής τους σε αρχεία. Στην παρούσα υλοποίηση, όλα τα νήματα αποθηκεύουν τα αποτελέσματά τους στο ίδιο αρχείο. Το γεγονός αυτό προκαλεί επιβάρυνση στο χρόνο I/O του προγράμματος, όμως κατά τις μετρήσεις αυτός ο χρόνος δεν έχει συμπεριληφθεί στα αποτελέσματα. Η εντολή αποθήκευσης των αποτελεσμάτων των προβλέψεων σε αρχείο περιλήφθηκε ως σχόλιο διότι σε διαφορετική περίπτωση δεν ήταν δυνατό να μετρηθεί σωστά ο χρόνος των προβλέψεων.

Για το σειριακό πρόγραμμα παράγονται περίπου 31847 προβλέψεις ανά δευτερόλεπτο και 55630 προβλέψεις με βάρη ανά δευτερόλεπτο. Ενώ με χρήση 16 νημάτων παράγονται 458999

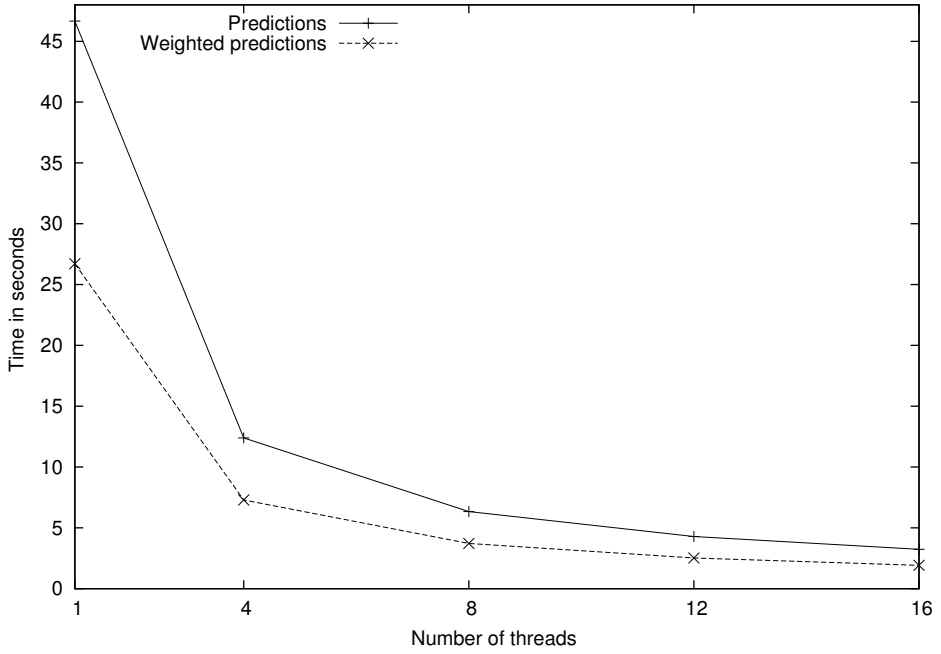


Σχήμα 6.3: Συνολικός χρόνος φάσης προεπεξεργασίας δεδομένων.

προβλέψεις ανά δευτερόλεπτο και 773889 προβλέψεις με βάρη ανά δευτερόλεπτο. Στην εικόνα 6.4 απεικονίζεται το διάγραμμα του πλήθους προβλέψεων και προβλέψεων με βάρη ανά δευτερόλεπτο, συναρτήσει του πλήθους νημάτων. Φαίνεται ότι όσο περισσότερα νήματα χρησιμοποιούνται, τόσο περισσότερες προβλέψεις παράγονται ανά δευτερόλεπτο και η αύξηση αυτή είναι γραμμική. Στην εικόνα 6.5 φαίνεται ο χρόνος παραγωγής προβλέψεων σε δευτερόλεπτα ανά πλήθος νημάτων, όπου και σε αυτή την περίπτωση αποδεικνύεται ότι η αύξηση του πλήθους νημάτων επιφέρει μείωση στο χρόνο.



Σχήμα 6.4: Προβλέψεις ανά δευτερόλεπτο.



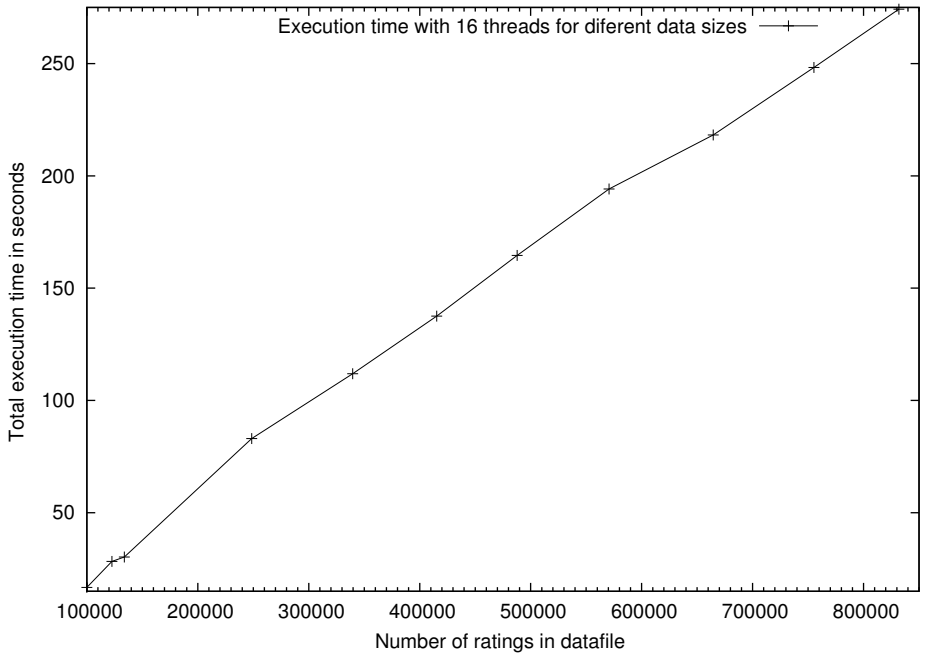
Σχήμα 6.5: Χρόνος προβλέψεων ανά αριθμό νημάτων.

Αποτελέσματα κλιμάκωσης.

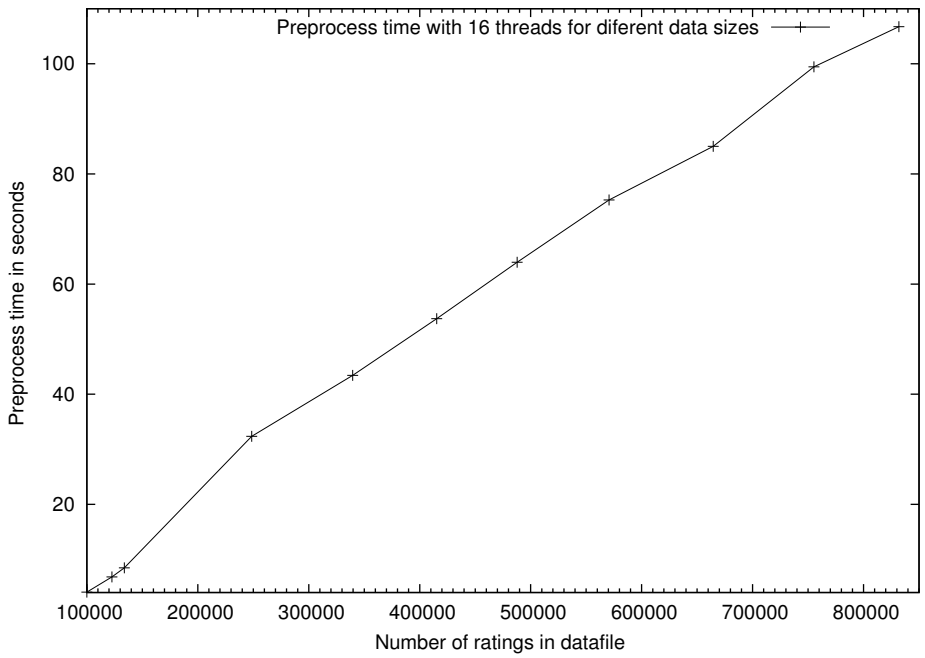
Παρατηρείται στα παρακάτω διαγράμματα ότι ο συνολικός χρόνος εκτέλεσης (σχήμα 6.6), ο χρόνος της φάσης προεπεξεργασίας των δεδομένων (σχήμα 6.7) και ο χρόνος παραγωγής προβλέψεων (σχήμα 6.8) είναι ανάλογοι του πλήθους δεδομένων που επεξεργάζονται κάθε φορά. Όσο περισσότερα δεδομένα επεξεργάζονται, τόσο περισσότερος χρόνος απαιτείται για την παραγωγή συστάσεων. Επιπλέον, παρατηρείται (σχήμα 6.9) ότι σε μικρά σύνολα δεδομένων παράγονται περισσότερες προβλέψεις ανά δευτερόλεπτο από ότι σε μεγάλα σύνολα δεδομένων, αλλά τελικά ο αριθμός των προβλέψεων σταθεροποιείται περίπου στις 214000 προβλέψεις ανά δευτερόλεπτο και στις 362000 προβλέψεις με βάρη ανά δευτερόλεπτο.

Το γεγονός αυτό πιθανώς σχετίζεται με την πυκνότητα των συνόλων δεδομένων που χρησιμοποιούνται. Στον πίνακα Α΄.2 του παραρτήματος Α΄ φαίνεται η πυκνότητα των συνόλων δεδομένων. Παρατηρείται ότι στο πρώτο σύνολο δεδομένων με τις 100000 αξιολογήσεις η πυκνότητα είναι μεγαλύτερη από τα άλλα σύνολα δεδομένων. Η πυκνότητα των συνόλων δεδομένων παραμένει σχεδόν σταθερή από το τέταρτο σύνολο δεδομένων και έπειτα, και στο σχήμα 6.9 φαίνεται να σταθεροποιείται ο αριθμός των προβλέψεων ανά δευτερόλεπτο από το τέταρτο σύνολο δεδομένων και μετά.

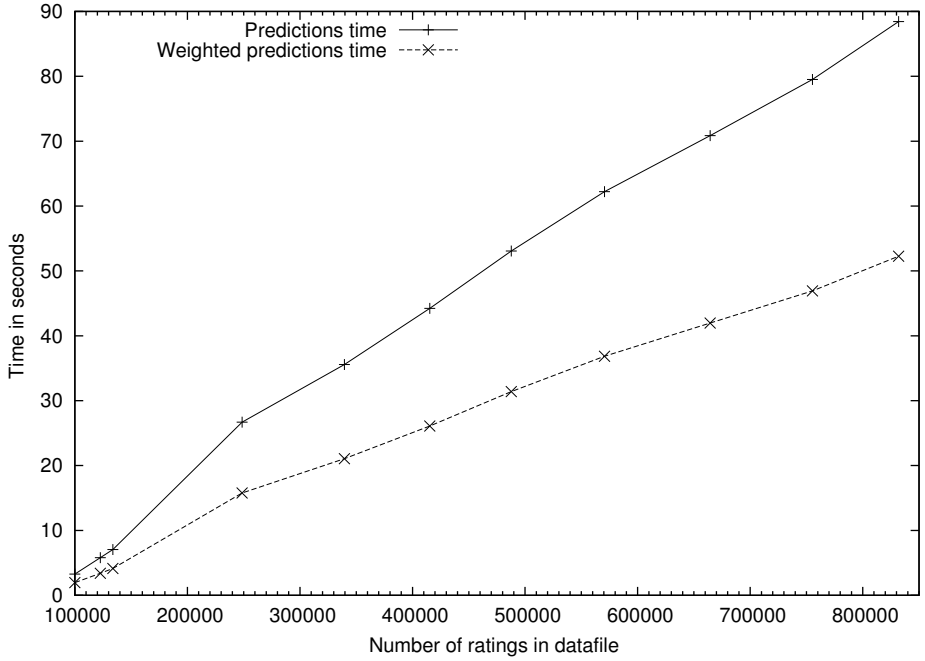
Στο διάγραμμα 6.10 συγκρίνεται ο συνολικός χρόνος της σειριακής εκτέλεσης της εφαρμογής, με το συνολικό χρόνο της παράλληλης εφαρμογής για τα διάφορα σύνολα δεδομένων. Συγκεκριμένα απεικονίζεται ο λόγος του σειριακού χρόνου εκτέλεσης προς τον παράλληλο χρόνο εκτέλεσης. Όσο μεγαλώνει το μέγεθος των συνόλων δεδομένων, επιτυγχάνεται βελτίωση του παράλληλου χρόνου εκτέλεσης 13×σε σχέση με το χρόνο εκτέλεσης της σειριακής εφαρμογής.



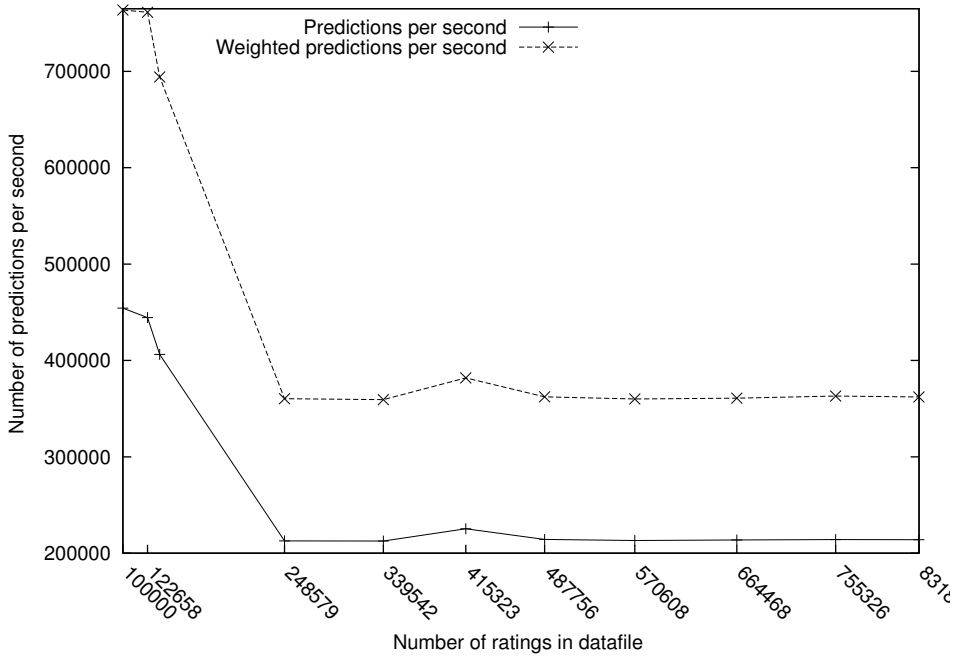
Σχήμα 6.6: Συνολικός χρόνος εκτέλεσης συναρτήσει μεγέθους συνόλου δεδομένων.



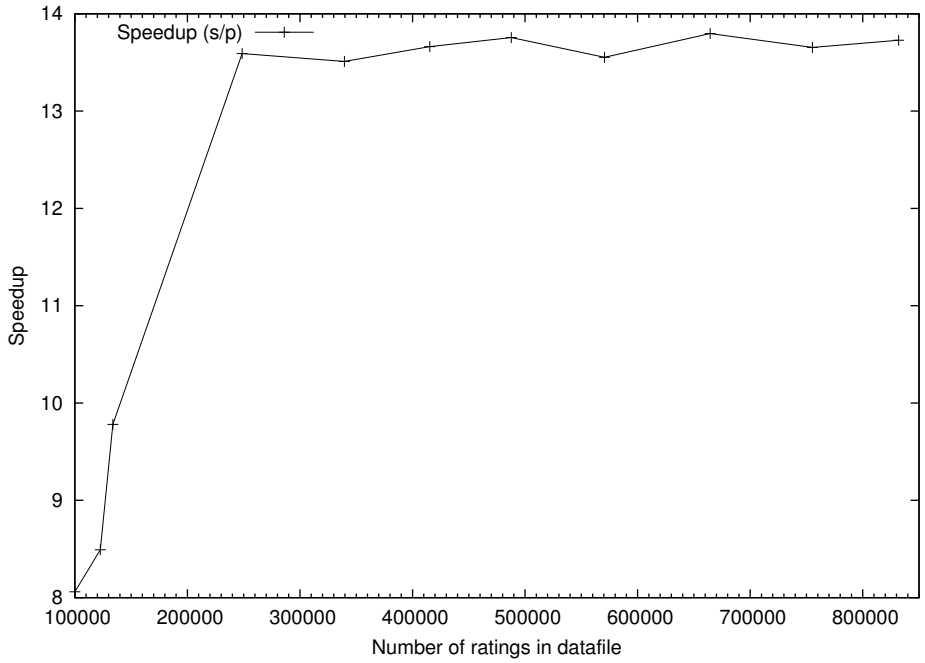
Σχήμα 6.7: Χρόνος προεπεξεργασίας συναρτήσει μεγέθους συνόλου δεδομένων.



Σχήμα 6.8: Χρόνος παραγωγής προβλέψεων συναρτήσει μεγέθους συνόλου δεδομένων.



Σχήμα 6.9: Προβλέψεις ανά δευτερόλεπτο συναρτήσει μεγέθους συνόλου δεδομένων.



Σχήμα 6.10: Σειριακός χρόνος εκτέλεσης προς παράλληλο χρόνο εκτέλεσης.

6.5.2 Υβριδική εφαρμογή με MPI και openMP.

Περιγράφεται παρακάτω η διαδικασία εκτέλεσης της εφαρμογής με ταυτόχρονη χρήση των διεπαφών OpenMP και MPI και παρουσιάζονται τα αποτελέσματά της.

Το σύστημα στο οποίο εκτελέστηκαν τα πειράματα αποτελείται από δώδεκα διπύρηνους υπολογιστές με hyperthreading, δηλαδή από εικοσιτέσσερις φυσικούς πυρήνες και άλλους τόσους εικονικούς, με επεξεργαστή Intel(R) Core(TM) i3 και με συχνότητα ρολογιού στα 2.93Ghz. Η μνήμη RAM κάθε κόμβου αποτελείται από 4 GB. Χρησιμοποιείται κάρτα δικτύου Realtec Semiconductor Co. Ltd. RTL8111/8168B PCI Express Gigabit Ethernet Controller (rev 03). Το λειτουργικό σύστημα στο οποίο εκτελέστηκαν τα πειράματα είναι το Ubuntu Linux 10.04. Τα παρακάτω παραδείγματα υλοποιήθηκαν με τη γλώσσα προγραμματισμού C. Για τη μέτρηση του χρόνου χρησιμοποιήθηκε η συνάρτηση MPI Wtime() του πρωτοκόλλου επικοινωνίας MPI. Οι μετρήσεις έγιναν στα σύνολα δεδομένων της MovieLens, τα οποία περιγράφονται στο παράρτημα Α΄ για μέγεθος συστοιχίας ίσο με 2, 4, 6, 8, 10 και 12.

Η προηγούμενη εφαρμογή διατηρήθηκε όπως ήταν και σε αυτή προστέθηκε το πρωτόκολλο επικοινωνίας MPI. Η μέθοδος που ακολουθήθηκε είναι αυτή του συντονιστή-εργαζομένων με δημιουργία δυναμικών διεργασιών. Δηλαδή ένας υπολογιστής αναλαμβάνει το ρόλο του συντονιστή και ρυθμίζει την εργασία που αναλαμβάνουν οι υπόλοιποι υπολογιστές της συστοιχίας. Ο συντονιστής χωρίζει τα δεδομένα σε τμήματα και αποστέλλει στους εργαζομένους τα τμήματα αυτά, τα οποία είναι περισσότερα σε πλήθος από το πλήθος των υπολογιστών της συστοιχίας. Όταν κάποιος εργαζόμενος τελειώνει την εργασία που του έχει ανατεθεί, ο συντονιστής φροντίζει να του αναθέσει νέο τμήμα των δεδομένων, μέχρι να εξαντληθούν τα δεδομένα.

Η παρούσα υλοποίηση πραγματοποιείται σε υβριδικό σύστημα υπολογιστών, όπου συνδυάζονται οι αρχιτεκτονικές μοιραζόμενης και κατανεμημένης μνήμης. Ακολουθείται το υβρι-

δικό μοντέλο παράλληλου προγραμματισμού, που συνδυάζει το μοντέλο διεργασιών, με τη μεταβίβαση μηνυμάτων, και το μοντέλο νημάτων. Οι λειτουργίες χωρίζονται σε τέσσερα μέρη, το συντονισμό των εργασιών, τους υπολογισμούς, τη συλλογή των αποτελεσμάτων των υπολογισμών και την παραγωγή προβλέψεων. Το συντονισμό τον αναλαμβάνει ένας υπολογιστής της συστοιχίας, ενώ οι υπολογισμοί γίνονται στους υπόλοιπους υπολογιστές. Η συλλογή των αποτελεσμάτων των υπολογισμών και η παραγωγή των προβλέψεων συντελούνται στον υπολογιστή που αναλαμβάνει το συντονισμό. Εκτός από τον επιμερισμό των λειτουργιών, παραλληλοποιούνται και τα δεδομένα, αφού ο συντονιστής διαμοιράζει τμήματα των δεδομένων στους υπόλοιπους υπολογιστές.

Τα χρονοβόρα τμήματα του προγράμματος εξακολουθούν να είναι τα σημεία όπου υπολογίζονται οι διάφοροι πίνακες. Για αυτό, και διαμοιράζεται η εργασία στα τμήματα αυτά. Με τη δημιουργία δυναμικών διεργασιών επιτυγχάνεται εξισορρόπηση φορτίου, ώστε να ελαχιστοποιείται ο χρόνος κατά τον οποίο δεν πραγματοποιούνται υπολογισμοί. Επιπλέον, με την εφαρμογή του MPI στο πρόγραμμα, απαιτείται επικοινωνία μεταξύ των υπολογιστών της συστοιχίας για την ανταλλαγή των δεδομένων.

Ο συντονιστής δημιουργεί πίνακες testratings σταθερού πάντα μεγέθους τους οποίους στέλνει κάθε φορά στους εργαζόμενους. Οι πίνακες αυτοί είναι υποπίνακες του πίνακα ratings που περιγράφηκε στην προηγούμενη υλοποίηση και περιέχουν users2 χρήστες ο καθένας, και τις εκτιμήσεις που έχουν δώσει αυτοί για τα αντικείμενα του συστήματος. Ο προσδιορισμός του καταλληλότερου μεγέθους των πακέτων προς αποστολή έγινε με τη μέθοδο ping pong, στην οποία επικοινωνούν δύο διεργασίες αποστέλλοντας η μία σστην άλλη πακέτα διαφορετικών μεγεθών, και μετρώντας το χρόνο αμφίδρομης επικοινωνίας.

Κάθε φορά που ο συντονιστής λαμβάνει σήμα από κάποιον εργαζόμενο ότι είναι διαθέσιμος να παραλάβει νέα δεδομένα, του τα αποστέλλει και συνεχίζει έτσι, ώσπου να τελειώσουν όλα τα δεδομένα. Όταν δεν υπάρχουν άλλα δεδομένα, ξεκινάει να παραλαμβάνει αποτελέσματα από τους εργαζόμενους και τα ταξινομεί στη σωστή θέση των πινάκων denominator και deviation. Υπολογίζει στη συνέχεια το σωστό πίνακα deviation που δεν μπορούσε να υπολογιστεί στους εργαζόμενους γιατί δεν είχαν όλοι όλα τα στοιχεία που χρειάζονται και τέλος παράγει τις προβλέψεις. Ο υπολογισμός του πίνακα deviation είναι όμοιος με αυτόν του πλαισίου 6.8 και ο υπολογισμός των προβλέψεων όμοιος με αυτόν του πλαισίου 6.9. Η λειτουργία του υπολογισμού των προβλέψεων θα μπορούσε να πραγματοποιηθεί ακόμα πιό γρήγορα, με χρήση περισσότερων υπολογιστών. Στην παρούσα εφαρμογή, χρησιμοποιείται μόνο ο συντονιστής για την παραγωγή τους. Οι εργαζόμενοι δεν αναλαμβάνουν τον υπολογισμό των προβλέψεων διότι αυτό θα απαιτούσε αύξηση του κόστους επικοινωνίας.

Οι εργαζόμενοι λαμβάνουν τους πίνακες testratings, υπολογίζουν τους πίνακες numerator, denominator και deviation, και όταν ενημερωθούν από το συντονιστή ότι δεν υπάρχουν άλλα δεδομένα, του αποστέλλουν τα αποτελέσματά τους. Οι πίνακες denominator και deviation που υπολογίζονται στους εργαζόμενους, όπως ακριβώς και στα πλαίσια 6.6 και 6.7, ονομάζονται denominator2 και deviation2 διότι οι υπολογισμοί που πραγματοποιούνται αφορούν μόνο το τμήμα δεδομένων που έχει ληφθεί και όχι όλο το σύνολο δεδομένων. Η διαφορά στο όνομα γίνεται για να διαφοροποιηθούν από τους πλήρεις πίνακες που σχηματίζονται στο συντονιστή. Ο πίνακας numerator χρησιμοποιείται τοπικά σε κάθε εργαζόμενο και δεν είναι απαραίτητη η αποστολή του στο συντονιστή. Η αποστολή αποτελεσμάτων γίνεται μόνο μία φορά από κάθε εργαζόμενο με σκοπό τη μείωση του χρόνου επικοινωνίας μεταξύ των εργαζομένων και του συντονιστή.

Παρακάτω παρατίθεται ο κώδικας που προστέθηκε στην υλοποίηση αυτή. Ο κώδικας που φαίνεται στο πλαίσιο 6.10 είναι ο κώδικας του συντονιστή και στο πλαίσιο 6.11 είναι ο κώδικας των εργαζομένων. Ο υπολογισμός του πίνακα deviation και των προβλέψεων στο συντονιστή γίνεται ακριβώς όπως περιγράφεται στα πλαίσια 6.8 και 6.9 αντίστοιχα.
Στον κώδικα των εργαζομένων, ο υπολογισμός των πινάκων numerator, denominator2 και deviation2 γίνεται όπως ακριβώς και στα πλαίσια 6.6 και 6.7.

```
/* ###### Set Ratings ###### */

// listing 6.5

/* ###### Sending testratings to workers ###### */

for(i=1;i<size;i++)
{
  q=counter*ITEMS*users2;

  #pragma omp parallel private(j)
  {
  #pragma omp for
  for(j=0;j<users2*ITEMS ;j++)
  {
    testratings[j]=ratings[q+j];

  }
  } /* end of parallel section */
  MPI_Send(testratings, users2*ITEMS, MPI_INT, i, FROM_MASTER,
     MPI_COMM_WORLD);
  counter++;
  active++;
}


do
{
/* ###### Master receiving sth from workers ###### */

MPI_Recv(&sth,1,MPI_INT,MPI_ANY_SOURCE,7,MPI_COMM_WORLD,&status);
active--;
sender=(status.MPI_SOURCE);
/* ###### Master sending next data or dietag to workers ###### */
if(counter*users2<=users)
{
  #pragma omp parallel private(j)
    {
    #pragma omp for
    for(j=0;j<users2*ITEMS ;j++)
    {
    testratings[j]=ratings[counter*ITEMS*users2+j];
    }
  } /* end of parallel section */
  MPI_Send(testratings, users2*ITEMS, MPI_INT, sender, FROM_MASTER,
     MPI_COMM_WORLD);
  counter++;
  active++;
}
```

```
else if ( counter*users2>users )
{
  tag=DIETAG;
  MPI_Send(testratings,users2*ITEMS, MPI_INT, sender, tag,MPI_COMM_WORLD
     );
  counter++;
  /* ###### Master receiving results from workers ###### */
  MPI_Recv(denominator2,ITEMS*ITEMS,MPI_INT,MPI_ANY_SOURCE,4,
     MPI_COMM_WORLD,&status);
  MPI_Recv(deviation2,ITEMS*ITEMS,MPI_FLOAT,MPI_ANY_SOURCE,5,
     MPI_COMM_WORLD,&status );
  /* ###### Master placing results to right position ###### */
  #pragma omp parallel private(i,j)
  {
  #pragma omp for
  for(i=0;i<ITEMS;i++)
  {
    for(j=0;j<ITEMS;j++)
    {
    denominator[ i*ITEMS+j]=denominator[ i*ITEMS+j ]+denominator2[ i*
       ITEMS+j ];
    deviation[ i*ITEMS+j ]=deviation[ i*ITEMS+j]+deviation2[ i*ITEMS+j
       ];
    }
  }
  } /* end of parallel section */
}

}while(active>0);
```

Listing 6.10: Πηγαίος κώδικας του συντονιστή.

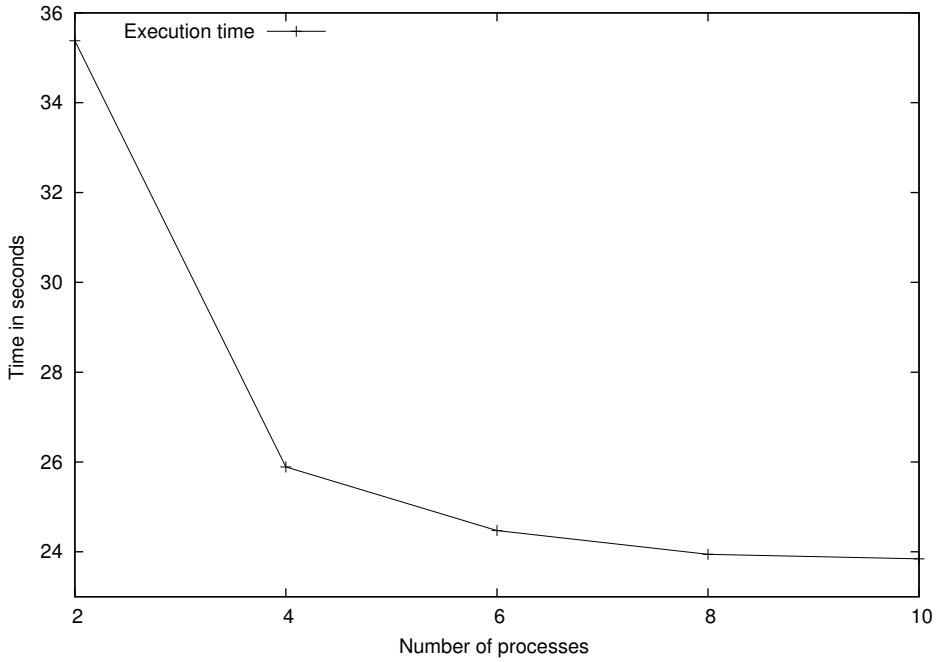
```
while(TRUE)
{
MPI_Recv(testratings, users2*ITEMS, MPI_INT,MASTER,MPI_ANY_TAG,
   MPI_COMM_WORLD,&status);

/* ####### Check the tag of the received message ###### */
tag=status.MPI_TAG;
if(tag==DIETAG){
MPI_Send(denominator2,ITEMS*ITEMS,MPI_INT,MASTER,4,MPI_COMM_WORLD);
MPI_Send(deviation2,ITEMS*ITEMS,MPI_FLOAT,MASTER,5,MPI_COMM_WORLD);
break;
}

/* ###### Set numerator(differences matrix) - denominator(frequencies
   matrix) ###### */
/* ###### Set (pre)deviation ###### */

// listing 6.6

// listing 6.7

/* ###### ------ ###### */

MPI_Send(&sth,1,MPI_INT,MASTER,7,MPI_COMM_WORLD);

}  /* end of while */
```

Listing 6.11: Πηγαίος κώδικας των εργαζομένων.

6.5.2.1 Παρουσίαση των αποτελεσμάτων.

Αποτελέσματα επιτάχυνσης.

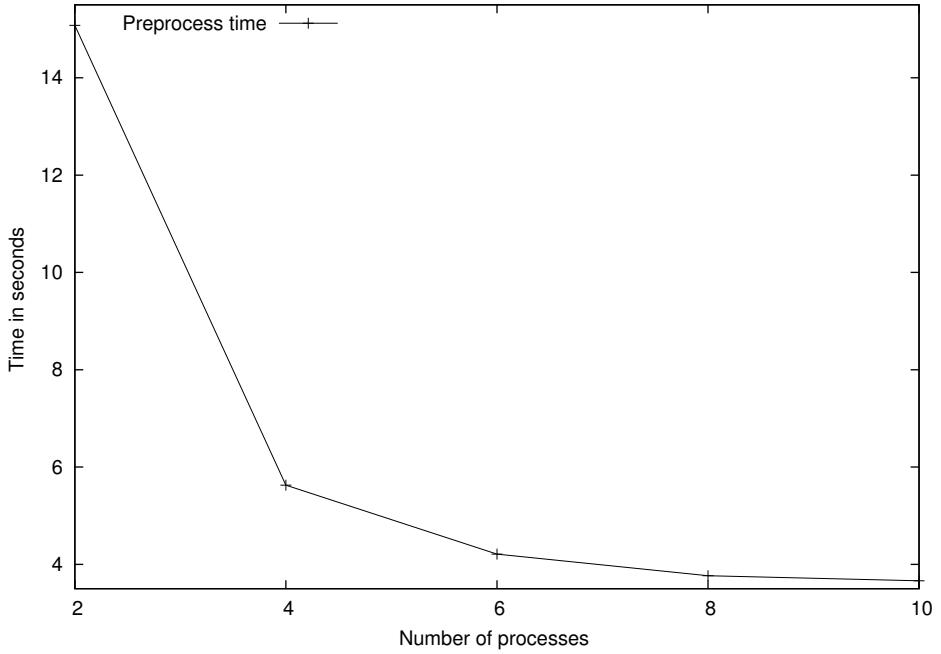
Σε αυτή την εφαρμογή παρατηρείται μείωση του συνολικού χρόνου εκτέλεσης κατά 5,7 φορές από το χρόνο εκτέλεσης της σειριακής εφαρμογής. Στο σχήμα 6.11 απεικονίζεται ο συνολικός χρόνος εκτέλεσης της υβριδικής εφαρμογής συναρτήσει του πλήθους των κόμβων που χρησιμοποιήθηκαν και παρατηρείται ότι ο χρόνος αυτός μειώνεται όσο προσθέτονται κόμβοι στο σύστημα. Επειδή οι μετρήσεις αφορούν το σύνολο δεδομένων MovieLens 100k το οποίο περιέχει 943 χρήστες και τα πακέτα δεδομένων που αποστέλλονται περιέχουν από 100 χρήστες το καθένα, δεν αξιοποιήθηκε ο δωδέκατος κόμβος της διαθέσιμης συστοιχίας στον υπολογισμό της επιτάχυνσης διότι όλα τα δεδομένα μπορούν να διαμοιρασθούν μόνο με 10 αποστολές. Σε αυτή την περίπτωση δε γίνεται χρήση της δυναμικής αποστολής δεδομένων διότι κανένας εργαζόμενος δε χρειάζεται να παραλάβει δεύτερο πακέτο δεδομένων.



Σχήμα 6.11: Χρόνος εκτέλεσης συναρτήσει πλήθους κόμβων.

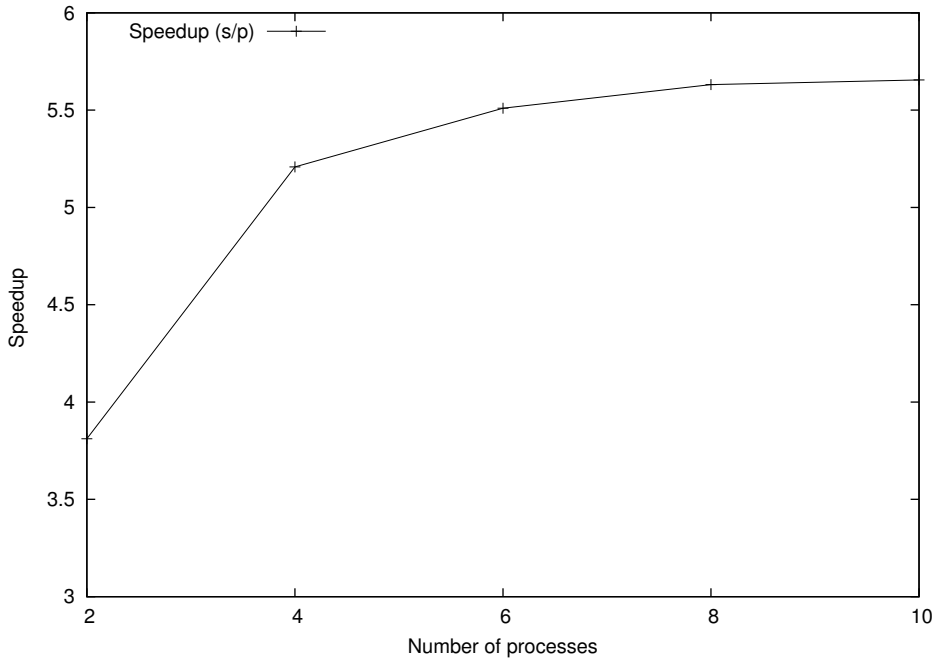
Ο χρόνος της φάσης πρόεπεξεργασίας δεδομένων επίσης παρουσιάζει μείωση όσο αυξάνεται το πλήθος των κόμβων που χρησιμοποιούνται. Παρατηρείται ότι ενώ στη σειριακή εκτέλεση η φάση αυτή διαρκεί 56,88 sec, στην υβριδική υλοποίηση απαιτούνται μόλις 3,66 sec, δηλαδή το υβριδικό πρόγραμμα είναι 15,5×ταχύτερο του σειριακού. Ο χρόνος της φάσης προεπεξεργασίας συναρτήσει του πλήθους κόμβων που χρησιμοποιούνται για το σύνολο δεδομένων MovieLens 100k φαίνεται στο σχήμα 6.12.

Στο διάγραμμα 6.13 απεικονίζεται ο λόγος του σειριακού χρόνου εκτέλεσης προς τον παράλληλο χρόνο εκτέλεσης για το σύνολο δεδομένων MovieLens 100k συναρτήσει του πλήθους



Σχήμα 6.12: Offline χρόνος συναρτήσει πλήθους κόμβων.

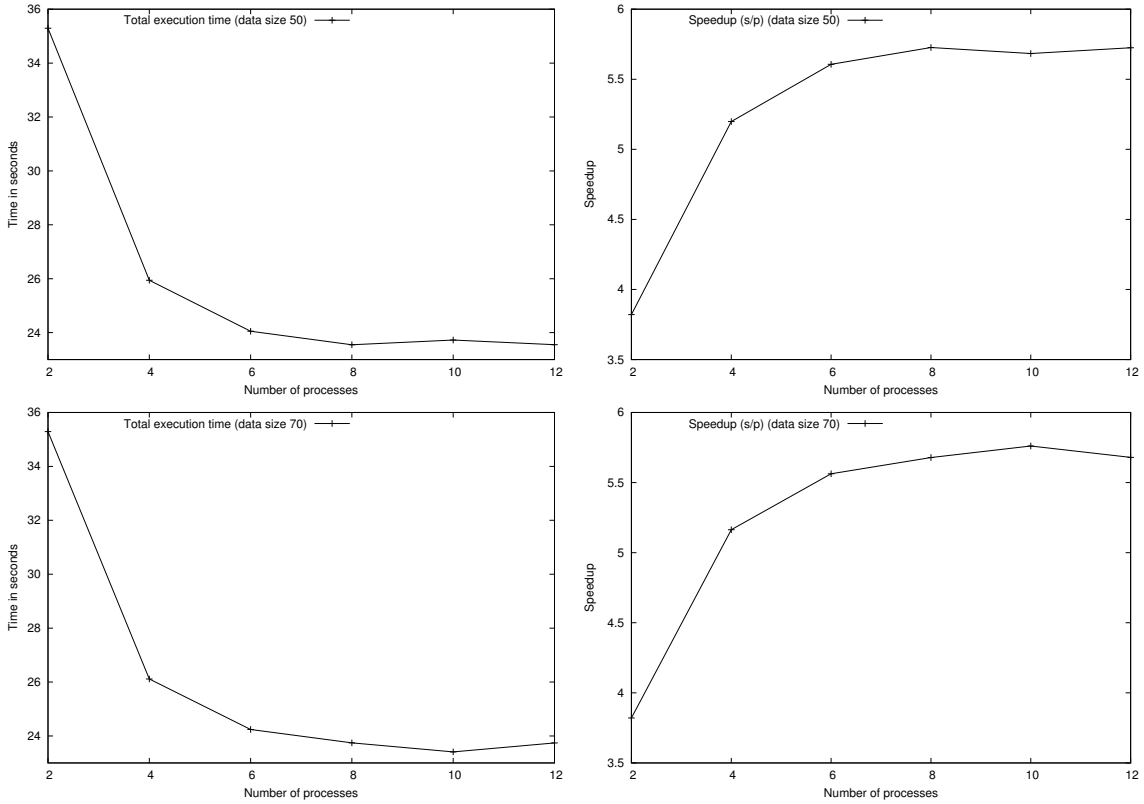
κόμβων. Παρατηρείται μεγάλη επιτάχυνση με χρήση τεσσάρων υπολογιστών, οπότε υπάρχει σημαντικό όφελος από την προσθήκη επιπλέον κόμβων. Χρησιμοποιώντας από τέσσερις και άνω υπολογιστές η επιτάχυνση είναι ομαλή.



Σχήμα 6.13: Λόγος χρόνου σειριακής υλοποίησης προς το χρόνο παράλληλης υλοποίησης συναρτήσει πλήθους κόμβων.

Τα αποτελέσματα που παρουσιάστηκαν μέχρι αυτό το σημείο αφορούν στατική αποστολή δεδομένων, αφού κανένας από τους εργαζόμενους δε λαμβάνει εκ νέου δεδομένα. Στη Συνέχεια παρουσιάζονται τα αποτελέσματα των εκτελέσεων του πειράματος με χρήση μικρότερων πα-

κέτων δεδομένων, ώστε να επιτυγχάνεται δυναμική αποστολή τους και να αξιοποιούνται όλοι οι υπολογιστές του συστήματος. Στο σχήμα 6.14 απεικονίζονται τέσσερα διαγράμματα. Το πρώτο και το δεύτερο αφορούν το συνολικό χρόνο εκτέλεσης του προγράμματος και την επιτάχυνση για το σύνολο δεδομένων MovieLens 100k και για μέγεθος πακέτων των 50 χρηστών, και τα επόμενα δύο για μέγεθος πακέτων των 70 χρηστών. Όταν αποστέλλεται πακέτο των 50 χρηστών επιτυγχάνονται ελαφρώς καλύτεροι χρόνοι, διότι σε αυτή την περίπτωση το τελευταίο πακέτο περιέχει λιγότερα άχρηστα δεδομένα.



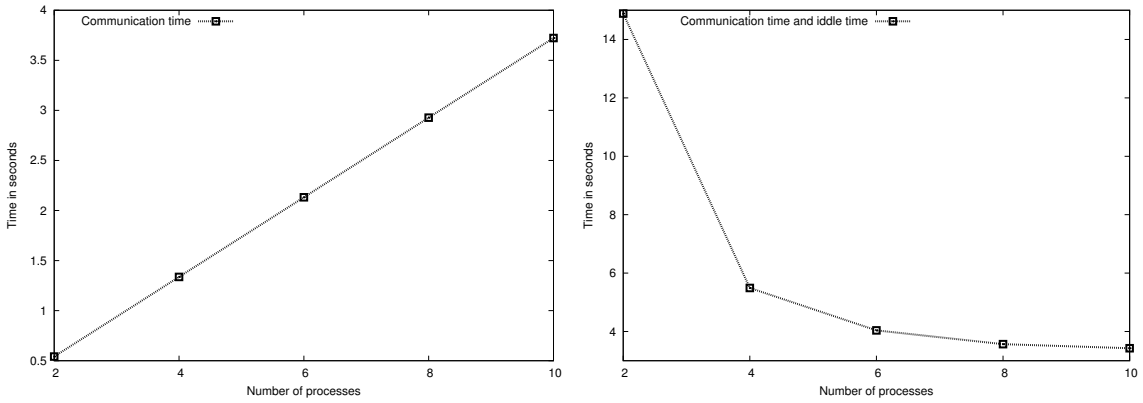
Σχήμα 6.14: Χρόνος εκτέλεσης και επιτάχυνση βάσει μεγέθους πακέτων δεδομένων.

Ο χρόνος επικοινωνίας μεταξύ των κόμβων της συστοιχίας μετρήθηκε στο συντονιστή και περιλαμβάνει τον καθαρό χρόνο όλων των αποστολών και παραλαβών δεδομένων μεταξύ του συντονιστή και των εργαζομένων και το χρόνο αναμονής κατά τον οποίο ο συντονιστής παραμένει άεργος και περιμένει να ολοκληρωθεί η παραλαβή δεδομένων που του έχουν στείλει οι εργαζόμενοι. Εφόσον αποστέλλονται τα ίδια δεδομένα σε κάθε μία από τις εκτελέσεις του πειράματος, δηλαδή για συγκεκριμένο σύνολο δεδομένων και για διαφορετικό πλήθος κόμβων, ο καθαρός χρόνος επικοινωνίας εξαρτάται από το πλήθος των εργαζομένων και από τον όγκο των δεδομένων προς αποστολή.

Ο χρόνος αναμονής του συντονιστή έγκειται στο σημείο μεταξύ της ολοκλήρωσης της πρώτης αποστολής δεδομένων στους εργαζόμενους και της πρώτης παραλαβής του σήματος από κάποιον εργαζόμενο ότι έχει τελειώσει την επεξεργασία. Κατά τη διάρκεια αυτού του χρονικού διαστήματος ο συντονιστής δεν εκτελεί κανέναν υπολογισμό, και λόγω της ανασταλτικής λειτουργίας της MPI_Recv, αναμένει μέχρι την ολοκλήρωση της παραλαβής ενός μηνύματος πρωτού προχωρήσει στην εκτέλεση των επόμενων εντολών.

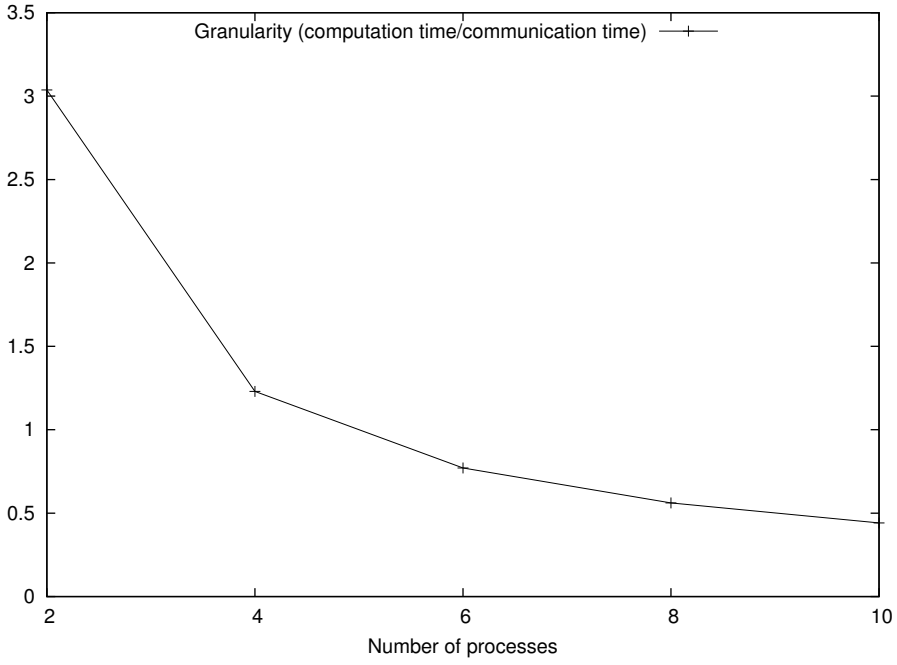
Επομένως, ο καθαρός χρόνος επικοινωνίας μπορεί να υπολογισθεί θεωρητικά μέσω της εφαρμογής ping pong, που μετράει το χρόνο αποστολής και παραλαβής δεδομένων διάφορων

μεγεθών ματαξύ δύο διεργασιών. Στο σχέδιο 6.15 απεικονίζονται δύο διαγράμματα. Το πρώτο αφορά τη θεωρητική επικοινωνία, και το δεύτερο την επικοινωνία που περιλαμβάνει και τον άεργο χρόνο του συντονιστή, συναρτήσει του πλήθους κόμβων. Η επικοινωνία αυξάνεται όσο προσθέτονται κόμβοι στο σύστημα, ενώ ο χρόνος που ο συντονιστής παραμένει άεργος μειώνεται όσο αυξάνονται οι κόμβοι του συστήματος. Ο θεωρητικός υπολογισμός του χρόνου επικοινωνίας περιγράφεται στο παράρτημα Β΄.



Σχήμα 6.15: Χρόνος επικοινωνίας.

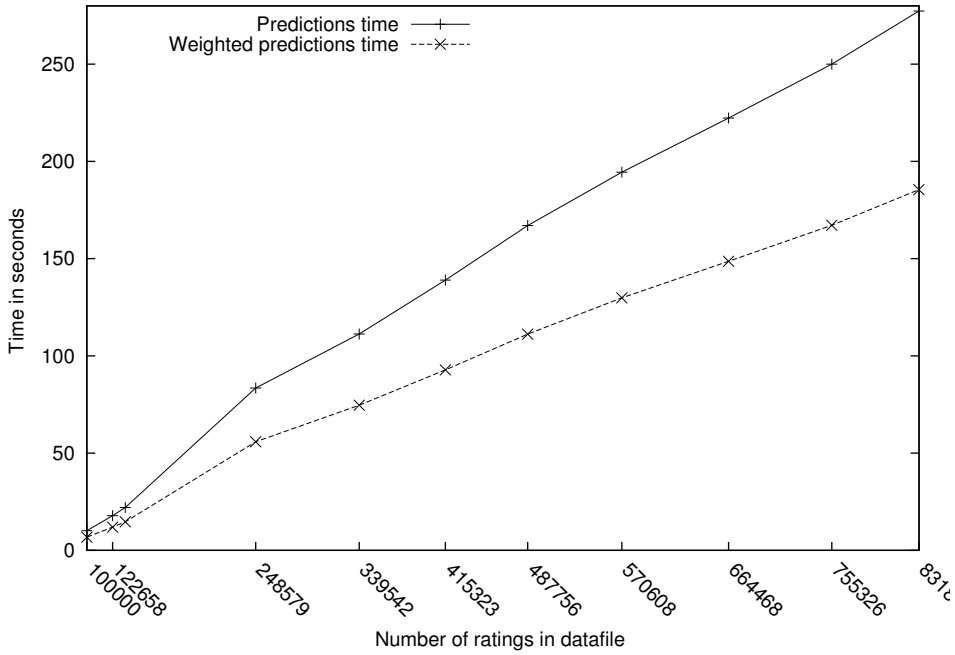
Ο χρόνος υπολογισμών αφορά μόνο τη διάρκεια των υπολογισμών των εργαζομένων. Για κάθε εκτέλεση του πειράματος μετρήθηκε η συνολική διάρκεια των υπολογισμών σε κάθε εργαζόμενο και λήφθηκε υπόψη η μέγιστη από αυτές. Για το σύνολο δεδομένων MovieLens 100k ο μέσος χρόνος υπολογισμών στους εργαζόμενους είναι περίπου 1.64 sec. Στο σχήμα 6.16 φαίνεται ότι η κοκκότητα ελαττώνεται όσο αυξάνεται το πλήθος των χρησιμοποιούμενων κόμβων, γεγονός που οφείλεται στην αύξηση της επικοινωνίας.



Σχήμα 6.16: Κοκκότητα (Χρόνος υπολογισμών/Χρόνο επικοινωνίας).

Αποτελέσματα κλιμάκωσης.

Παρατηρείται από το διάγραμμα 6.17 ότι ο χρόνος παραγωγής προβλέψεων και προβλέψεων με βάρη αυξάνεται ομαλά σε σχέση με το μέγεθος του συνόλου δεδομένων που χρησιμοποιείται κάθε φορά και επιπλέον για να παραχθούν οι προβλέψεις με βάρη απαιτείται λιγότερος χρόνος από ότι για να παραχθούν οι απλές προβλέψεις. Επίσης, στο διάγραμμα 6.18 φαίνεται ότι το πλήθος των προβλέψεων που παράγονται ανά δευτερόλεπτο δεν εξαρτάται από το μέγεθος του συνόλου δεδομένων, αλλά από τη σποραδικότητα των εκτιμήσεών του. Στα πρώτα σύνολα δεδομένων, τα οποία είναι πυκνότερα όπως φαίνεται στον πίνακα Α΄.2 του παραρτήματος Α΄, παράγονται περισσότερες προβλέψεις ανά δευτερόλεπτο, όμως τελικά ο αριθμός προβλέψεων ανά δευτερόλεπτο σταθεροποιείται περίπου στις 68000 για τις απλές προβλέψεις και στις 100000 για τις προβλέψεις με βάρη.



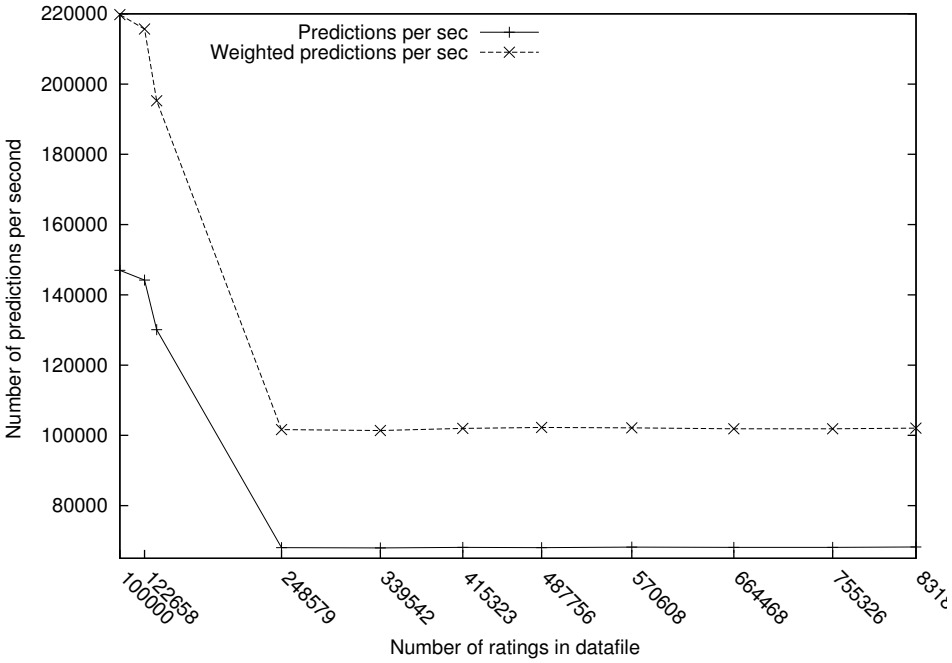
Σχήμα 6.17: Χρόνος προβλέψεων ανά σύνολο δεδομένων.

Ο συνολικός χρόνος παραγωγής των προβλέψεων και των προβλέψεων με βάρη θα μπορούσε να ελαττωθεί ακόμα περισσότερο διότι σε αυτή την υλοποίηση παρά το γεγονός ότι χρησιμοποιούνται πολλοί κόμβοι, μόνο ένας παράγει τις προβλέψεις, και επομένως στον πηγαίο κώδικα για τις προβλέψεις χρησιμοποιείται μόνο OpenMP και έτσι οι προβλέψεις παράγονται αξιοποιώντας μόνο ένα μικρό μέρος των πόρων του συστήματος. Θα μπορούσε λοιπόν να χρησιμοποιηθεί ένας ισχυρότερος υπολογιστής για την παραγωγή προβλέψεων, ώστε να αξιοποιούνται περισσότεροι πυρήνες από τους τέσσερις που χρησιμοποιούνται σε αυτή την εφαρμογή, ή θα μπορούσαν να παράγονται προβλέψεις από όλους τους κόμβους, γεγονός όμως που θα επιβάρυνε το κόστος επικοινωνίας διότι θα έπρεπε να αποσταλλούν σε όλους τους κόμβους τα αποτελέσματα της φάσης πρόεπεξεργασίας των δεδομένων.

6.5.2.2 Υβριδική υλοποίηση σε ετερογενή συστοιχία υπολογιστών.

Σε αυτή την ενότητα υλοποιείται η προηγούμενη υβριδική εφαρμογή σε ετερογενή συστοιχία υπολογιστών και παρουσιάζονται τα αποτελέσματα του πειράματος.

Η ετερογενής συστοιχία υπολογιστών στην οποία εφαρμόστηκε το πείραμα, αποτελείται από



Σχήμα 6.18: Προβλέψεις ανά δευτερόλεπτο ανά σύνολο δεδομένων.

τους εξής έξι κόμβους:

- Έναν υπολογιστή με δύο επεξεργαστές AMD Opteron (tm) Processor 6128 HE όπου κάθε ένας περιέχει οκτώ πυρήνες. Η συχνότητα ρολογιού είναι στα 800 Mhz και το μέγεθος της cache μνήμης είναι 512 KB. Η μνήμη RAM του συστήματος είναι χωρητικότητος 16 GB και η κάρτα δικτύου Intel Corporation 82574L Gigabit Network Connection.

- Από έναν τετραπύρηνο υπολογιστή με Intel(R) Core(TM)2 Quad CPU Q6600 @ 2.40GHz, με 8GB RAM και μέγεθος cache μνήμης 4096 KB. Η συχνότητα ρολογιού είναι στα 1596 Mhz και η κάρτα δικτύου Atheros Communications L1 Gigabit Ethernet Adapter (rev b0).

- Έναν τετραπύρηνο υπολογιστή με Intel(R) Core(TM)2 Quad CPU Q9300 @ 2.50GHz, με 8GB RAM και μέγεθος cache μνήμης 3072 KB. Η συχνότητα ρολογιού είναι στα 1998 Mhz και η κάρτα δικτύου Atheros Communications L1 Gigabit Ethernet Adapter (rev b0).

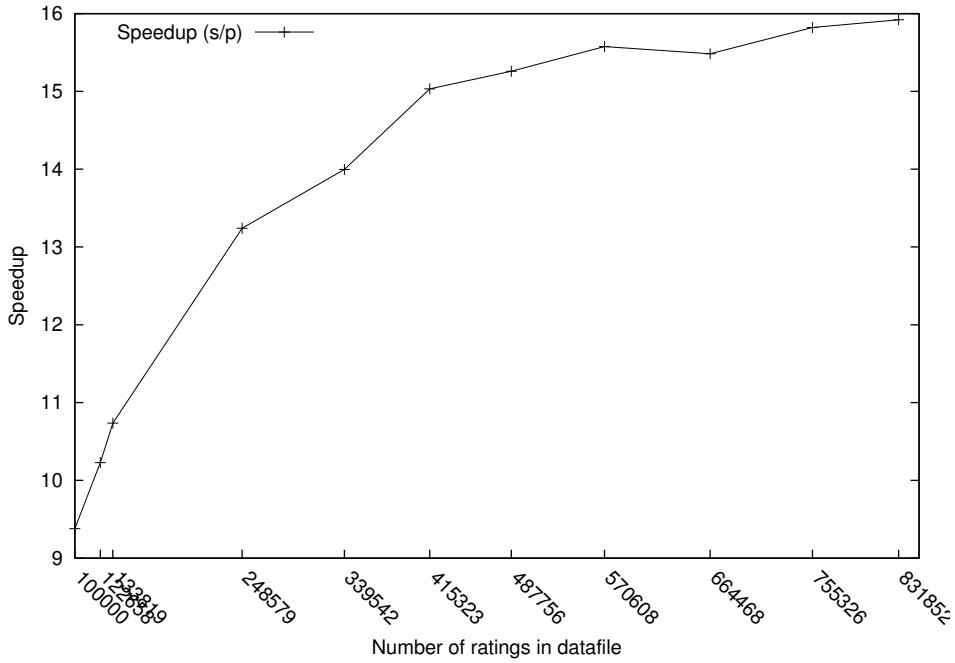
- Τρείς διπύρηνους υπολογιστές με επεξεργαστές Intel(R) Core(TM) i3 και με συχνότητα ρολογιού στα 2.93Ghz. Η μνήμη RAM κάθε κόμβου αποτελείται από 3 GB, το μέγεθος της cache μνήμης είναι 2048 KB και χρησιμοποιείται κάρτα δικτύου Realtec Semiconductor Co. Ltd. RTL8111/8168B PCI Express Gigabit Ethernet Controller (rev 03).

Το λειτουργικό σύστημα στο οποίο εκτελέστηκαν τα πειράματα είναι το Ubuntu Linux 10.04 σε όλους τους παραπάνω υπολογιστές.

Πρώτα γίνεται έλεγχος για την εύρεση του συντονιστή που θα αποδώσει ταχύτερα τα αποτελέσματα, διότι στην υλοποίηση του πειράματος ο συντονιστής εκτός από το συντονισμό

της επικοινωνίας και της κατανομής φόρτου εργασίας στους εργαζομένους υπολογίζει και τις προβλέψεις, οπότε υπάρχει κόστος υπολογισμών καί στο τμήμα του συντονιστή. Για την επιλογή του κατάλληλου συντονιστή, μετρήθηκε ο συνολικός χρόνος εκτέλεσης και επιλέχθηκε ώς συντονιστής αυτός που πραγματοποίησε το μικρότερο. Επιλέχθηκε ώς συντονιστής ο τετραπύρηνος υπολογιστής με Intel(R) Core(TM)2 Quad CPU Q9300 @ 2.50GHz, 8GB RAM και μέγεθος cache μνήμης 3072 KB. Με τη χρήση αυτού του υπολογιστή ώς συντονιστή επιτεύχθηκε για το σύνολο δεδομένων MovieLens 100k χρόνος εκτέλεσης 14.06 sec. Στη συνέχεια, μετά την επιλογή του κατάλληλου συντονιστή, λαμβάνονται μετρήσεις για τα διάφορα σύνολα δεδομένων.

Παρατηρείται αύξηση στο λόγο του χρόνου της σειριακής εκτέλεσης προς το χρόνο της παράλληλης όσο αυξάνεται ο όγκος των συνόλων δεδομένων. Για μεγαλύτερα σύνολα δεδομένων επιτυγχάνεται μεγαλύτερη επιτάχυνση, που φτάνει μέχρι και 15,9×. Η επιτάχυνση φαίνεται στο σχήμα 6.19.

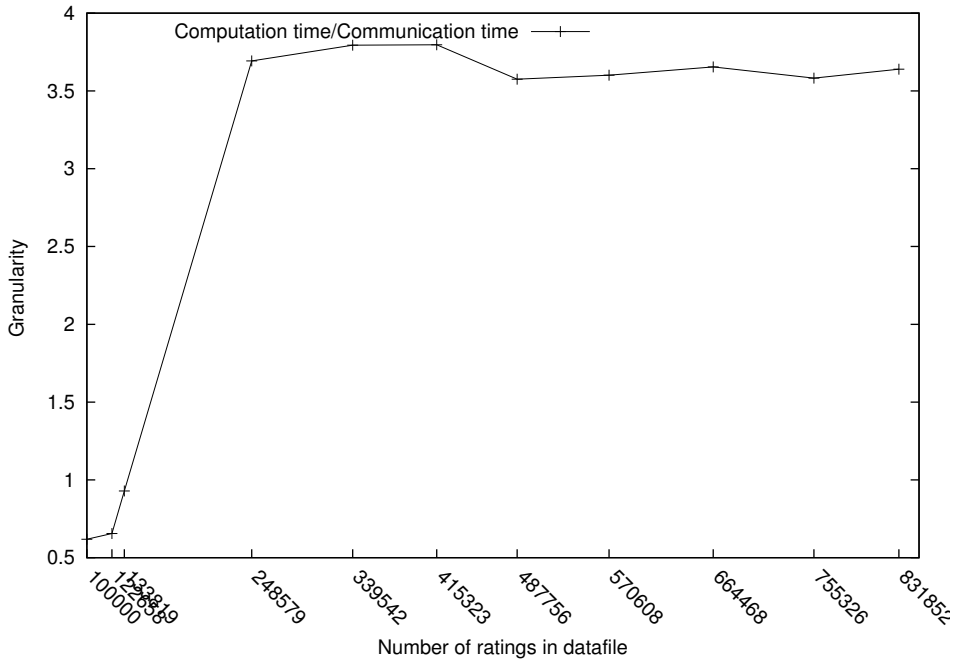


Σχήμα 6.19: Λόγος σειριακού προς παράλληλο χρόνο εκτέλεσης.

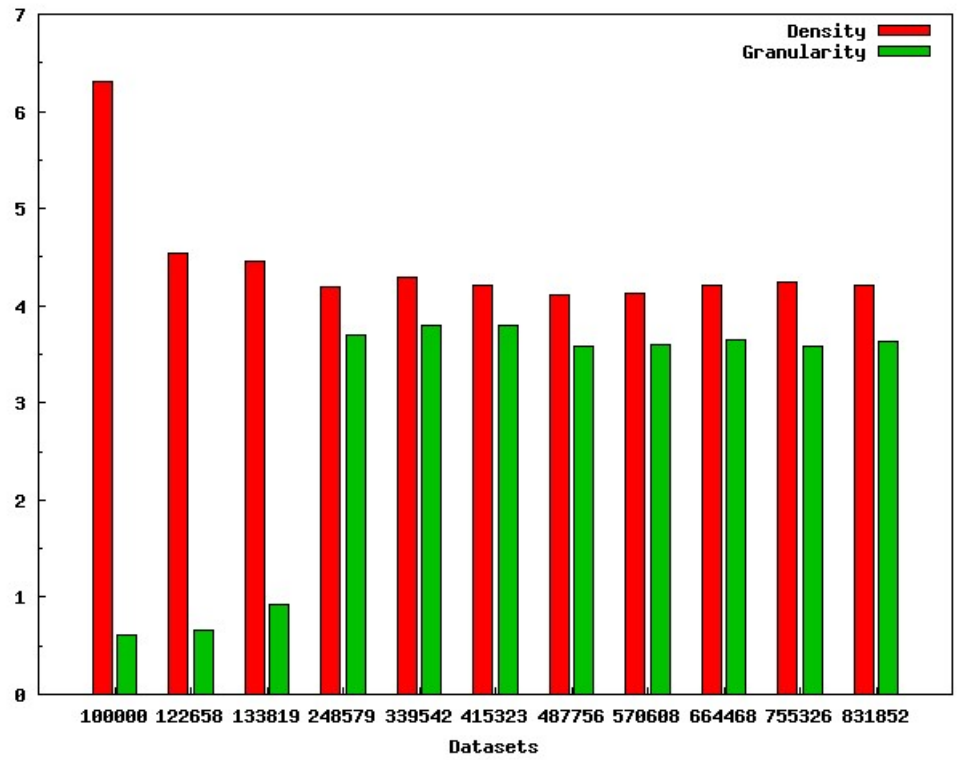
Ο λόγος του χρόνου υπολογισμών προς το χρόνο επικοινωνίας απεικονίζεται στο σχήμα 6.20. Ο χρόνος επικοινωνίας μετρήθηκε με θεωρητικό τρόπο, όπως εξηγείται στο παράρτημα Β′. Παρατηρείται ότι όσο αυξάνεται ο όγκος των συνόλων δεδομένων, η κοκκότητα τείνει να σταθεροποιηθεί περίπου στην τιμή 3,6. Αυτό είναι μια ένδειξη ότι η σχέση μεταξύ του χρόνου επικοινωνίας και του χρόνου υπολογισμών παραμένει ίδια όσο τα σύνολα δεδομένων έχουν την ίδια πυκνότητα. Η σχέση της κοκκότητας με την πυκνότητα του συνόλου δεδομένων απεικονείζεται στο ραβδόγραμμα του σχήματος 6.21.

6.5.3 Συγκριτική παρουσίαση των αποτελεσμάτων.

Σε αυτή την ενότητα παρουσιάζονται συνοπτικά και συνεκτιμούνται τα αποτελέσματα της πειραματικής μελέτης των εφαρμογών παραλληλοποίησης του αλγόριθμου slope one. Στους



Σχήμα 6.20: Χρόνος υπολογισμών προς χρόνο επικοινωνίας για την υβριδική υλοποίηση σε ετερογενή συστοιχία.



Σχήμα 6.21: Πυκνότητα και κοκκότητα ανά σύνολο δεδομένων για την υβριδική υλοποίηση σε ετερογενή συστοιχία.

πίνακες 6.2 και 6.3 φαίνεται ο συνολικός χρόνος εκτέλεσης της εφαρμογής με χρήση OpenMP και της υβριδικής εφαρμογής αντίστοιχα, συναρτήσει του πλήθους νημάτων που χρησιμοποιήθηκαν, για το σύνολο δεδομένων MovieLens 100k. Για το ίδιο σύνολο δεδομένων, η υβριδική εφαρμογή στην ετερογενή συστοιχία που αποτελείται από 30 νήματα συνολικά χρειάστηκε 14.068586 sec.

Παρατηρείται ότι η εφαρμογή με OpenMP είναι γρηγορότερη από την υβριδική εφαρμογή, και απαιτούνται επιπλέον πόροι στο σύστημα στο οποίο εκτελείται το πείραμα για να μπορέσουν να επιτευχθούν ίδιοι χρόνοι. Για παράδειγμα, στην εφαρμογή με OpenMP, με χρήση 8 νημάτων επιτυγχάνεται συνολικός χρόνος εκτέλεσης ίσος με 21,4 sec, ενώ στην υβριδική εφαρμογή απαιτείται μεγαλύτερος αριθμός νημάτων για να επιτευχθεί ο ίδιος χρόνος.

Η εφαρμογή της υβριδικής υλοποίησης σε ετερογενές σύστημα είναι δυνατό να επιφέρει παραπλήσια αποτελέσματα με την εφαρμογή με OpenMP, χωρίς να απαιτείται τόσο μεγάλη αύξηση των υπολογιστικών πόρων του συστήματος. Στην ετερογενή συστοιχία, με χρήση τριάντα νημάτων επιτυγχάνεται ίδιο αποτέλεσμα με αυτό που επιτυγχάνεται στην εφαρμογή με OpenMP με δεκαέξι νήματα. Ωστόσο, δεν αποκλείεται το ενδεχόμενο η υβριδική εφαρμογή σε ετερογενές σύστημα να παρουσιάσει ακόμα μικρότερους χρόνους, αν χρησιμοποιηθεί ισχυρότερος υπολογιστής στο ρόλο του συντονιστή, αφού όλες οι προβλέψεις παράγονται σε αυτόν.

Αριθμός νημάτων	**Συνολικός χρόνος**
2	71.534929
4	38.224012
6	27.231218
8	21.441523
10	20.575978
12	19.334583
14	18.095302
16	14.700209

Πίνακας 6.2: Εφαρμογή με OpenMP.

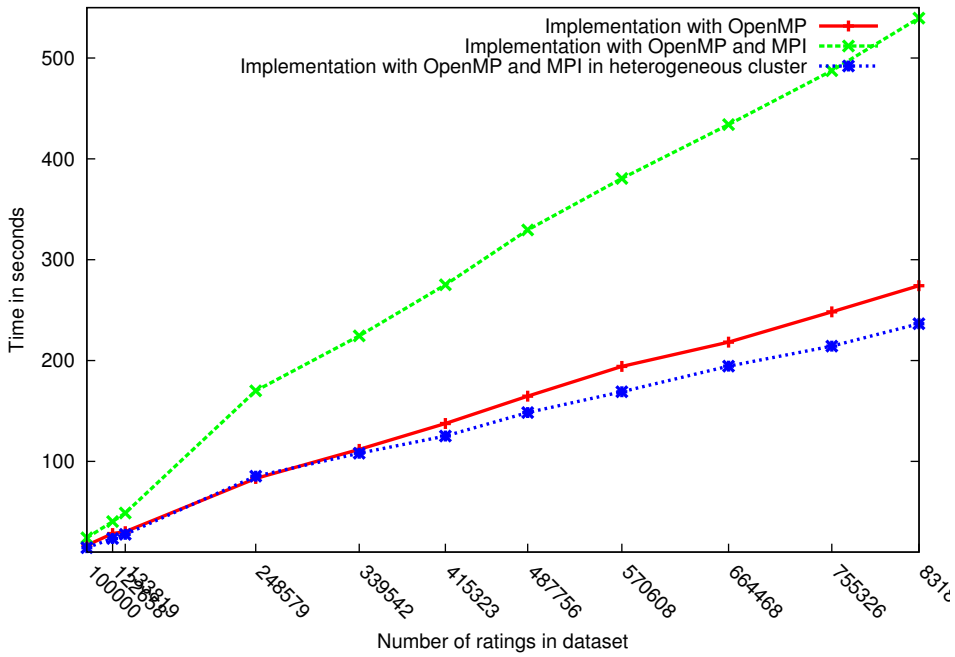
Αριθμός νημάτων	**Συνολικός χρόνος**
8	35.379519
16	25.889350
24	24.472460
32	23.945617
40	23.843781

Πίνακας 6.3: Υβριδική Εφαρμογή.

Στο σχήμα 6.22 φαίνεται ο χρόνος εκτέλεσης των υλοποιήσεων συναρτήσει του μεγέθους του συνόλου δεδομένων που χρησιμοποιείται. Με κόκκινο χρώμα αναπαρίσταται ο χρόνος της εφαρμογής με OpenMP, με πράσινο ο χρόνος της υβριδικής εφαρμογής και με μπλέ χρώμα αναπαρίσταται ο χρόνος της υβριδικής υλοποίησης στην ετερογενή συστοιχία. Οι εφαρμογές χρησιμοποιούν διαφορετικό πλήθος νημάτων, αξιοποιώντας έτσι πλήρως τους πόρους του

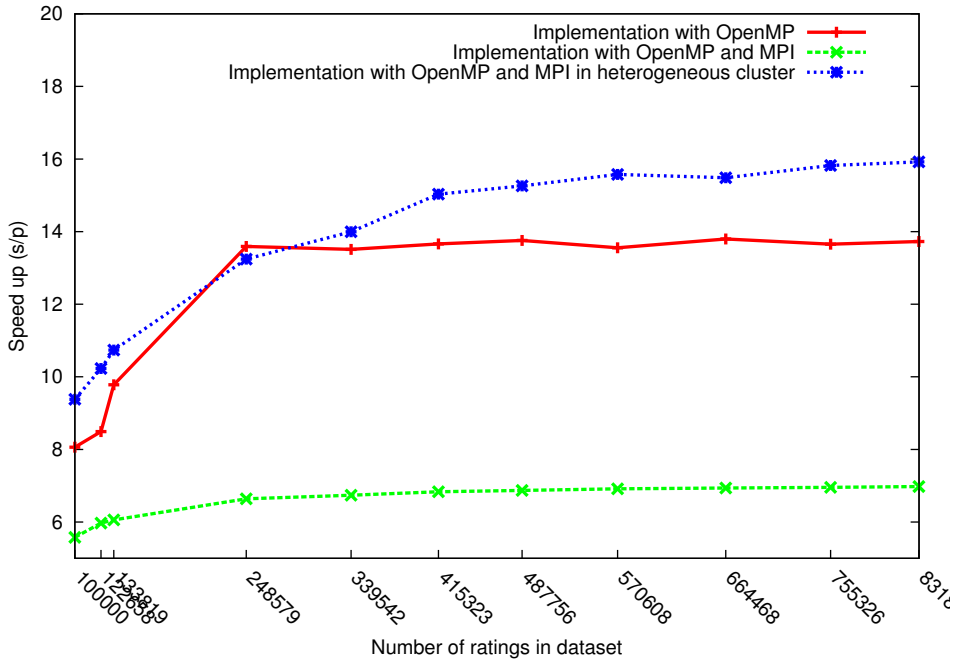
συστήματος στο οποίο εφαρμόζονται. Τα αποτελέσματα στο σχήμα 6.22 αφορούν το μέγιστο αριθμό νημάτων που είναι δυνατό να χρησιμοποιηθεί σε κάθε εφαρμογή. Η εφαρμογή με OpenMP αξιοποιεί 16 νήματα, η υβριδική εφαρμογή περίπου 36 και η υβριδική στην ετερογενή συστοιχία αξιοποιεί 30. Παρατηρείται ότι η υβριδική εφαρμογή στο ετερογενές σύστημα υπολογιστών παρουσιάζει ολοένα και καλύτερους χρόνους από τις υπόλοιπες εφαρμογές, όσο μεγαλώνουν τα σύνολα δεδομένων. Ωστόσο, το ιδανικό θα ήταν να μετρηθούν οι χρόνοι αυτοί με χρήση του ίδιου πλήθους νημάτων σε κάθε εφαρμογή, έτσι ώστε να γίνουν πιό εμφανή τα πλεονεκτήματα και οι αδυναμίες κάθε υλοποίησης.

Επιπλέον, από το διάγραμμα του σχήματος 6.23 στο οποίο απεικονίζεται ο λόγος του ακολουθιακού προς τον παράλληλο χρόνο εκτέλεσης σε κάθε σύνολο δεδομένων, προκύπτει ότι με την εφαρμογή της υβριδικής υλοποίησης στην ετερογενή συστοιχία επιτυγχάνεται εώς και 15.9×καλύτερη επιτάχυνση. Επίσης, παρατηρείται καί στα δύο διαγράμματα (6.22 και 6.23) ότι η υβριδική υλοποίηση παρουσιάζει σημαντική διαφορά από τις άλλες δύο. Αυτό οφείλεται στο χρόνο αναμονής που συμπεριλαμβάνεται στην επικοινωνία σε αυτή την εφαρμογή, ο οποίος δεν υπάρχει ή είναι πολύ μικρότερος στις άλλες υλοποιήσεις. Στην εφαρμογή του υβριδικού κώδικα σε ετερογενές σύστημα, ο χρόνος κατά τον οποίο ο συντονιστής παραμένει άεργος ελαχιστοποιείται σημαντικά επειδή χρησιμοποιούνται ως εργαζόμενοι και πιό ισχυροί υπολογιστές από το συντονιστή. Έτσι, εφόσον αυτοί έχουν μικρότερο χρόνο απόκρισης από τους υπολογιστές που χρησιμοποιήθηκαν στην ομογενή συστοιχία, ο συντονιστής θα αναμένει για μικρότερο χρονικό διάστημα μέχρι να λάβει μήνυμα ότι κάποιος από τους εργαζόμενους είναι διαθέσιμος να αναλάβει νέα εργασία.



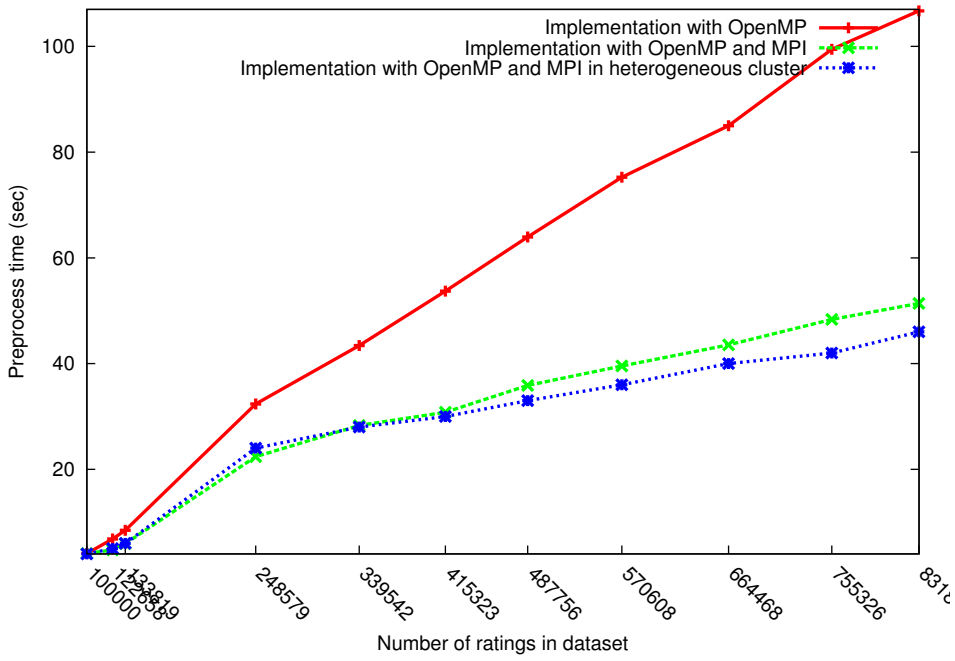
Σχήμα 6.22: Συνολικός χρόνος εκτέλεσης σε κάθε σύνολο δεδομένων.

Στο σχήμα 6.24 φαίνεται ο συνολικός χρόνος της φάσης προεπεξεργασίας των δεδομένων για κάθε υλοποίηση ανά σύνολο δεδομένων. Ο χρόνος αυτός, όπως είναι αναμενόμενο, αυξάνεται όσο μεγαλύτερος είναι ο όγκος των δεδομένων προς επεξεργασία. Παρατηρείται ότι η εφαρμογή με OpenMP απαιτεί μεγαλύτερο χρόνο προεπεξεργασίας πριν μεταβεί στην παραγωγή των προβλέψεων από ότι οι υβριδικές εφαρμογές. Στο σχήμα 6.25 φαίνεται ο αριθμός των προβλέψεων και των προβλέψεων με βάρη που παράγονται ανά δευτερόλεπτο σε

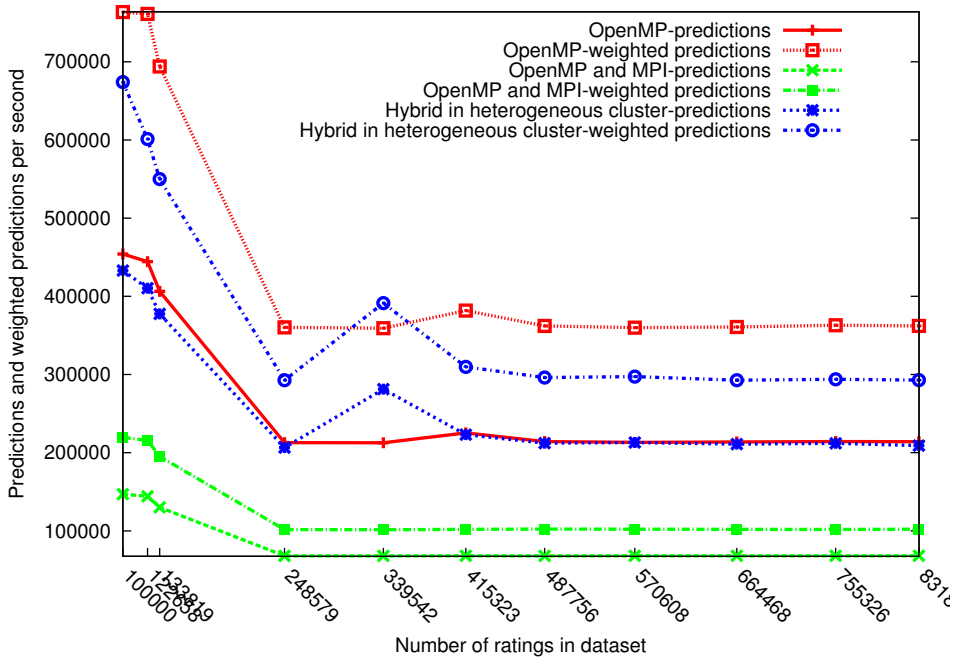


Σχήμα 6.23: Επιτάχυνση των τριών υλοποιήσεων.

κάθε εφαρμογή για κάθε σύνολο δεδομένων. Παράγονται περισσότερες προβλέψεις με βάρη ανά δευτερόλεπτο στην εφαρμογή με OpenMP, και σχεδόν ο ίδιος αριθμός προβλέψεων στην υβριδική εφαρμογή σε ετερογενή συστοιχία και στην εφαρμογή με OpenMP.



Σχήμα 6.24: Χρόνος φάσης προεπεξεργασίας για τις τρείς υλοποιήσεις.



Σχήμα 6.25: Προβλέψεις και προβλέψεις με βάρη ανά δευτερόλεπτο και ανά σύνολο δεδομένων για κάθε υλοποίηση.

Κεφάλαιο 7

Συμπεράσματα

Στην ενότητα αυτή γίνεται μια σύντομη ανασκόπηση των προηγούμενων κεφαλαίων και παρουσιάζονται τα σημαντικότερα συμπεράσματα που προέκυψαν καθώς και προτάσεις για μελλοντική εργασία.

Στα παραπάνω κεφάλαια έγινε περιγραφή των βασικών εννοιών των συστημάτων συστάσεων και των κυριότερων αλγορίθμων συνεργατικής διήθησης. Παρουσιάστηκαν οι μετρικές αξιολόγησής τους και τα κυριότερα προβλήματα που συναντώνται στις εφαρμογές των συστημάτων συστάσεων.

Στη συνέχεια δόθηκαν βασικοί ορισμοί και έννοιες της παράλληλης επεξεργασίας και οι μετρικές που εφαρμόζονται στην αξιολόγηση παράλληλων προγραμμάτων. Επιπλέον, περιγράφηκαν συνοπτικά οι τεχνολογίες παράλληλης επεξεργασίας που χρησιμοποιήθηκαν στην εφαρμογή που αναπτύχθηκε στο κεφάλαιο 6.

Οι τεχνικές παράλληλου προγραμματισμού εφαρμόζονται στα συστήματα συστάσεων με στόχο την ταχύτερη επεξεργασία του μεγάλου όγκου δεδομένων. Οι παράλληλοι αλγόριθμοι συνεργατικής διήθησης που υλοποιούνται τα τελευταία χρόνια περιγράφονται στο κεφάλαιο 5. Μεγάλη ποικιλία τεχνολογιών εφαρμόζονται, από τις καθιερωμένες όπως τα Posix Threads, το MPI και το OpenMP, ως και τις πιο σύγχρονες, όπως τα προγραμματιστικά πλαίσια Hadoop και GraphLab.

Η προσπάθεια σύγκρισης των αλγορίθμων αυτών αποδείχθηκε ανέφικτη διαδικασία διότι δεν υλοποιούνται υπό κοινό πλαίσιο. Χρησιμοποιούνται διαφορετικά σύνολα δεδομένων και διαφορετικές μετρικές για την αξιολόγησή τους. Οι μετρικές που χρησιμοποιούνται περισσότερο είναι το RMSE, η επιτάχυνση και ο χρόνος εκτέλεσης. Όμως δεν έχουν μετρηθεί σε όλες τις εφαρμογές. Η κλιμάκωση που είναι σημαντική μετρική για την εκτίμηση της απόδοσης ενός αλγορίθμου σε μεγάλα σύνολα δεδομένων, όπως αυτά που χρησιμοποιούν τα συστήματα συστάσεων, μετράται μόνο σε μία εφαρμογή. Έτσι, η συγκεντρωτική αξιολόγηση και η σύγκριση των παράλληλων αλγορίθμων συνεργατικής διήθησης αποτελεί ένα ανοιχτό πεδίο για περαιτέρω έρευνα, το οποίο θα μπορεί να αποτελέσει βάση για την υλοποίηση νέων παράλληλων αλγορίθμων, αναδεικνύοντας τα καταλληλότερα πεδία αλγορίθμων για παραλληλοποίηση και καταγράφοντας τις σύγχρονες τάσεις.

Τέλος, δίνεται έμφαση στην περιγραφή του αλγορίθμου slope one, εφαρμόζονται τεχνικές παραλληλοποίησης σε αυτόν και πραγματοποιείται πειραματική μελέτη τους. Εξετάζονται

δύο παράλληλες εφαρμογές του. Η πρώτη με χρήση OpenMP και η δεύτερη με ταυτόχονη χρήση OpenMP και MPI. Η υβριδική εφαρμογή αξιολογείται υπό το πλαίσιο δύο διαφορετικών υπολογιστικών συστημάτων. Μίας ομογενούς συστοιχίας υπολογιστών, όπου όλοι οι υπολογιστές είναι της ίδιας δυναμικότητας και μίας ετερογενούς.

Τα αποτελέσματα της πειραματικής μελέτης των προγραμμάτων παραλληλοποίησης του αλγόριθμου slope one έδειξαν ότι μπορεί να επιτευχθεί μεγάλη βελτίωση στο χρόνο εκτέλεσής τους με την εφαρμογή τεχνικών παραλληλοποίησης. Η πρώτη εφαρμογή με OpenMP είναι ώς και 9,5 φορές ταχύτερη από τον ακολουθιακό κώδικα για το σύνολο δεδομένων MovieLens 100k, η υβριδική εφαρμογή περίπου 5,5 φορές ταχύτερη, και η υβριδική εφαρμογή σε ετερογενή συστοιχία είναι επίσης περίπου 9,5 φορές ταχύτερη της ακολουθιακής.

Τα αποτελέσματα αυτά επιδέχονται περαιτέρω βελτίωση διότι υπάρχουν σημεία στον κώδικα των υλοποιήσεων που μπορούν να διαμορφωθούν με τέτοιο τρόπο ώστε να επιλυθούν τα προβλήματα χώρου που παρουσιάζονται και να βελτιωθεί η ταχύτητα εκτέλεσής τους. Ένα σημαντικό πρόβλημα στον κώδικα και των δύο εφαρμογών είναι η χρήση μεγάλων και αραιών πινάκων, που μπορεί να επιλυθεί χρησιμοποιώντας κάποια άλλη δομή δεδομένων όπως structs ή πίνακες συσχετίσεων (hashmaps). Η υβριδική εφαρμογή παρουσιάζει μερικά ακόμα προβλήματα που η αντιμετώπισή τους μπορεί να επιφέρει βελτίωση των αποτελεσμάτων. Ένα από αυτά είναι το περιεχόμενο του τελευταίου πακέτου που αποστέλλει ο συντονιστής. Ένα μεγάλο μέρος του πακέτου αυτού είναι δυνατό να περιέχει άχρηστα δεδομένα, λόγω του σταθερού μεγέθους των πακέτων, τα οποία καθυστερούν άσκοπα την εκτέλεση της εφαρμογής. Το πρόβλημα αυτό θα μπορούσε να επιλυθεί με την εφαρμογή δυναμικής κατανομής πακέτων δυναμικού μεγέθους. Δηλαδή, ρυθμίζοντας το μέγεθος των πακέτων σε κάθε αποστολή ανάλογα με το μέγεθος των δεδομένων ή ακόμα και ανάλογα με την υπολογιστική ισχύ του παραλήπτη. Επίσης, η εφαρμογή εργαλείων ανάλυσης της απόδοσης παράλληλων εφαρμογών, όπως αυτά που περιγράφονται στα [33] και [21], μπορεί να χρησιμοποιηθεί για τον έλεγχο προβληματικών σημείων στον κώδικα, όπως το σημείο στο οποίο πραγματοποιείται αναμονή στο συντονιστή.

Ωστόσο, λαμβάνοντας υπόψη τη βελτίωση που επιφέρουν οι παράλληλες υλοποιήσεις του αλγορίθμου slope one, γίνεται εύκολα αντιληπτό ότι η περαιτέρω ενασχόληση για τη βελτιστοποίησή τους δε θα είναι μάταιη. Επίσης, η συνεχής και αλματώδης αύξηση του όγκου των συνόλων δεδομένων και η επιτακτική ανάγκη για την άμεση, σχεδόν στιγμιαία, παραγωγή αποτελεσμάτων, καθώς και η βελτίωση της ποιότητας των αποτελεσμάτων, αποδεικνύουν ότι η βελτιστοποίηση υπάρχοντων αλγορίθμων ή ακόμα και η επινόηση νέων, αποτελούν ανοιχτό πεδίο μελέτης και έρευνας, είτε οι αλγόριθμοι αυτοί ανήκουν στον τομέα της συνεργατικής διήθησης, είτε σε άλλους εξίσου σημαντικούς τομείς της Μηχανικής Μάθησης.

Παράρτημα Α΄

Τα σύνολα δεδομένων MovieLens της GroupLens Research.

Η GroupLens Research διαθέτει τα σύνολα δεδομένων MovieLens [40] τα αποία αποτελούνται από δεδομένα που συλλέχθηκαν σε διάφορες χρονικές περιόδους.Τρία σύνολα δεδομένων είναι διαθέσιμα, το MovieLens 100k που αποτελείται από 100000 αξιολογήσεις (ratings) ταινιών από 1000 χρήστες σε σύνολο 1700 ταινιών, το MovieLens 1M με 1 εκατομμύριο αξιολογήσεις 4000 ταινιών από 6000 χρήστες και το MovieLens 10M με 10 εκατομμύρια αξιολογήσεις σε 10000 ταινίες από 72000 χρήστες.

Όλες οι αξιολογήσεις γίνονται σε κλίμακα από 1 εώς 5 και κάθε χρήστης έχει αξιολογήσει τουλάχιστον 20 ταινίες. Στο MovieLens 100k οι 100000 αξιολογήσεις γίνονται από 943 χρήστες για 1682 ταινίες. Στο MovieLens 1M περιέχονται 1000209 αξιολογήσεις από 6040 χρήστες για 3952 ταινίες. Τα δεδομένα είναι τυχαία κατανεμημένα και παρουσιάζονται χωρισμένα με | στη μορφή :

user id| item id| rating | timestamp

Τα timestamp είναι δευτερόλεπτα unix από την 1/1/1970 UTC. Η αρίθμηση χρηστών και ταινιών ξεκινάει από το 1. Παρέχονται επίσης απλά δημογραφικά δεδομένα για τους χρήστες με τη μορφή :

user id | age | gender | occupation | zip code

και για τις ταινίες:

movie id | movie title | release date | video release date | IMDb URL | unknown | Action | Adventure | Animation | Children's | Comedy | Crime | Documentary | Drama | Fantasy | Film-Noir | Horror | Musical | Mystery | Romance | Sci-Fi | Thriller | War | Western |

Το MovieLens 10M περιέχει 10000054 αξιολογήσεις και 95580 tags εφαρμόζονται σε 10681 ταινιές από 71567 χρήστες. Σε αυτό το σύνολο δεδομένων δεν περιλαμβάνονται δημογραφικά στοιχεία. Τα δεδομένα στο αρχείο αυτό είναι ταξινομημένα ανά UserID και ανά MovieID και οι αξιολογήσεις γίνονται σε κλίμακα από το 1 εώς το 5, ανά 0.5, και παρουσιάζονται με την εξής μορφή:

UserID::MovieID::Rating::Timestamp.

Στα πειράματα που αναλύονται στο κεφάλαιο 6 χρησιμοποιήθηκαν τα συνολα δεδομένων που φαίνονται στον παρακάτω πίνακα.Το σύνολο u.data είναι το MovieLens 100k και τα σύνολα data 1 εώς data 10 είναι υποσύνολα του συνόλου MovieLens 1M που δημιουργήθηκαν λαμβάνοντας κάθε φορά τις πρώτες εγγραφές του συνόλου, έτσι ώστε να αυξάνεται σταδιακά το πλήθος των εκτιμήσεων και των χρηστών.

File	Users	Items	Ratings
u.data	943	1682	100000
data1	1500	3952	248579
data2	1500	2000	133819
data3	1500	1799	122658
data4	2000	3952	339542
data5	2500	3952	415323
data6	3000	3952	487756
data7	3500	3952	570608
data8	4000	3952	664468
data9	4500	3952	755326
data10	5000	3952	831852

Πίνακας Α΄.1: Στοιχεία συνόλων δεδομένων

File	Density %
u.data	6,30
data1	4,19
data2	4,46
data3	4,54
data4	4,29
data5	4,20
data6	4,11
data7	4,12
data8	4,20
data9	4,24
data10	4,20

Πίνακας Α΄.2: Πυκνότητα συνόλων δεδομένων

Παράρτημα Β′

Θεωρητικός υπολογισμός του χρόνου επικοινωνίας.

Για το θεωρητικό υπολογισμό του χρόνου επικοινωνίας δημιουργήθηκε μια υλοποίηση της μεθόδου ping pong, στην οποία δύο διεργασίες στέλνουν και λαμβάνουν μηνύματα ίδιου μεγέθους με αυτά που διαχειρίζεται η υβριδική εφαρμογή του κεφαλαίου 6. Τα μηνύματα που αποστέλλονται είναι τα εξής:

- Ένα μήνυμα που περιέχει έναν ακέραιο, το οποίο αντιστοιχεί στη μεταβλητή sth που εκφράζει την ετοιμότητα του εργαζομένου να λάβει νέα δεδομένα.
- Ένα μήνυμα μεγέθους users2 ×items ακεραίων, που αντιστοιχεί στα πακέτα δεδομένων που στέλνει ο συντονιστής στους εργαζόμενους (testratings) και στα πακέτα που περιέχουν τη συνθήκη τερματισμού εργασίας.
- Ένα μήνυμα μεγέθους items ×items ακεραίων, που αντιστοιχεί στον πίνακα denominator2.
- Ένα μήνυμα μεγέθους items ×items πραγματικών αριθμών, που αντιστοιχεί στον πίνακα deviation2.

Οι χρόνοι που απαιτούνται για την αποστολή και λήψη των παραπάνω πακέτων φαίνονται στον πίνακα που ακολουθεί.

Μήνυμα	**Χρόνος**	**Πλήθος μηνυμάτων**
Μεταβλητή για αποστολή σήματος ετοιμότητας(sth)	0,002137	p
Πακέτα δεδομένων (testratings)	0,012240	p
Σήμα τερματισμού εργασίας (dietags)	0.012240	w
Αποτελέσματα (denominator2)	0.192619	w
Αποτελέσματα (deviation2)	0.192784	w

Πίνακας Β′.1: Χρόνος αποστολής και λήψης μηνυμάτων.

Η τελευταία στήλη του πίνακα Β′.1 δείχνει πόσες φορές θα πρέπει να αποσταλεί το καθένα από τα μηνύματα. Με p συμβολίζεται το πλήθος των πακέτων στα οποία διασπάται το σύνολο

δεδομένων και με w το πλήθος των εργαζομένων της συστοιχίας. Συνεπώς, η επικοινωνία υπολογίζεται σύμφωνα με τον παρακάτω τύπο:

$$(t_{sth} + t_{testratings}) \cdot p + (t_{dietags} + t_{denominator2} + t_{deviation2}) \cdot w$$

Στην περίπτωση του συνόλου δεδομένων MovieLens 100k (u.data) που περιέχει 943 χρήστες, για πακέτα των 100 χρηστών (users2=100) απαιτούνται δέκα πακέτα για την ολοκλήρωση της αποστολής των δεδομένων, οπότε $p = 10$. Οι χρόνοι επικοινωνίας υπολογίζονται για πλήθος εργαζομένων w ίσο με 1,3,5,7 και 9. Οι χρόνοι αυτοί φαίνονται στον επόμενο πίνακα (Β′.2).

Πλήθος υπολογιστών συστοιχίας	**Χρόνος Επικοινωνίας**
2	0,541413
4	1,336699
6	2,131985
8	2,927271
10	3,722557

Πίνακας Β′.2: Θεωρητικά υπολογισμένος χρόνος επικοινωνίας για το σύνολο δεδομένων MovieLens 100k.

Παράρτημα Γ′

Πίνακες αριθμητικών αποτελεσμάτων.

Σε αυτό το παράρτημα παρουσιάζονται οι πίνακες με τα αριθμητικά αποτελέσματα που προέκυψαν από την εκτέλεση των πειραμάτων. Οι πίνακες Γ′.1 και Γ′.2 αφορούν την εφαρμογή με OpenMP, και οι χρόνοι αφορούν την εκτέλεση της εφαρμογής με 16 νήματα. Οι πίνακες από τον Γ′.3 ώς τον Γ′.24 περιέχουν τα αποτελέσματα της υβριδικής εφαρμογής για τα διάφορα σύνολα δεδομένων, και ο πίνακας Γ′.25 περιέχει τα αποτελέσματα της υριδικής εφαρμογής στην ετερογενή συστοιχία.

Σύνολο δεδομένων	**Συνολικός Χρόνος**	**Χρόνος Προεπεξεργασίας**
u.data	16.717665	4.063877
data1	83.026942	32.354549
data2	30.237320	8.475188
data3	28.324049	6.828610
data4	111.896601	43.424517
data5	137.574666	53.710728
data6	164.582891	63.959317
data7	194.168638	75.266237
data8	218.208604	85.001731
data9	248.259966	99.452063
data10	274.223358	106.729780

Πίνακας Γ′.1: Συνολικός χρόνος και χρόνος προεπεξεργασίας της εφαρμογής με OpenMP.

Σύνολο δεδομένων	Χρόνος Προβλέψεων	Χρόνος Προβλέψεων με βάρη
u.data	3.271999	1.946093
data1	26.689507	15.764811
data2	7.054487	4.129264
data3	5.796595	3.383605
data4	35.577054	21.056151
data5	44.221493	26.096553
data6	53.063218	31.390168
data7	62.220497	36.838085
data8	70.852742	41.969896
data9	79.510694	46.913174
data10	88.444879	52.272644

Πίνακας Γ′.2: Συνολικός χρόνος προβλέψεων και προβλέψεων με βάρη της εφαρμογής με OpenMP.

Πλήθος Κόμβων	Συνολικός Χρόνος	Χρόνος Επικοινωνίας	Υπολογισμοί Συντονιστή	Χρόνος Προεπεξεργασίας
2	35.379519	14.887844	0.119484	15.077770
4	25.889350	5.491751	0.068878	5.627910
6	24.472460	4.037514	0.100983	4.209857
8	23.945617	3.566252	0.132511	3.767121
10	23.843781	3.426488	0.165205	3.662743

Πίνακας Γ′.3: Αποτελέσματα της υβριδικής εφαρμογής για το σύνολο δεδομένων u.data.

Πλήθος Κόμβων	Μέγιστοι Υπολογισμοί Εργαζομένων	Χρόνος Προβλέψεων	Χρόνος Προβλέψεων Με Βάρη
2	1.632724	10.116080	6.783931
4	1.630888	10.097261	6.800228
6	1.631193	10.124105	7.015996
8	1.638301	10.086992	6.746616
10	1.687514	10.099085	6.777072

Πίνακας Γ′.4: Αποτελέσματα της υβριδικής εφαρμογής για το σύνολο δεδομένων u.data.

Πλήθος Κόμβων	Συνολικός Χρόνος	Χρόνος Επικοινωνίας	Υπολογισμοί Συντονιστή	Χρόνος Προεπεξεργασίας
2	280.108544	132.658084	0.194947	133.011499
4	194.605693	46.825925	0.365223	47.345925
6	179.387225	31.723432	0.538049	32.414422
8	174.953264	26.820572	0.706883	27.685737
10	172.389826	23.672123	0.952889	24.778696
12	169.938360	21.190579	1.045132	22.394827

Πίνακας Γ′.5: Αποτελέσματα της υβριδικής εφαρμογής για το σύνολο δεδομένων data1.

Πλήθος Κόμβων	Μέγιστοι Υπολογισμοί Εργαζομένων	Χρόνος Προβλέψεων	Χρόνος Προβλέψεων Με Βάρη
2	8.740834	83.501734	55.871998
4	8.816592	83.609608	55.931999
6	8.826091	83.504227	55.903996
8	8.925811	83.464979	55.912003
10	8.848848	83.436089	55.879996
12	8.819056	83.496473	55.867996

Πίνακας Γ′.6: Αποτελέσματα της υβριδικής εφαρμογής για το σύνολο δεδομένων data1.

Πλήθος Κόμβων	Συνολικός Χρόνος	Χρόνος Επικοινωνίας	Υπολογισμοί Συντονιστή	Χρόνος Προεπεξεργασίας
2	77.917185	34.675076	0.055004	34.829173
4	55.202094	12.314600	0.099763	12.500497
6	51.383983	8.263098	0.145697	8.501106
8	50.316744	7.003574	0.255073	7.346121
10	49.295539	6.178905	0.263858	6.532277
12	48.817030	5.562783	0.279394	5.934219

Πίνακας Γ′.7: Αποτελέσματα της υβριδικής εφαρμογής για το σύνολο δεδομένων data2.

Πλήθος Κόμβων	Μέγιστοι Υπολογισμοί Εργαζομένων	Χρόνος Προβλέψεων	Χρόνος Προβλέψεων Με Βάρη
2	2.290845	21.946440	14.706987
4	2.356139	21.973335	14.655182
6	2.320369	21.979690	14.615995
8	2.515318	21.968224	14.635997
10	2.313592	21.966418	14.647995
12	2.311475	22.030547	14.679207

Πίνακας Γ′.8: Αποτελέσματα της υβριδικής εφαρμογής για το σύνολο δεδομένων data2.

Πλήθος Κόμβων	Συνολικός Χρόνος	Χρόνος Επικοινωνίας	Υπολογισμοί Συντονιστή	Χρόνος Προεπεξεργασίας
2	63.798264	27.859233	0.358471	28.313030
4	45.739186	9.764288	0.261844	10.106223
6	42.680172	6.666066	0.333375	7.080783
8	41.382123	5.621231	0.155807	5.859849
10	40.844515	4.960195	0.191499	5.232512
12	40.292530	4.459761	0.227102	4.768725

Πίνακας Γ′.9: Αποτελέσματα της υβριδικής εφαρμογής για το σύνολο δεδομένων data3.

Πλήθος Κόμβων	Μέγιστοι Υπολογισμοί Εργαζομένων	Χρόνος Προβλέψεων	Χρόνος Προβλέψεων Με Βάρη
2	1.845596	17.955753	11.959996
4	1.836312	17.879936	11.896399
6	1.846727	17.968585	11.951736
8	1.844421	17.939519	11.927351
10	1.852820	17.907616	11.957111
12	1.874231	17.861218	11.943997

Πίνακας Γ′.10: Αποτελέσματα της υβριδικής εφαρμογής για το σύνολο δεδομένων data3.

Πλήθος Κόμβων	Συνολικός Χρόνος	Χρόνος Επικοινωνίας	Υπολογισμοί Συντονιστή	Χρόνος Προεπεξεργασίας
2	373.858623	176.376084	0.201131	176.803506
4	260.429346	63.056236	0.373342	63.639969
6	237.036006	40.177605	0.698663	41.086042
8	230.821505	32.638941	0.716398	33.564846
10	225.288995	27.775250	0.883894	28.868656
12	224.385501	27.027294	1.055200	28.291221

Πίνακας Γ′.11: Αποτελέσματα της υβριδικής εφαρμογής για το σύνολο δεδομένων data4.

Πλήθος Κόμβων	Μέγιστοι Υπολογισμοί Εργαζομένων	Χρόνος Προβλέψεων	Χρόνος Προβλέψεων Με Βάρη
2	8.897781	111.813735	74.797817
4	8.931924	111.301735	74.584000
6	8.825802	111.340333	74.409538
8	8.863973	111.312518	74.679759
10	8.852968	111.225928	74.580000
12	8.935980	111.286711	74.615997

Πίνακας Γ′.12: Αποτελέσματα της υβριδικής εφαρμογής για το σύνολο δεδομένων data4.

Πλήθος Κόμβων	Συνολικός Χρόνος	Χρόνος Επικοινωνίας	Υπολογισμοί Συντονιστή	Χρόνος Προεπεξεργασίας
2	465.455338	220.281546	0.207533	220.768062
4	324.244110	79.242555	0.721252	80.218602
6	294.008851	49.129441	0.696181	50.078734
8	284.819501	38.853563	0.720788	39.826207
10	279.492313	33.430117	0.962822	34.645364
12	275.076211	29.481878	1.060069	30.798317

Πίνακας Γ′.13: Αποτελέσματα της υβριδικής εφαρμογής για το σύνολο δεδομένων data5.

Πλήθος Κόμβων	Μέγιστοι Υπολογισμοί Εργαζομένων	Χρόνος Προβλέψεων	Χρόνος Προβλέψεων Με Βάρη
2	8.745126	138.983348	92.722443
4	8.775300	139.023032	92.835993
6	8.780427	138.990255	92.919993
8	8.888865	138.851344	92.744001
10	9.417167	138.918613	92.876000
12	8.981078	138.981247	92.803997

Πίνακας Γ′.14: Αποτελέσματα της υβριδικής εφαρμογής για το σύνολο δεδομένων data5.

Πλήθος Κόμβων	Συνολικός Χρόνος	Χρόνος Επικοινωνίας	Υπολογισμοί Συντονιστή	Χρόνος Προεπεξεργασίας
2	557.544704	263.406296	0.213665	263.981028
4	384.379350	90.631952	0.385126	91.313077
6	351.495808	57.930177	0.559282	58.786799
8	339.389713	45.371596	0.726215	46.394130
10	331.371994	37.542473	0.956721	38.797152
12	329.464560	34.505461	1.069218	35.871992

Πίνακας Γ′.15: Αποτελέσματα της υβριδικής εφαρμογής για το σύνολο δεδομένων data6.

Πλήθος Κόμβων	Μέγιστοι Υπολογισμοί Εργαζομένων	Χρόνος Προβλέψεων	Χρόνος Προβλέψεων Με Βάρη
2	8.717234	167.091482	111.499995
4	8.793124	166.650193	111.460250
6	8.793588	166.759032	111.431999
8	8.774387	166.919970	111.196001
10	8.810813	166.840812	111.448003
12	8.784387	167.049561	111.191743

Πίνακας Γ′.16: Αποτελέσματα της υβριδικής εφαρμογής για το σύνολο δεδομένων data6.

Πλήθος Κόμβων	Συνολικός Χρόνος	Χρόνος Επικοινωνίας	Υπολογισμοί Συντονιστή	Χρόνος Προεπεξεργασίας
2	649.122483	307.509899	0.221101	308.095533
4	449.440644	106.901594	0.395416	107.638929
6	409.771254	66.922956	0.566398	67.833044
8	394.720976	51.283086	0.732997	52.359200
10	384.453837	43.507833	0.933902	44.787657
12	380.525852	38.082703	1.120397	39.547440

Πίνακας Γ′.17: Αποτελέσματα της υβριδικής εφαρμογής για το σύνολο δεδομένων data7.

Πλήθος Κόμβων	Μέγιστοι Υπολογισμοί Εργαζομένων	Χρόνος Προβλέψεων	Χρόνος Προβλέψεων Με Βάρη
2	8.687980	194.326259	129.317536
4	8.941786	194.374441	129.524002
6	8.827414	194.945689	130.144349
8	8.867864	194.745269	130.024004
10	8.833299	194.241367	129.696000
12	8.793050	194.445981	129.819992

Πίνακας Γ′.18: Αποτελέσματα της υβριδικής εφαρμογής για το σύνολο δεδομένων data7.

Πλήθος Κόμβων	Συνολικός Χρόνος	Χρόνος Επικοινωνίας	Υπολογισμοί Συντονιστή	Χρόνος Προεπεξεργασίας
2	742.951660	351.426677	0.226397	352.108267
4	513.590363	122.726725	0.416789	123.544724
6	465.396361	75.110803	0.653984	76.158861
8	449.263415	57.589785	0.742308	58.735420
10	439.016030	47.656853	0.914796	48.971637
12	434.008312	41.997358	1.150478	43.547749

Πίνακας Γ′.19: Αποτελέσματα της υβριδικής εφαρμογής για το σύνολο δεδομένων data8.

Πλήθος Κόμβων	Μέγιστοι Υπολογισμοί Εργαζομένων	Χρόνος Προβλέψεων	Χρόνος Προβλέψεων Με Βάρη
2	8.747226	222.783974	149.291751
4	8.848914	222.521152	148.663237
6	8.755205	222.991445	148.679999
8	8.850229	222.642200	148.767994
10	9.421801	222.594350	148.679999
12	8.841371	222.283120	148.628241

Πίνακας Γ′.20: Αποτελέσματα της υβριδικής εφαρμογής για το σύνολο δεδομένων data8.

Πλήθος Κόμβων	Συνολικός Χρόνος	Χρόνος Επικοινωνίας	Υπολογισμοί Συντονιστή	Χρόνος Προεπεξεργασίας
2	835.573102	396.056541	0.233006	396.757538
4	575.173455	133.755655	0.404808	134.610159
6	524.258113	84.193323	0.576130	85.217455
8	503.366514	64.149918	0.812584	65.409216
10	492.350958	53.260060	0.921293	54.630237
12	487.483525	46.804315	1.091989	48.347533

Πίνακας Γ′.21: Αποτελέσματα της υβριδικής εφαρμογής για το σύνολο δεδομένων data9.

Πλήθος Κόμβων	Μέγιστοι Υπολογισμοί Εργαζομένων	Χρόνος Προβλέψεων	Χρόνος Προβλέψεων Με Βάρη
2	8.786642	250.202415	167.275994
4	8.804262	250.410540	167.396496
6	8.802359	250.390813	167.419999
8	8.864529	250.424540	167.023998
10	8.895660	250.157751	167.507747
12	8.971907	250.007404	167.159996

Πίνακας Γ′.22: Αποτελέσματα της υβριδικής εφαρμογής για το σύνολο δεδομένων data9.

Πλήθος Κόμβων	Συνολικός Χρόνος	Χρόνος Επικοινωνίας	Υπολογισμοί Συντονιστή	Χρόνος Προεπεξεργασίας
2	928.959064	441.035282	0.238677	441.796159
4	637.606274	150.625740	0.409202	151.532712
6	580.879822	92.498778	0.756047	93.748393
8	556.548447	70.174471	0.751846	71.418280
10	543.778354	57.283122	0.972464	58.754012
12	539.580781	49.795766	1.097366	51.392063

Πίνακας Γ′.23: Αποτελέσματα της υβριδικής εφαρμογής για το σύνολο δεδομένων data10.

Πλήθος Κόμβων	Μέγιστοι Υπολογισμοί Εργαζομένων	Χρόνος Προβλέψεων	Χρόνος Προβλέψεων Με Βάρη
2	8.844484	277.206876	185.099748
4	8.779440	277.336155	185.867999
6	8.907091	277.508033	185.507999
8	8.913145	277.416542	185.227752
10	8.964046	278.759554	186.775999
12	8.953666	277.357141	185.467745

Πίνακας Γ′.24: Αποτελέσματα της υβριδικής εφαρμογής για το σύνολο δεδομένων data10.

Σύνολο Δεδομένων	**Συνολικός Χρόνος**	**Χρόνος Προεπεξεργασίας**	**Χρόνος Επικοινωνίας**
u.data	14,36671	4,158849	4,02964
data1	85,2239	24,089806	23,527717
data2	27,546623	6,446106	6,231255
data3	23,515332	5,325623	5,149683
data4	108,00658	28,08652	27,488792
data5	125,037171	30,467201	29,811919
data6	148,368264	33,628804	32,923777
data7	168,95198	36,02678	35,272561
data8	194,428803	40,551043	39,729871
data9	214,253129	42,754345	41,873646
data10	236,452386	46,275173	45,33791
Σύνολο Δεδομένων	**Χρόνος Προβλέψεων**	**Χρόνος Προβλέψ. Με Βάρη**	**Μέγ. Χρόνος Υπολ. Εργαζ.**
u.data	3,432209	2,204459	1,319116
data1	27,491281	19,402385	7,872836
data2	7,590426	5,211058	1,979323
data3	6,276093	4,284034	1,396246
data4	26,88717	19,330578	8,088889
data5	44,646391	32,165933	8,094165
data6	53,569547	38,395419	7,623236
data7	62,254804	44,594233	7,678182
data8	71,742535	51,741436	7,79161
data9	80,319114	57,949922	7,637606
data10	90,529653	64,635949	7,760389

Πίνακας Γ′.25: Αποτελέσματα της υβριδικής εφαρμογής στην ετερογενή συστοιχία.

Παράρτημα Δ΄

Αριθμητικό παράδειγμα του αλγόριθμου **Slope One.**

Το παράδειγμα αφορά την εκτέλεση του αλγορίμθου Slope One για τέσσερις χρήστες και πέντε αντικείμενα, όπου μόνο ένας χρήστης έχει δώσει αξιολόγηση για όλα τα αντικείμενα. Τα δεδομένα αποτελούνται από 14 εγγραφές και φαίνονται στον πίνακα Δ΄.1. Οι χρήστες αριθμούνται από το 1 ώς το 4, τα αντικείμενα από το 1 ώς το 5, και οι αξιολογήσεις κυμαίνονται από το 1 ώς το 5 σε ακέραια κλίμακα, αλλά στις προβλέψεις γίνονται αποδεκτές και πραγματικές τιμές.

Χρήστης	**Αντικείμενο**	**Αξιολόγηση**
1	1	2
1	2	4
1	3	3
1	4	1
1	5	5
2	1	4
2	2	3
2	4	3
2	5	2
3	3	1
3	4	1
3	5	5
4	1	2
4	5	1

Πίνακας Δ΄.1: Σύνολο Δεδομένων Παραδείγματος.

Από τα δεδομένα σχηματίζεται ο πίνακας ratings που φαίνεται παρακάτω. Στον πίνακα αυτό κάθε γραμμή αντιστοιχεί σε έναν χρήστη και κάθε στήλη σε ένα αντικείμενο. Επιλέγοντας το στοιχείο που βρίσκεται στην u γραμμή και στη j στήλη, λαμβάνεται η αξιολόγηση που έχει δώσει ο χρήστης u για το αντικείμενο j. Στην περίπτωση όπου δεν υπάρχει διαθέσιμη αξιολόγηση, η θέση στον πίνακα ratings περιέχει την τιμή 0.

$$ratings = \begin{pmatrix} 2 & 4 & 3 & 1 & 5 \\ 4 & 3 & 0 & 3 & 2 \\ 0 & 0 & 1 & 1 & 5 \\ 2 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Στη συνέχεια σχηματίζεται ο πίνακας numerator ο οποίος περιέχει για κάθε χρήστη u τις διαφορές των εκτιμήσεων των αντικειμένων i και j, $dif(i,j) = u_i - u_j$. Το εύρος των τιμών που μπορούν να πάρουν οι διαφορές αυτές είναι το διάστημα [-4,4]. Επιλέγεται ο αριθμός $10 \notin [-4,4]$ για να εκφράσει τις περιπτώσεις όπου δεν είναι δυνατό να υπολογισθεί η διαφορά, επειδή για ένα από τα δύο ή και για τα δύο αντικείμενα δεν υπάρχει αξιολόγηση. Επειδή η τιμή $0 \in [-4,4]$, δεν μπορεί να χρησιμοποιηθεί στις περιπτώσεις αυτές διότι θα δημιουργούταν σύγχυση για το πότε η διαφορά είναι 0 και πότε δεν είναι διαθέσιμη. Τα στοιχεία των υποπινάκων Δ′.2, Δ′.3, Δ′.4 και Δ′.5 αποτελούν τον πίνακα numerator.

item i \ item j	1	2	3	4	5
1	0	-2	-1	1	-3
2	2	0	1	3	-1
3	1	-1	0	2	-2
4	-1	-3	-2	0	-4
5	3	1	2	4	0

Πίνακας Δ′.2: Πίνακας διαφορών για τον χρήστη 1.

item i \ item j	1	2	3	4	5
1	0	1	10	1	2
2	-1	0	10	0	1
3	10	10	10	10	10
4	-1	0	10	0	1
5	-2	-1	10	-1	0

Πίνακας Δ′.3: Πίνακας διαφορών για τον χρήστη 2.

item i \ item j	1	2	3	4	5
1	10	10	10	10	10
2	10	10	10	10	10
3	10	10	0	0	-4
4	10	10	0	0	-4
5	10	10	4	4	0

Πίνακας Δ′.4: Πίνακας διαφορών για τον χρήστη 3.

item i \ item j	1	2	3	4	5
1	0	10	10	10	1
2	10	10	10	10	10
3	10	10	10	10	10
4	10	10	10	10	10
5	-1	10	10	10	0

Πίνακας Δ′.5: Πίνακας διαφορών για τον χρήστη 4.

Ο πίνακας συχνοτήτων denominator περιέχει πόσες φορές συναντάται εκτίμηση του αντικειμένου i και ταυτόχρονα συναντάται εκτίμηση του αντικειμένου j. Η κύρια διαγώνιός του δείχνει πόσες φορές συναντάται εκτίμηση για κάθε αντικείμενο και είναι ο μέγιστος αριθμός που μπορεί να εμφανισθεί σε κάθε γραμμή ή στήλη. Επίσης, ο πίνακας αυτός είναι συμμετρικός.

Ο πίνακας αποκλίσεων deviation σχηματίζεται τοποθετώντας στην (i,j) θέση του, το άθροισμα όλων των (i,j) στοιχείων των υποπινάκων του πίνακα numerator εκτός από τις θέσεις που περιέχουν την τιμή 10, και διαιρώντας με την τιμή της (i,j) θέσης του πίνακα denominator. Αν η τιμή αυτή είναι 0, τότε η αντίστοιχη θέση του πίνακα deviation παίρνει και αυτή την τιμή 0.

Οι πίνακες denominator και deviation απεικονίζονται παρακάτω, στους πίνακες Δ′.6 και Δ′.7

αντίστοιχα.

item i \ item j	1	2	3	4	5
1	3	2	1	2	3
2	2	2	1	2	2
3	1	1	2	2	2
4	2	2	2	3	3
5	3	2	2	3	4

Πίνακας Δ΄.6: Πίνακας συχνοτήτων.

item i \ item j	1	2	3	4	5
1	0	-1/2	-1	1	0
2	1/2	0	1	3/2	0
3	1	-1	0	1	-3
4	-1	-3/2	-1	0	-7/3
5	0	0	3	7/3	0

Πίνακας Δ΄.7: Πίνακας αποκλίσεων.

Στη συνέχεια υπολογίζονται οι προβλέψεις και οι προβλέψεις με βάρη για το χρήστη 3 και το αντικείμενο 2 και για τον χρήστη 4 και το αντικείμενο 4. Οι προβλέψεις υπολογίζονται σύμφωνα με τον τύπο

$$pred(u,j) = \bar{u} + \frac{1}{card(R_j)} \sum_{i \in R_j} dev_{j,i}$$

και οι προβλέψεις με βάρη από τον

$$pred(u,j) = \frac{\sum_{i \in S(u)-\{j\}} (dev_{j,i} + u_i) c_{j,i}}{\sum_{i \in S(u)-\{j\}} c_{j,i}}$$

στους τύπους αυτούς, $card(R_j)$ είναι το πλήθος όλων των αντικειμένων i εκτός του j, τα οποία ο χρήστης u έχει εκτιμήσει και υπάρχει στο σύνολο δεδομένων ταυτόχρονα εκτίμηση του i και του j. Το $c_{j,i}$ λαμβάνει την τιμή της θέσης (j,i) του πίνακα denominator.

Πρόβλεψη της αξιολόγησης του χρήστη 3 για το αντικείμενο 2.

$\bar{u} = \frac{7}{3}$

$card(R_2) = 3$

$\sum dev_{2,i} = dev_{2,3} + dev_{2,4} + dev_{2,5} = 1 + 3/2 + 0 = 5/2$

$pred(3,2) = 7/3 + 1/3 \cdot 5/2 = 3,16666667$

Πρόβλεψη της αξιολόγησης του χρήστη 4 για το αντικείμενο 4.

$\bar{u} = \frac{3}{2}$

$card(R_4) = 2$

$\sum dev_{4,i} = dev_{4,1} + dev_{4,5} = -1 - 7/3 = -3,333333$

$pred(4,4) = 3/2 + 1/2 \cdot (-3,333333) = -0,1666665$

Πρόβλεψη με βάρη της αξιολόγησης του χρήστη 3 για το αντικείμενο 2.

$$pred(3,2) = \frac{(dev_{2,3} + u_3) \cdot c_{2,3} + (dev_{2,4} + u_4) \cdot c_{2,4} + (dev_{2,5} + u_5) \cdot c_{2,5}}{c_{2,3} + c_{2,5} + c_{2,4}} =$$

$$= \frac{(1+1) \cdot 1 + (3/2+1) \cdot 2 + (0+5) \cdot 2}{1+2+2} = 17/5 = 3,4$$

Πρόβλεψη με βάρη της αξιολόγησης του χρήστη 4 για το αντικείμενο 4.

$$pred(4,4) = \frac{(dev_{4,1} + u_1) \cdot c_{4,1} + (dev_{4,5} + u_5) \cdot c_{4,5}}{c_{4,1} + c_{4,5}} =$$
$$= \frac{(-1+2) \cdot 2 + (-7/3+1) \cdot 3}{2+3} = -2/5 = -0,4$$

Παρατηρείται ότι ο αλγόριθμος κάποιες φορές παράγει αρνητικό αριθμό σαν αποτέλεσμα. Ο αριθμός αυτός δεν έχει φυσική σημασία, διότι οι αποδεκτές τιμές των προβλέψεων είναι το διάστημα $\{0\} \cup [1,5]$. Το μηδέν συμπεριλαμβάνεται στις αποδεκτές τιμές των προβλέψεων για τις περιπτώσεις εκείνες που δεν είναι δυνατή η παραγωγή των προβλέψεων. Επομένως πρέπει να ληφθούν μέτρα για τις περιπτώσεις όπου παράγεται αρνητική τιμή ή τιμή που ανήκει στο διάστημα (0,1).

Παράρτημα Ε΄

Πηγαίος κώδικας και σύνολα δεδομένων.

Στο σημείο αυτό επισυνάπτεται δίσκος δεδομένων, ο οποίος περιέχει τα αρχεία πηγαίου κώδικα και τα σύνολα δεδομένων που χρησιμοποιήθηκαν στις εφαρμογές των πειραμάτων.

Βιβλιογραφία

[1] P. S Vigklas A. G. Akritas, G. I. Malaschonok. The svd-fundamental theorem of linear algebra. *Nonlinear Analysis: Modelling and Control*, Vol.11 No.2:123–136, 2006.

[2] Diego Andrade, Basilio B Fraguela, James Brodman, and David Padua. Task-parallel versus data-parallel library-based programming in multicore systems. *Library*, pages 101–110, 2009.

[3] Paterek Arkadiusz. Improving regularized singular value decomposition for collaborative filtering. *In Proceedings of the KDD Cup Workshop at the 13th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD '07)*, pages 39–42, 2007.

[4] R V Babu, K Srinivas, and S A Devi. A new approach for cluster based collaborative filters. *International Journal of Engineering Science*, vol.2(11):6585–6592, 2010.

[5] Blacklight. `http://www.psc.edu/machines/sgi/uv/blacklight.php`.

[6] Lawrence Livermore National Laboratory Blaise Barney. Introduction to parallel computing. `https://computing.llnl.gov/tutorials/parallel_comp/`.

[7] Lawrence Livermore National Laboratory Blaise Barney. Message passing interface (mpi). `https://computing.llnl.gov/tutorials/mpi/`.

[8] Lawrence Livermore National Laboratory Blaise Barney. Openmp. `https://computing.llnl.gov/tutorials/openMP/`.

[9] Jose Luis Bosque, Oscar D. Robles, Pablo Toharia, and Luis Pastor. H-isoefficiency: Scalability metric for heterogeneous systems. *Proceedings of the 10th International Conference on Computational and Mathematical Methods in Science and Engineering, CMMSE 2010*, June 2010.

[10] J S Breese, D Heckerman, and C Kadie. Empirical analysis of predictive algorithms for collaborative filtering. *Proceedings of the 14th conference on Uncertainty in Artificial Intelligence*, vol.461:43–52, 1998.

[11] Barbara Chapman, Gabriele Jost, and Ruud van der Pas. *Using OpenMP: Portable Shared Memory Parallel Programming (Scientific and Engineering Computation).* The MIT Press, 2007.

[12] Principal component analysis. `http://en.wikipedia.org/wiki/Principal_component_analysis`.

[13] Walker Matt Daruru Srivatsava, Marín Nena and Ghosh Joydeep. Pervasive parallelism in data mining : Dataflow solution to co-clustering large and sparse netflix data. *KDD '09 Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1115–1123, 2009.

[14] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. *In OSDI'04: Proceedings of the 6th conference on Symposium on Opearting Systems Design and Implementation*, 2004.

[15] Konstantinos G. Margaritis Emmanouil Vozalis. Analysis of recommender systems' algorithms. *The 6th Hellenic European Conference on Computer Mathematics & its Applications (HERCMA), Athens, Greece*, pages 732–745, 2003.

[16] Konstantinos G. Margaritis Emmanouil Vozalis and Angelos Markos. Evaluation of standart svd-based techniques for colaborative filtering. *The 9th Hellenic European Conference on Computer Mathematics & its Applications (HERCMA), Athens, Greece*, September 2009.

[17] Diego Fernández Fidel Cacheda, Victor Carneiro and Vreixo Formoso. Comparison of collaborative filtering algorithms:limitations of current techniques and proposals for scalable, high performance recommender systems. *ACM Transactions on the Web*, vol. 05,(No. 1), February 2011.

[18] GraphLab: A New Parallel Framework for Machine Learning. `http://graphlab.org/`.

[19] Rainer Gemulla, Erik Nijkamp, Peter J. Haas, and Yannis Sismanis. Large-scale matrix factorization with distributed stochastic gradient descent. *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining KDD '11*, pages 69–77, 2011.

[20] T George and S Merugu. A scalable collaborative filtering framework based on co-clustering. *Fifth IEEE International Conference on Data Mining ICDM05*, pages 625–628, 2005.

[21] Michael Gerndt. Parallel programming models , tools and performance analysis. *Quantum*, 10:27–45, 2000.

[22] Ananth Grama, Anshul Gupta, and Vipin Kumar. Isoefficiency function: A scalability metric for parallel algorithms and architectures. 1993.

[23] Apache Hadoop. `http://hadoop.apache.org/`.

[24] Terveen G. Herlocker J., Konstan J. and Riedl J. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems.*, Vol. 22(1):5–53, January 2004.

[25] Guangquan Zhang Jing Jiang, Jie Lu and Guodong Long. Scaling-up item-based collaborative filtering recommendation algorithm based on hadoop. *World Congress Services (SERVICES), 2011 IEEE*, pages 490 –497, july 2011.

[26] Yehuda Koren. Factorization meets the neighborhood:a multifaceted collaborative filtering model. *Baseline*, vol. 08:426–434, 2008.

[27] Daniel Lemire and Anna Maclachlan. Slope one predictors for online rating-based collaborative filtering. *In SIAM Data Mining (SDM '05), Newport Beach, California*, pages 471–476, April 2005.

[28] Yucheng Low, Joseph Gonzalez, Aapo Kyrola, Danny Bickson, Carlos Guestrin, and Joseph M. Hellerstein. Graphlab: A new parallel framework for machine learning. *Conference on Uncertainty in Artificial Intelligence (UAI)*, July 2010.

[29] Chih-chao Ma. A guide to singular value decomposition for collaborative filtering. *Computer*, pages 1–14, 2008.

[30] Apache Mahout. `http://mahout.apache.org/`.

[31] Konstantinos G. Margaritis. `http://pdplab.it.uom.gr/teaching/llnl-gr/`.

[32] Zhongfu Wu Min Gao and Feng Jiang. Userrank for item-based collaborative filtering recommendation. *Information Processing Letters*, vol. 111,(No. 9):440–446, April 2011.

[33] Shirley Moore, David Cronk, Kevin London, and Jack Dongarra. Review of performance analysis tools for mpi parallel programs. *Statistics*, pages 241–248, 2001.

[34] Christopher C. Johnson Muqeet Ali and Alex K. Tang. Parallel collaborative filtering for streaming data. *http://www.cs.utexas.edu/ cjohnson/*, December 2011.

[35] Richi Nayak Namita Mittal, MC Govil and KC Jain. Recommender system framework using clustering and collaborative filtering. *Third International Conference on Emerging Trends in Engineering and Technology ICETET*, pages 555–558, 2010.

[36] A. Narang, R. Gupta, A. Joshi, and V.K. Garg. Highly scalable parallel collaborative filtering algorithm. *High Performance Computing (HiPC), 2010 International Conference on*, pages 1 –10, dec. 2010.

[37] Srivastava Abhinav Narang Ankur and Naga Praveen Kumar Katta. Distributed scalable collaborative filtering algorithm. *Euro-Par'11 Proceedings of the 17th international conference on Parallel processing*, 2011.

[38] Hong Wu Ye Pu Wang. A personalized recommendation algorithm combining slope one scheme and user based collaborative filtering. *International Conference on Industrial and Information Systems IIS '09*, pages 152–154, June 2009.

[39] GroupLens Research. Book-crossing data set. `http://www.grouplens.org/node/74`.

[40] GroupLens Research. Movielens data sets. `http://www.grouplens.org/node/73`.

[41] Shapira B. Ricci F., Rokach L. and Kantor P.B. *Recommender Systems Handbook*. Springer US, 2011.

[42] John Riedl. Research challenges in recommender systems. *Tutorial sessions Recommender Systems Conference ACM RecSys*, October 2009.

[43] Li Sa. Collaborative filtering recommendation algorithm based on cloud model clustering of multi-indicators item evaluation. *Business Computing and Global Informatization (BCGIN), 2011 International Conference*, pages 645 – 648, August 2011.

[44] Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. *Advances in Neural Information Processing Systems*, Vol. 20, 2008.

[45] Konstan J. Sarwar B., Karypis G. and Riedl J. Analysis of recommendation algorithms for e-commerce. *ACM E-Commerce 2000 Conference*, October 2000.

[46] Konstan J. Sarwar B., Karypis G. and Riedl J. Application of dimensionality reduction in recommender system - a case study. *In ACM WebKDD Workshop*, pages 285–295, 2000.

[47] Konstan J. Sarwar B., Karypis G. and Riedl J. Item-based collaborative filtering recommendation algorithms. *10th International Conference on World Wide Web*, pages 285–295, May 2001.

[48] Herlocker J. Schafer J., Frankowski D. and Sen S. *The Adaptive Web: Methods and Strategies of Web Personalization*. Springer-Verlag Berlin Heidelberg, 2007.

[49] Xiaoyuan Su and Taghi M. Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in Artificial Intelligence*, Vol. 2009, January 2009.

[50] Introduction to Parallel Programming and MapReduce. `code.google.com/edu/parallel/mapreduce-tutorial.html`.

[51] Singular Value Decomposition (SVD) tutorial. `http://web.mit.edu/be.400/www/SVD/Singular_Value_Decomposition.htm`.

[52] Slobodan Vucetic and Zoran Obradovic. Collaborative filtering using a regression-based approach. *Knowledge and Information Systems*, 7(1):1–22, 2004.

[53] J Wang, Ap De Vries, and Mjt Reinders. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval SIGIR 06*, pages 501–508, 2006.

[54] Wikipedia. Data parallelism. `http://en.wikipedia.org/wiki/Data_parallelism`.

[55] Wikipedia. Message passing interface. `http://en.wikipedia.org/wiki/Message_Passing_Interface`.

[56] Wikipedia. Slope one. `http://en.wikipedia.org/wiki/Slope_One`.

[57] Wikipedia. Task parallelism. `http://en.wikipedia.org/wiki/Task_parallelism`.

[58] Yao Wu, Qiang Yan, Danny Bickson, Yucheng Low, and Qing Yang. Efficient multicore collaborative filtering. *Matrix*, 2011.

[59] Liang Xiong, Xi Chen, Tzu-Kuo Huang, Jeff Schneider, and Jaime G. Carbonell. Temporal collaborative filtering with bayesian probabilistic tensor factorization. *In Proceedings of SIAM Data Mining*, 2010.

[60] Gui-Rong Xue, Chenxi Lin, Qiang Yang, Wensi Xi, Hua-Jun Zeng, Yong Yu, and Zheng Chen. Scalable collaborative filtering using cluster-based smoothing. *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval SIGIR 05*, pages 114–121, August 2005.

[61] Chris Volinsky Yehuda Koren, Robert Bell. Matrix factorization techniques for recommender systems. *IEEE Computer*, Vol. 42, No. 8:30–37, August 2009.

[62] DeJia Zhang. An item-based collaborative filtering recommendation algorithm using slope one scheme smoothing. *Second International Symposium on Electronic Commerce and Security ISECS '09*, May 2009.

[63] Zhi-Dan Zhao and Ming-Sheng Shang. User-based collaborative-filtering recommendation algorithms on hadoop. *2010 Third International Conference on Knowledge Discovery and Data Mining*, pages 478 –481, jan. 2010.

[64] Yunhong Zhou, Dennis Wilkinson, Robert Schreiber, and Rong Pan. Large-scale parallel collaborative filtering for the netflix prize. *Algorithmic Aspects in Information and Management*, Vol.5034:337–348, 2008.

[65] Nianlong Luo Zilei Sun and Wei Kuang. One real-time personalized recommendation systems based on slope one algorithm. *2011 Eighth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, pages 1826–1830, July 2011.