ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΧΗΜΙΚΩΝ ΜΗΧΑΝΙΚΩΝ

ΤΟΜΕΑΣ ΑΝΑΛΥΣΗΣ, ΣΧΕΔΙΑΣΜΟΥ & ΑΝΑΠΤΥΞΗΣ
ΔΙΕΡΓΑΣΙΩΝ & ΣΥΣΤΗΜΑΤΩΝ

ΑΠΟΔΟΤΙΚΗ ΕΠΙΛΥΣΗ ΜΕΓΑΛΩΝ, ΑΡΑΙΩΝ ΣΥΣΤΗΜΑΤΩΝ ΣΕ ΠΟΛΥΠΥΡΗΝΑ ΣΥΣΤΗΜΑΤΑ ΚΟΙΝΗΣ ΔΙΑΧΕΙΡΙΣΗΣ ΜΝΗΜΗΣ, ΜΕ ΧΡΗΣΗ ΤΟΥ PARDISO

Διπλωματική Εργασία Ανδρέα Γ. Στεφαδούρου

Επιβλέπων Καθηγητής : Ανδρέας Γ. Μπουντουβής

Αθήνα
Φεβρουάριος 2012

Περίληψη

Στην παρούσα διπλωματική εργασία γίνεται μια σύντομη παρουσίαση του μαθηματικού πακέτου PARDISO και της αποδοτικής χρήσης αυτού για την επίλυση μεγάλων αραιών συστημάτων σε πολυπύρηνα συστήματα με αρχιτεκτονική διαμοιραζόμενης μνήμης.

Στο 1^{ο} κεφάλαιο παρουσιάζεται επιγραμματικά ο τρόπος λειτουργίας ενός υπολογιστικού συστήματος με αρχιτεκτονική διαμοιραζόμενης μνήμης, ενώ αναλύονται διεξοδικά οι διαθέσιμες τεχνολογίες που υιοθετούνται για την ελαχιστοποίηση των φαινομένων συμφόρησης στον κεντρικό δίαυλο επικοινωνίας και του ανταγωνισμού στην κεντρική μνήμη από τους επεξεργαστές.

Στο 2^{ο} κεφάλαιο παρουσιάζονται οι κατηγορίες των προβλημάτων που μπορούν να λυθούν με την βοήθεια του PARDISO, ενώ αναλύονται οι εσωτερικές λειτουργίες του επιλύτη για κάθε περίπτωση. Τέλος, παρουσιάζεται η μορφή με την οποία πρέπει να εισάγονται τα στοιχεία του προβλήματος σε τέτοιου είδους επιλύτες.

Στο 3^{ο} κεφάλαιο αναλύεται η διαδικασία σύνδεσης του PARDISO με το οπτικό περιβάλλον προγραμματισμού Microsoft Visual Studio και παρουσιάζονται αναλυτικά οι παράμετροι λειτουργίας του πακέτου και ο κώδικας που χρησιμοποιήθηκε για την δοκιμή της απόδοσής του.

Στο 4^{ο} κεφάλαιο αναλύονται τα τεχνικά χαρακτηριστικά των υπολογιστών που χρησιμοποιήθηκαν για τις δοκιμές της απόδοσης του επιλύτη και παρατίθενται αναλυτικά τα αποτελέσματα των δοκιμών του PARDISO ως προς τον χρόνο επίλυσης δώδεκα προβλημάτων αυξανόμενου μεγέθους με ένα, δύο και τέσσερις πυρήνες σε λειτουργία. Τα αποτελέσματα αναλύονται και σε συνδυασμό με αποτελέσματα δοκιμών του επιλύτη από την βιβλιογραφία προκύπτουν συμπεράσματα για την συμπεριφορά του PARDISO σε διαφορετικούς επεξεργαστές, για την κλιμάκωση των επιδόσεων των συστημάτων δοκιμών όσο οι λειτουργικοί πυρήνες αυξάνονται και για την αποδοτική οικονομικά χρήση των υπαρχόντων πολυπύρηνων αρχιτεκτονικών για την επίλυση κάθε μεγέθους προβλήματος.

Ευχαριστίες

Θα ήθελα να εκφράσω τις ευχαριστίες μου στον Καθηγητή Ανδρέα Μπουντουβή, για την υπομονή και την καθοδήγησή του κατά τη διάρκεια της εκπόνησης της διπλωματικής μου εργασίας.

Επίσης, ευχαριστώ θερμά τον Νίκο Χειμαριό για τις πολύτιμες συμβουλές του και τις ατελείωτες συζητήσεις πάνω στο προγραμματιστικό κομμάτι της εργασίας.

Ευχαριστίες θα ήθελα επίσης να απευθύνω στον Αναπληρωτή Καθηγητή Χ. Κυρανούδη, καθώς και στον Αναπληρωτή Καθηγητή Χ. Σαρίμβεη, οι οποίοι αξιολόγησαν την εργασία μου ως μέλη της τριμελούς επιτροπής εξέτασης.

Τέλος, θα ήθελα να ευχαριστήσω θερμά για την υποστήριξη που πάντοτε μου προσέφερε την οικογένειά μου και ιδιαίτερα την μητέρα μου. Η συμπεριφορά της και η στάση της απέναντι σε τρομακτικές δυσκολίες και εμπόδια αποτέλεσε αστείρευτη πηγή έμπνευσης και καθοδήγησης για να ολοκληρώσω τον κύκλο των σπουδών μου. Για τον λόγο αυτό, θα ήθελα να της αφιερώσω το αποτέλεσμα των προσπαθειών μου.

Περιεχόμενα

Κεφάλαιο 3^{ο} : Εγκατάσταση και χρήση του PARDISO

Κεφάλαιο 4^{ο} : Αξιολόγηση του PARDISO

Κεφάλαιο 1ο

1.1 Εισαγωγή

Παρά το γεγονός ότι η παρεχόμενη υπολογιστική ισχύς από τους σειριακούς επεξεργαστές έχει αυξηθεί δραματικά τα τελευταία χρόνια, εξακολουθεί να υπάρχει μία ευρύτατη περιοχή επιστημονικών προβλημάτων όπου η απαιτούμενη ισχύς αποτελεί πολλαπλάσιο της παρεχόμενης σε επίπεδο αρκετά μεγάλων τάξεων μεγέθους. Τέτοια επιστημονικά πεδία αποτελούν η προσομοίωση νευρωνικών δικτύων στον τομέα της βιολογίας, η πρόγνωση καιρού, η προχωρημένη φυσική, η δημιουργία και επεξεργασία γραφικών υψηλής ανάλυσης, η προσομοίωση χημικών αντιδράσεων και η αεροδυναμική. Η επίλυση τέτοιων προβλημάτων απαιτεί σε πολλές περιπτώσεις υπολογιστές με ισχύ από μερικές δεκάδες έως μερικές χιλιάδες φορές μεγαλύτερη από αυτή των ταχύτερων σειριακών επεξεργαστών που κυκλοφορούν στην αγορά.

Καθώς τα υλικά κατασκευής επεξεργαστών (πυρίτιο), αλλά και οι τεχνολογίες ολοκλήρωσης κυκλωμάτων τείνουν να φτάσουν στα όριά τους, η σχεδίαση και κατασκευή ισχυρότερων σειριακών επεξεργαστών γίνεται συνεχώς δυσκολότερη και ιδιαίτερα δαπανηρή. Για τον λόγο αυτόν, οι κατασκευάστριες εταιρίες κατέληξαν στο συμπέρασμα ότι είναι πολύ πιο οικονομική η κατασκευή υπολογιστών με πολλούς επεξεργαστές παρά η εκ νέου σχεδίαση ενός σειριακού επεξεργαστή που να καλύπτει τις ανάγκες κάθε χρήστη. Έτσι, σε περιπτώσεις όπου απαιτείται υπολογιστική ισχύς πολλαπλάσια της παρεχόμενης από οποιονδήποτε σειριακό επεξεργαστή, ο χρήστης μπορεί να συνδέσει τον απαραίτητο αριθμό επεξεργαστών ώστε να πετύχει το επιθυμητό αποτέλεσμα.

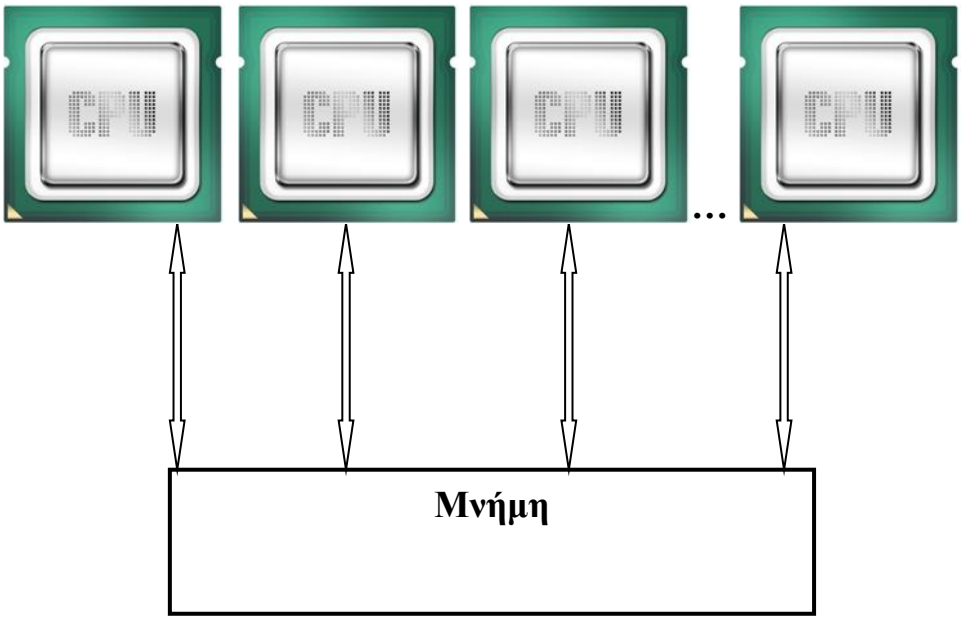
Φυσικά, με τη χρήση πολλαπλών επεξεργαστών σε ένα υπολογιστικό σύστημα, προκύπτουν κάποια ζητήματα αναφορικά με την αρχιτεκτονική του συστήματος. Συγκεκριμένα, για να υπάρξει δυνατότητα ταυτόχρονης λειτουργίας των επεξεργαστών (ώστε το σύστημα να μπορεί να χαρακτηρισθεί ως παράλληλο), θα πρέπει να υπάρχει δυνατότητα διαμοιρασμού δεδομένων και επικοινωνίας μεταξύ τους σε πραγματικό χρόνο. Οι βασικές αρχιτεκτονικές που χρησιμοποιούνται κατά κόρον στις μέρες μας είναι :

- Αρχιτεκτονική διαμοιραζόμενης μνήμης. Στην περίπτωση αυτή, όλοι οι επεξεργαστές έχουν πρόσβαση στην ίδια μνήμη από όπου αντλούν δεδομένα.

- Αρχιτεκτονική κατανεμημένης μνήμης (επικοινωνία με πέρασμα μηνυμάτων). Στην περίπτωση αυτή, ο κάθε επεξεργαστής έχει αποκλειστική μνήμη στην οποία δεν έχουν πρόσβαση οι υπόλοιποι επεξεργαστές του συστήματος. Η επικοινωνία και η ανταλλαγή δεδομένων μεταξύ των επεξεργαστών επιτυγχάνεται μέσα από ένα δίκτυο επικοινωνίας και περάσματος μηνυμάτων.

1.2 Εισαγωγή στα πολυπύρηνα συστήματα διαμοιραζόμενης μνήμης (shared memory multiprocessors)

Στην ενότητα αυτή θα εξετάσουμε με λεπτομέρεια το πώς λειτουργεί ένα σύστημα πολλαπλών επεξεργαστών με διαμοιραζόμενη μνήμη (Σχήμα 1.1) και τους τρόπους που χρησιμοποιούνται στην σύγχρονη βιομηχανία για την βελτιστοποίηση της απόδοσης αυτών.



Σχήμα 1.1
Ένα πολυπύρηνο υπολογιστικό σύστημα με διαμοιραζόμενη μνήμη

Σε ένα πολυπύρηνο υπολογιστικό σύστημα με διαμοιραζόμενη μνήμη, όλοι οι επεξεργαστές εκτελούν υπολογισμούς παράλληλα και ο κάθε ένας ξεχωριστά έχει την δυνατότητα να προσπελάσει την μνήμη του συστήματος μέσα από μία αρχιτεκτονική κοινού διαύλου. Ο δίαυλος που συνδέει τους επεξεργαστές με την μνήμη και τα περιφερειακά του συστήματος, καλείται να εκτελέσει χρέη διαιτησίας και να διαχειριστεί την πιθανή ταυτόχρονη επιθυμία προσπέλασης της μνήμης από τους διάφορους επεξεργαστές καθώς αυτοί εκτελούν υπολογισμούς. Η παραπάνω

διαδικασία εξασφαλίζει την ισορροπημένη εξυπηρέτηση του πλήθους των επεξεργαστών του συστήματος με την ελάχιστη δυνατή υστέρηση.

Παρότι ένα πολυπύρηνο υπολογιστικό σύστημα με διαμοιραζόμενη μνήμη διαθέτει πολλαπλούς επεξεργαστές που πραγματοποιούν υπολογισμούς παράλληλα, ο τρόπος λειτουργίας του κάθε επεξεργαστή είναι ταυτόσημος με αυτόν που θα είχε αν τοποθετούνταν από μόνος του για να εξυπηρετήσει ένα σειριακό (μονοπύρηνο) υπολογιστικό σύστημα. Ο κάθε επεξεργαστής διαβάζει τιμές μεταβλητών από την κοινή μνήμη, πραγματοποιεί τους υπολογισμούς του ανεξάρτητα από τους υπόλοιπους και με το πέρας των εργασιών του επιστρέφει νέες τιμές στην μνήμη. Κατά αυτόν τον τρόπο και σε θεωρητικό επίπεδο, αν ένα πολυπύρηνο υπολογιστικό σύστημα με διαμοιραζόμενη μνήμη έχει n επεξεργαστές με υπολογιστική ισχύ $x\ GigaFLOPS/CPU$ (FLoating point OPerations per Second) [1], ο συνδυασμός τους θα δώσει ισχύ $(n \cdot x)GigaFLOPS$. Φυσικά, ο παραπάνω υπολογισμός ισχύος είναι το θεωρητικό μέγιστο της ισχύος που μπορεί να έχει ένα πολυπύρηνο σύστημα και είναι σχεδόν αδύνατο να επιτευχθεί όσο ο αριθμός των επεξεργαστών αυξάνεται. Αυτό οφείλεται στο γεγονός ότι όταν ένας μεγάλος αριθμός επεξεργαστών προσπαθεί να προσπελάσει ταυτόχρονα την μνήμη του συστήματος, η μνήμη δεν μπορεί να ανταπεξέλθει και αρκετοί από τους επεξεργαστές αναγκάζονται να περιμένουν σε ανενεργή κατάσταση όσο άλλοι εξυπηρετούνται. Αυτό έχει ως αποτέλεσμα την πτώση της απόδοσης του συστήματος, οπότε κατά τον σχεδιασμό πολυπύρηνων συστημάτων δίνεται ιδιαίτερη σημασία στο να αποφευχθούν τέτοια ζητήματα και να μειωθεί κατά το μέγιστο δυνατό η πιθανότητα ανταγωνισμού πρόσβασης στη μνήμη.

Για να περιοριστούν τα φαινόμενα ανταγωνισμού πρόσβασης στη μνήμη και να γίνει ένα πολυπύρηνο σύστημα πιο αποδοτικό χρησιμοποιούνται διάφορες τεχνικές βελτίωσης της αρχιτεκτονικής του, οι οποίες θα αναλυθούν διεξοδικά στην συνέχεια του κεφαλαίου. Αναφορικά, θα αναλυθούν οι παρακάτω τεχνολογίες :

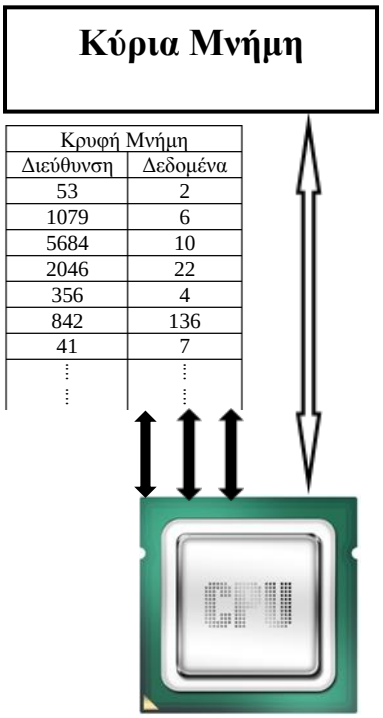
- Χρήση μιας τοπικής κρυφής μνήμης από κάθε επεξεργαστή, η οποία χρησιμοποιείται για να αποθηκεύονται δεδομένα από τα πιο συχνά προσπελάσιμα τμήματα της κεντρικής μνήμης του υπολογιστή. Κατά αυτόν τον τρόπο, τα πιο συχνά χρησιμοποιούμενα δεδομένα είναι συνεχώς διαθέσιμα σε κάθε επεξεργαστή και μπορούν να προσπελαστούν ταχύτατα και χωρίς πιθανότητα ανταγωνισμού.
- Εξέλιξη των δικτύων σύνδεσης των επεξεργαστών με την μνήμη για την εξάλειψη του φαινομένου του ανταγωνισμού στην μνήμη.

- Ανάπτυξη της τεχνολογίας κατασκευής αρθρωμάτων μνήμης ώστε να επιτυγχάνεται μεγαλύτερη ταχύτητα επικοινωνίας των επεξεργαστών με την μνήμη και χαμηλότεροι λανθάνοντες χρόνοι.
- Χρήση πολλαπλών καναλιών μνήμης σε απλούς διαύλους, για την βελτιστοποίηση της απόδοσης πολυπύρηνων συστημάτων με σχετικά μικρό αριθμό επεξεργαστών.

1.3 Κρυφή μνήμη (Cache)

1.3.1 Κρυφή μνήμη σε σειριακό υπολογιστικό σύστημα

Σε ένα σειριακό υπολογιστικό σύστημα (με έναν επεξεργαστή), η κυριότερη τεχνολογία που χρησιμοποιείται για την μείωση του ρυθμού πρόσβασης του επεξεργαστή στην μνήμη, είναι η χρήση της κρυφής μνήμης (cache memory). Η cache του επεξεργαστή είναι ένα μικρό τμήμα μνήμης ιδιαίτερα υψηλής ταχύτητας, ενσωματωμένη πλέον στο ολοκληρωμένο κύκλωμα του επεξεργαστή, που χρησιμοποιείται από την κεντρική μονάδα επεξεργασίας του υπολογιστή για να μειωθεί ο μέσος χρόνος που ξοδεύεται για την πρόσβαση στη μνήμη. Σε αυτήν αποθηκεύονται οι τιμές δεδομένων από τις βασικές θέσεις μνήμης που χρησιμοποιούνται πιο συχνά από τον επεξεργαστή. Κατά αυτόν τον τρόπο μπορούν να προσπελαστούν πολύ πιο γρήγορα από ότι οι τιμές δεδομένων που αποθηκεύονται στην μεγαλύτερη αλλά πιο αργή κύρια μνήμη. Το μέγεθος της κρυφής μνήμης για έναν σύγχρονο επεξεργαστή ανέρχεται από μερικές εκατοντάδες kilobytes μέχρι κάποια megabytes, σε αντίθεση με την κύρια μνήμη όπου ακόμα και στα πιο απλά συστήματα υπολογιστών ανέρχεται σε μερικά gigabytes. Από τη στιγμή όμως που η κρυφή μνήμη είναι πολύ μικρότερη και αποτελεί μέρος του ολοκληρωμένου κυκλώματος του επεξεργαστή, είναι δυνατόν να χρησιμοποιηθεί για την κατασκευή της ακριβή αλλά πολύ πιο αποδοτική τεχνολογία υλικού από ότι για κατασκευή της κύριας μνήμης.



Σχήμα 1.2
Cache σε έναν επεξεργαστή

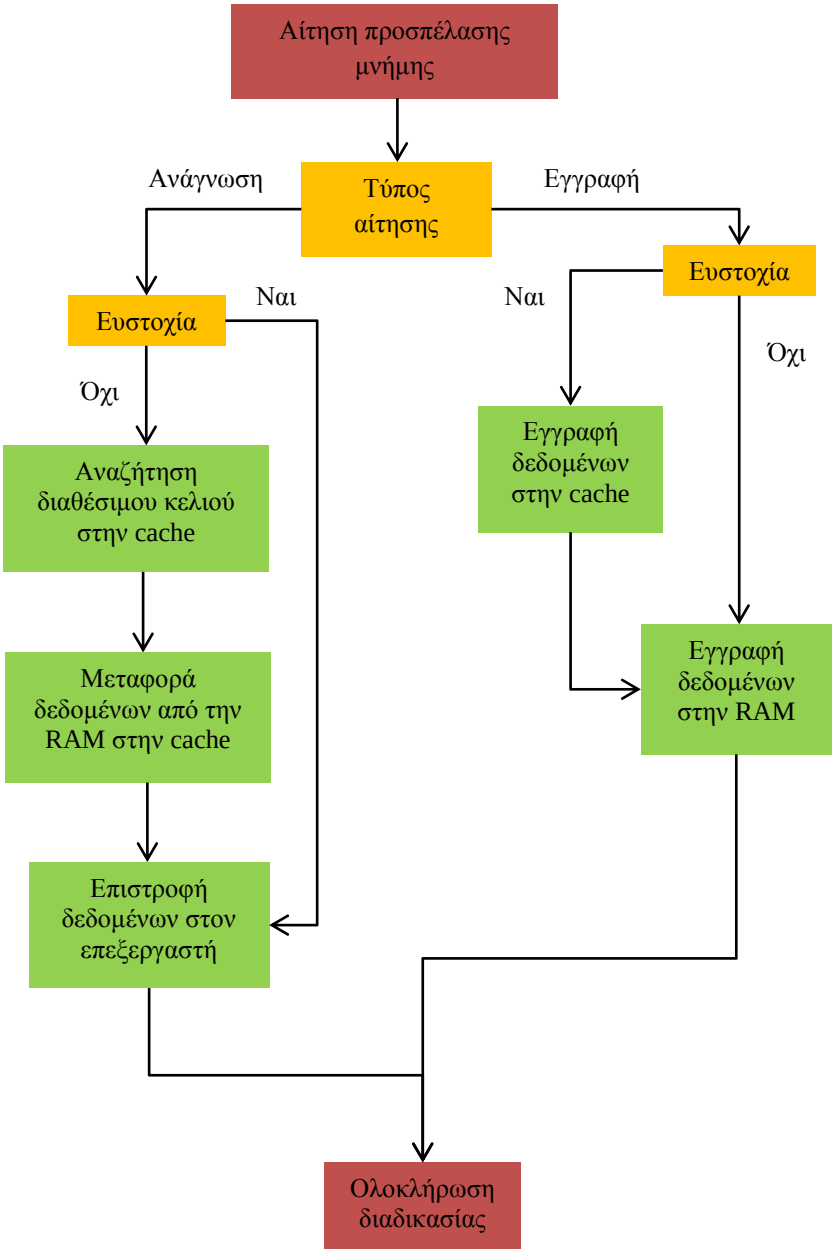
Η βασική αρχή της συνεργασίας ενός επεξεργαστή με την κρυφή του μνήμη παρουσιάζεται στο Σχήμα 1.2. Οι καταχωρήσεις της κρυφής μνήμης αποτελούνται από ζεύγη τιμών δεδομένων και διεύθυνσης αυτών στην κύρια μνήμη του συστήματος. Η τιμή της διεύθυνσης αντιστοιχεί στην πραγματική διεύθυνση ενός κελιού της κύριας μνήμης, και τα δεδομένα είναι τα περιεχόμενα του αντίστοιχου κελιού. Η συσχετιστική αυτή δομή της κρυφής μνήμης, δίνει την δυνατότητα όλες οι καταχωρήσεις της να μπορούν να αναζητηθούν με μία μόνο ενέργεια και με πολύ μεγάλη ταχύτητα. Όταν ο επεξεργαστής χρειάζεται να προσπελάσει μια θέση στην κύρια μνήμη, ελέγχει πρώτα αν ένα αντίγραφο των εν λόγω δεδομένων βρίσκεται στην cache. Αν η καταχώρηση μνήμης που αναζητείται είναι αποθηκευμένη στην cache, ο επεξεργαστής διαβάζει αμέσως από ή γράφει στη κρυφή μνήμη, η οποία είναι πολύ ταχύτερη σε σχέση με την κύρια μνήμη. Στην περίπτωση αυτήν προκύπτει μια «ευστοχία της κρυφής μνήμης» (cache hit). Αν η αναφορά μνήμης δεν βρίσκεται στην κρυφή μνήμη, θα πρέπει να προσπελαστεί η κύρια μνήμη οπότε προκύπτει αστοχία της κρυφής μνήμης. Το ιδιαίτερα μικρό μέγεθος της κρυφής μνήμης συγκριτικά με την κύρια οδηγεί στο φαινόμενο οι καταχωρήσεις της να αποτελούν ουσιαστικά ένα πολύ μικρό ποσοστό του συνόλου των καταχωρήσεων της κύριας μνήμης. Αυτό οδηγεί αναπόφευκτα σε μη μηδενικό ρυθμό αστοχιών (cache miss rate).

Για να μειωθεί στο ελάχιστο δυνατό ο ρυθμός αστοχιών, οι κρυφές μνήμες εκμεταλλεύονται το φαινόμενο της τοπικότητας [2] στα προγράμματα υπολογιστών και αποθηκεύουν τις πιο πρόσφατα χρησιμοποιημένες τιμές μνήμης, αφού αυτές είναι πιο πιθανόν ότι θα χρησιμοποιηθούν και στο άμεσο μέλλον. Ένα τυπικό συστήματα απλού επεξεργαστή μπορεί να επιτύχει ευστοχία μνήμης σε ποσοστό μέχρι και 90%. Δηλαδή, μόνο το 10% από τις αναφορές μνήμης του επεξεργαστή χρειάζεται να οδηγηθούν στην πιο αργή κύρια μνήμη. Κατά αυτόν τον τρόπο επιτυγχάνεται η μείωση των πακέτων δεδομένων που ανταλλάσσονται μεταξύ επεξεργαστή και κύριας μνήμης μέσω του διαύλου και άρα του χρόνου που απαιτείται για την επικοινωνία του επεξεργαστή με τη μνήμη, αφού όσο περισσότερες διαδικασίες πρόσβασης πραγματοποιούνται μέσω της cache, τόσο ο μέσος λανθάνων χρόνος πρόσβασης (latency) στη μνήμη θα είναι πιο κοντά στον λανθάνων χρόνο της cache από ότι σε αυτόν της κύριας μνήμης. Στην περίπτωση που παρατηρηθεί αστοχία μνήμης, η κύρια μνήμη πρέπει να προσπελαύνεται από τον επεξεργαστή και η τιμή

των δεδομένων του συγκεκριμένου κελιού μνήμης αντιγράφεται στην κρυφή μνήμη, αντικαθιστώντας μια καταχώρησή της που δεν έχει χρησιμοποιηθεί πρόσφατα.

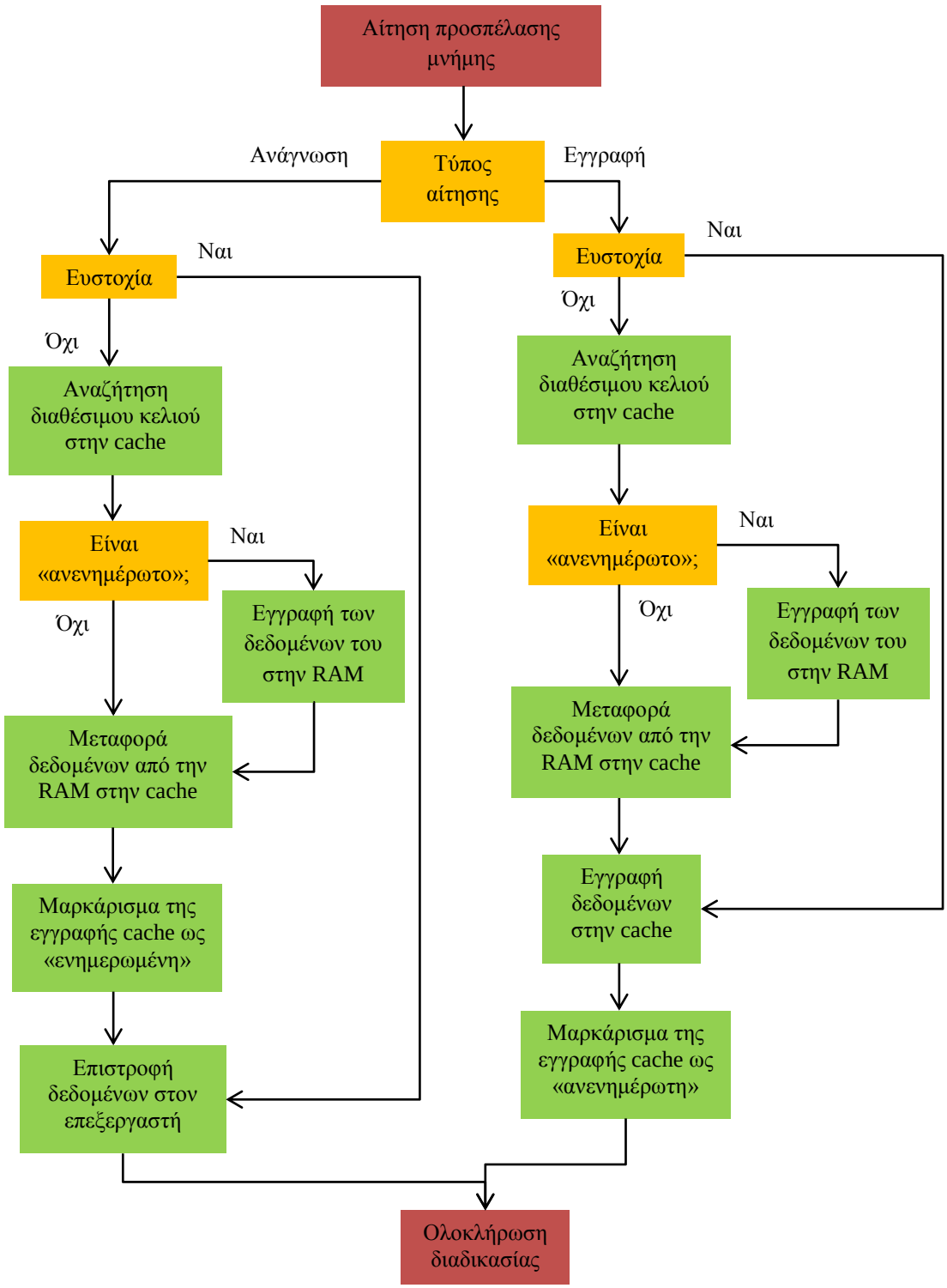
Όποτε πραγματοποιείται μια λειτουργία εγγραφής σε κάποιο τμήμα της κρυφής μνήμης, το αντίστοιχο κελί της κύρια μνήμης κρατάει την παλιά του τιμή. Για να ενημερωθεί η κύρια μνήμη για τυχόν αλλαγές στην κρυφή ακολουθούνται κατά κύριο λόγο δύο τεχνικές. Η τεχνική της απευθείας εγγραφής (write-through) και η τεχνική της ανάστροφης εγγραφής (write-back ή write-behind).

Με την προσέγγιση της απευθείας εγγραφής (Σχήμα 1.3), κάθε φορά που μια καινούργια τιμή γράφεται στην κρυφή μνήμη τότε ταυτόχρονα γράφεται και στην κύρια μνήμη του συστήματος.



Σχήμα 1.3
Απευθείας εγγραφή στην cache

Με την προσέγγιση της ανάστροφης εγγραφής (Σχήμα 1.4), η καινούργια τιμή γράφεται στην κρυφή μνήμη, το κελί της κρυφής μνήμης του οποίου η τιμή άλλαξε χαρακτηρίζεται ως «ανενημέρωτο» (dirty), ενώ το «αυθεντικό» του αντίγραφο στην κύρια μνήμη δεν ενημερώνεται. Όταν η «ανενημέρωτη» καταχώρηση στην κρυφή μνήμη δεν χρησιμοποιηθεί για αρκετή ώρα και αντικατασταθεί από κάποια άλλη θέση της κύρια μνήμης, τότε το μη ενημερωμένο αντίγραφο της κύριας μνήμης ενημερώνεται.



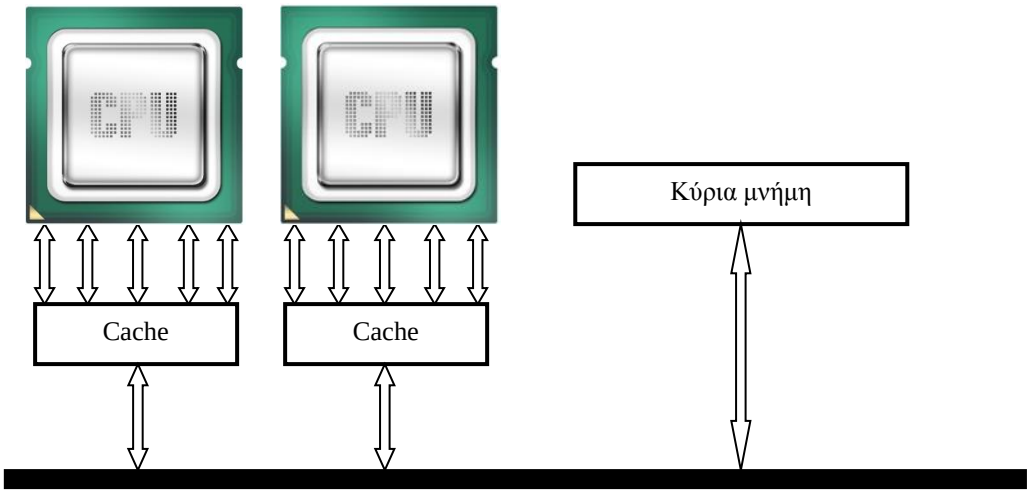
Σχήμα 1.4
Ανάστροφη εγγραφή στην cache

Με οποιαδήποτε από τις παραπάνω τεχνικές καταναλώνεται κάποιος χρόνος πρόσβασης στην μνήμη, γεγονός που καθιστά την κρυφή μνήμη πιο αποδοτική κατά τις λειτουργίες ανάγνωσης από αυτήν παρά κατά τις λειτουργίες εγγραφής σε αυτήν. Αναφορικά, στα περισσότερα προγράμματα υπολογιστών, από τις συνολικές

προσβάσεις του επεξεργαστή στην μνήμη, περίπου το 20% είναι λειτουργίες εγγραφής ενώ το 80% είναι λειτουργίες ανάγνωσης.

1.3.2 Κρυφή μνήμη σε πολυπύρηνο σύστημα διαμοιραζόμενης μνήμης

Στα πολυπύρηνα συστήματα, ο κάθε επεξεργαστής διαθέτει επίσης την δική του κρυφή μνήμη στην οποία μπορεί να αποθηκεύσει τις τιμές των πιο πρόσφατα χρησιμοποιημένων κελιών της διαμοιραζόμενης κύριας μνήμης. Αυτές οι τοπικές κρυφές μνήμες μειώνουν αρκετά των ανταγωνισμό πρόσβασης στην διαμοιραζόμενη μνήμη (Σχήμα 1.5). Όπως και στην περίπτωση ενός σειριακού συστήματος, όποτε ένας από τους επεξεργαστές επιθυμεί να προσπελάσει ένα κελί της μνήμης, πρώτα ελέγχει την προσωπική του cache για να δει αν η καταχώρηση είναι αποθηκευμένη εκεί. Αν τα απαιτούμενα δεδομένα δεν βρίσκονται στην κρυφή του μνήμη, τότε ο επεξεργαστής θα ζητήσει από τον ελεγκτή μνήμης να προσπελάσει την κύρια διαμοιραζόμενη μνήμη μέσω του διαύλου. Αν οι επεξεργαστές έχουν μέσο ποσοστό ευστοχίας 80% στην κρυφή τους μνήμη, τότε ο μέσος ρυθμός πρόσβασης στην διαμοιραζόμενη μνήμη θα μειωθεί κατά περίπου πέντε φορές, ελαχιστοποιώντας ιδιαίτερα την πιθανότητα ανταγωνισμού πρόσβασης στη μνήμη και δίνοντας την δυνατότητα σύνδεσης μεγαλύτερου αριθμού επεξεργαστών στον δίαυλο χωρίς αυτός να υπερφορτώνεται.



Σχήμα 1.5
Η επικοινωνία των επεξεργαστών με την «προσωπική» τους cache μειώνει την συμφόρηση του διαύλου

Η ύπαρξη όμως πολλαπλών κρυφών μνημών μέσα σε ένα υπολογιστικό σύστημα οδηγεί αρκετές φορές σε προβλήματα συνοχής της κρυφής μνήμης (cache coherence).

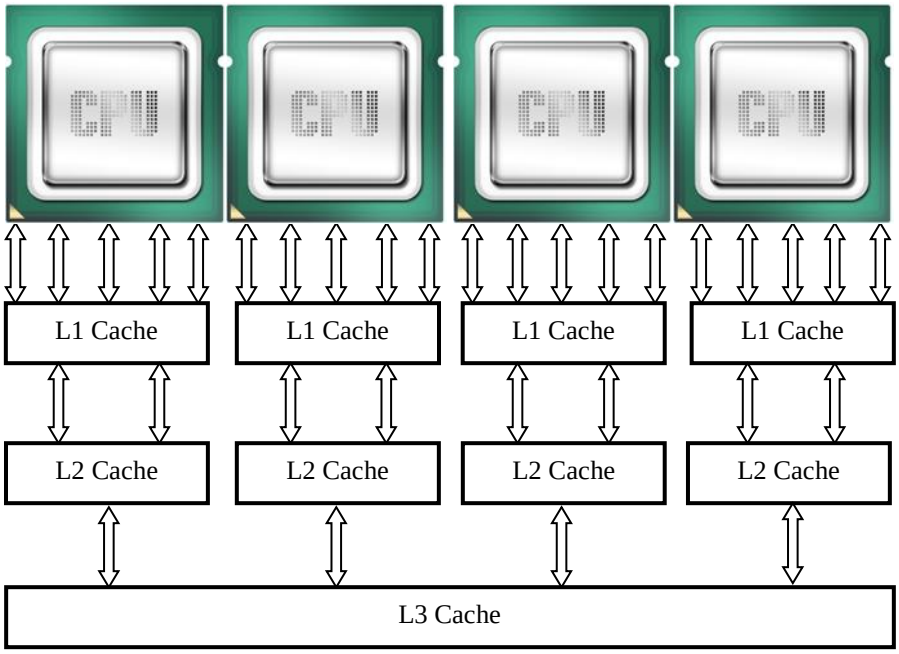
Ας υποθέσουμε ότι σε ένα σύστημα τεσσάρων πυρήνων ο πρώτος επεξεργαστής επιθυμεί να προσπελάσει τα δεδομένα μίας συγκεκριμένης διεύθυνσης στην διαμοιραζόμενη μνήμη. Αφού πραγματοποιήσει την λειτουργία ανάγνωσης, θα δημιουργηθεί ένα αντίγραφο της συγκεκριμένης διεύθυνσης στην κρυφή του μνήμη. Αν στη συνέχεια ο τρίτος επεξεργαστής διαβάσει την ίδια διεύθυνση από την διαμοιραζόμενη μνήμη, τότε και η δικιά του κρυφή μνήμη θα αποκτήσει ένα αντίγραφο. Κατά αυτόν τον τρόπο θα υπάρχουν τρία αντίγραφα από την ίδια θέση της διαμοιραζόμενης μνήμης (ένα αντίγραφο στην διαμοιραζόμενη μνήμη και δύο στις κρυφές μνήμες του πρώτου και του τρίτου πυρήνα). Όσο οι προσπελάσεις στην διεύθυνση αυτήν είναι λειτουργίες ανάγνωσης δεν παρατηρείται κανένα απολύτως πρόβλημα από την ύπαρξη πολλαπλών αντιγράφων. Κάθε επεξεργαστής θα έχει την δυνατότητα να εκτελεί λειτουργίες ανάγνωσης χρησιμοποιώντας το αντίγραφο που βρίσκεται στην κρυφή του μνήμη, χωρίς να απαιτείται καμία επαφή με την διαμοιραζόμενη μνήμη.

Αν στην παραπάνω περίπτωση ένας εκ των δύο επεξεργαστών εκτελέσει μια λειτουργία εγγραφής στην θέση μνήμης που εξετάζουμε, τότε θα προκύψει πρόβλημα συνοχής. Η τιμή του αντίγραφου που βρίσκεται στην cache του επεξεργαστή που εκτέλεσε λειτουργία εγγραφής θα έχει αλλάξει, ενώ τα άλλα δύο αντίγραφα θα έχουν την παλιά τιμή και θα πρέπει να ενημερωθούν.

Η πιο απλή λύση για το πρόβλημα συνοχής της κρυφής μνήμης είναι να απαιτείται κάθε λειτουργία εγγραφής από οποιονδήποτε επεξεργαστή να εκπέμπεται στον κοινό δίαυλο ώστε να ενημερώνεται η διαμοιραζόμενη μνήμη. Η κρυφή μνήμη κάθε επεξεργαστή θα πρέπει φυσικά να παρακολουθεί συνεχώς τον κοινό δίαυλο για να δει αν κάποιες από τις δικές της καταχωρήσεις στην μνήμη έχουν ενημερωθεί από κάποιον άλλο επεξεργαστή. Το μειονέκτημα της παραπάνω μεθόδου είναι η αύξηση του φόρτου εργασίας του διαύλου επικοινωνίας με την διαμοιραζόμενη μνήμη, αφού κάθε εγγραφή σε αυτήν από έναν επεξεργαστή θα ακολουθείται από πιθανές προσπελάσεις της διαμοιραζόμενης μνήμης και από άλλους επεξεργαστές ώστε να ενημερώσουν τα αντίγραφα στις δικές τους κρυφές μνήμες.

Για να αποφευχθεί πιθανός ανταγωνισμός στην μνήμη λόγω προβλημάτων συνοχής στην κρυφή μνήμη, χρησιμοποιούνται πολλαπλά επίπεδα cache (Σχήμα 1.6). Σε αυτήν την αρχιτεκτονική ο κάθε επεξεργαστής έχει δύο «προσωπικά» επίπεδα cache (L1 και L2) τα οποία είναι σχετικά μικρά σε μέγεθος αλλά λειτουργούν σε ιδιαίτερα υψηλή ταχύτητα, ενώ οι πυρήνες που βρίσκονται στο ίδιο chip (μέχρι οχτώ

στα πιο σύγχρονα chip ευρείας χρήσης) μοιράζονται ένα τρίτο κοινόχρηστο επίπεδο cache (L3) μεγαλύτερου μεγέθους αλλά λίγο χαμηλότερης ταχύτητας.



Σχήμα 1.6
Αρχιτεκτονική τετραπύρηνου επεξεργαστή με
τρία επίπεδα κρυφής μνήμης

Με την προσθήκη του κοινόχρηστου επιπέδου κρυφής μνήμης (L3), όποτε ένας επεξεργαστής εκτελεί μία λειτουργία εγγραφής σε μία από τις προσωπικές του cache, αυτές ενημερώνουν μόνο το αντίστοιχο αντίγραφο στην L3. Τότε, η L3 αναλαμβάνει να ενημερώσει τα αντίστοιχα αντίγραφα στους υπόλοιπους επεξεργαστές και στην διαμοιραζόμενη μνήμη, χρησιμοποιώντας μία από τις τεχνικές που αναφέρθηκαν στην ενότητα 1.2.1 (write-through ή write-back). Κατά αυτόν τον τρόπο, κάθε φορά που πραγματοποιείται λειτουργία εγγραφής από κάποιον επεξεργαστή, οι εκπομπές στον δίαυλο περιορίζονται σημαντικά.

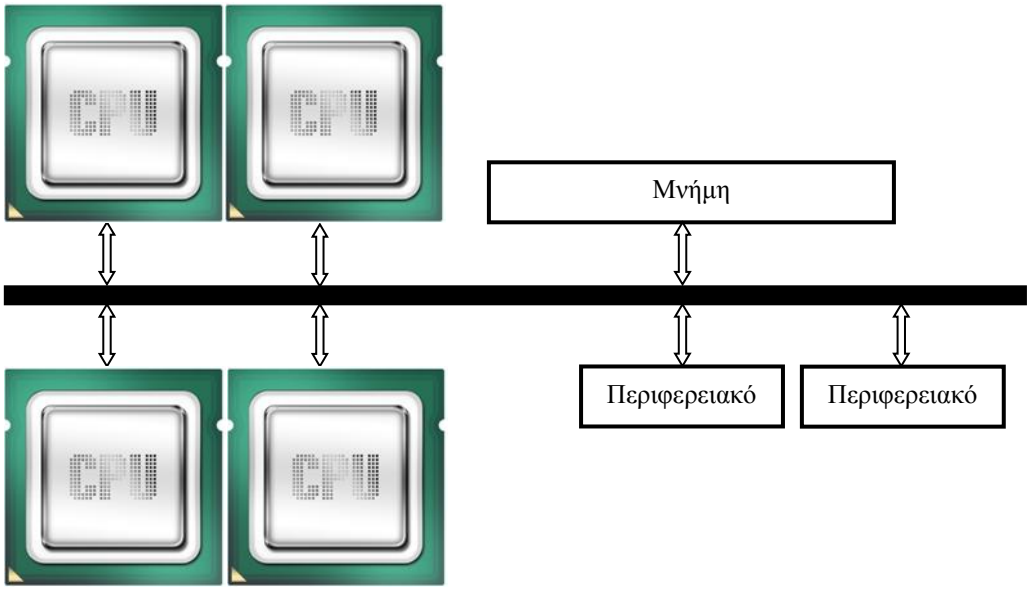
Στην πραγματικότητα, στις πιο γρήγορες αρχιτεκτονικές απλού διαύλου με επεξεργαστές που διαθέτουν πολλαπλά επίπεδα κρυφής μνήμης, το εύρος λειτουργίας της μνήμης είναι αρκετό για να υποστηριχθούν έως και 20 επεξεργαστές χωρίς να παρατηρηθούν προβλήματα αντιπαραθέσεων μεταξύ των επεξεργαστών κατά την πρόσβασή τους στην κύρια μνήμη του συστήματος χρησιμοποιώντας unit stride [3]. Αν οι επεξεργαστές έχουν πρόσβαση στη μνήμη χωρίς την χρήση unit stride, τότε μπορεί να παρατηρηθεί συμφόρηση του διαύλου και σε χαμηλότερο αριθμό επεξεργαστών.

1.4 Απλοί και σύνθετοι δίαυλοι επικοινωνίας

Οι δύο πιο κοινές αρχιτεκτονικές βάσεις για την κατασκευή πολυπύρηνων συστημάτων με διαμοιραζόμενη μνήμη είναι οι απλοί δίαυλοι επικοινωνίας (busses) και οι ραβδεπαφικοί σύνθετοι δίαυλοι επικοινωνίας (crossbars). Άλλες μορφές σύνθετων διαύλων, στις οποίες δεν θα γίνει ιδιαίτερη αναφορά στην παρούσα εργασία, είναι οι σύνθετοι δίαυλοι διαπλοκής-εναλλαγής και πεταλούδας.

1.4.1 Απλός δίαυλος επικοινωνίας (Bus)

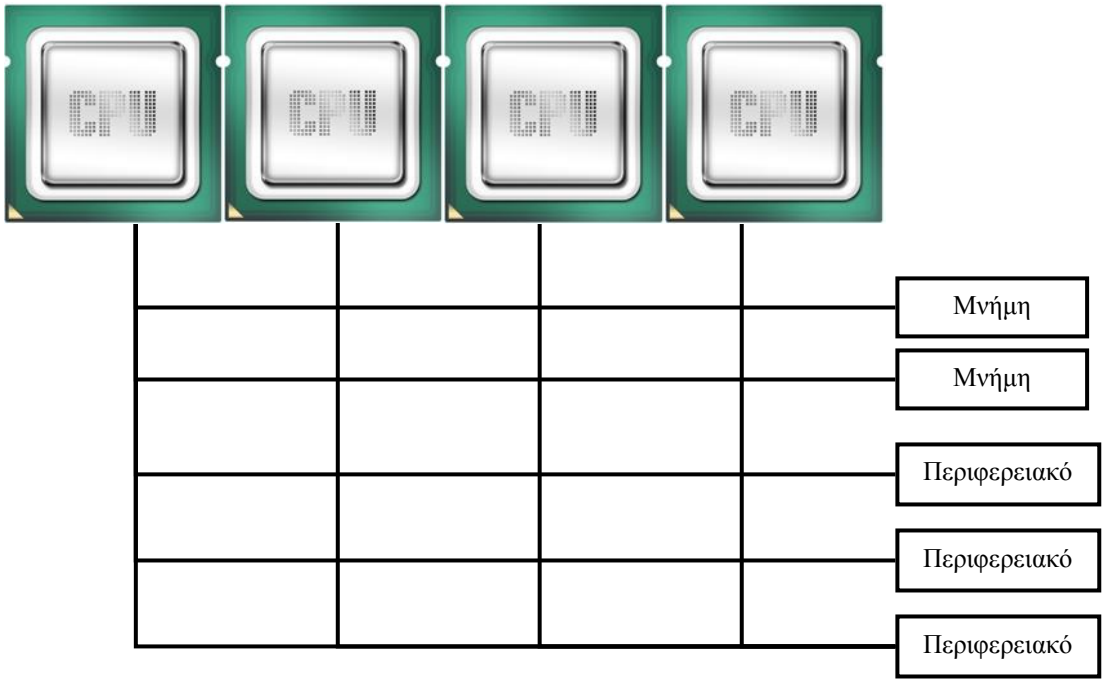
Ο απλός δίαυλος επικοινωνίας (Σχήμα 1.7) αποτελεί την απλούστερη προσέγγιση για την σύνδεση μιας σειράς επεξεργαστών με την μνήμη και τα υπόλοιπα τμήματα ενός υπολογιστή. Μία τέτοια μορφή διαύλου μπορεί να παρασταθεί ως δύο παράλληλα καλώδια τα οποία συνδέουν τα δομικά στοιχεία του υπολογιστή (επεξεργαστές, μνήμη, σκληροί δίσκοι, κάρτες γραφικών και άλλα περιφερειακά), ένα σύνολο πρωτοκόλλων για την μεταξύ τους επικοινωνία (ελεγκτές μνήμης και λοιπών συσκευών) και φυσικά, το σύνολο του υλισμικού που αναλαμβάνει να την πραγματοποιήσει. Ο απλός δίαυλος επικοινωνίας είναι η φθηνότερη και απλούστερη κατασκευαστικά λύση για να στηθεί ένα πολυπύρηνο υπολογιστικό σύστημα. Το γεγονός όμως ότι κάθε είδους επικοινωνία των επεξεργαστών με τα μέρη του υπολογιστή πραγματοποιείται από τον ίδιο δίαυλο μπορεί να προκαλέσει φαινόμενα συμφόρησης του διαύλου και πτώση των επιδόσεων του υπολογιστικού συστήματος. Αποτέλεσμα του παραπάνω μειονεκτήματος είναι η αδυναμία μίας διάταξης απλού διαύλου να εξυπηρετήσει πολύ μεγάλο αριθμό επεξεργαστών.



Σχήμα 1.7
Αρχιτεκτονική απλού διαύλου

1.4.2 Ραβδεπαφικός σύνθετος δίαυλος επικοινωνίας (Crossbar)

Ένας ραβδεπαφικός δίαυλος επικοινωνίας αποτελεί την υλισμική προσέγγιση που στοχεύει στην εξάλειψη της συμφόρησης που προκαλείται σε έναν απλό δίαυλο. Τα ραβδεπαφικά δίκτυα είναι ουσιαστικά μία σειρά απλών διαύλων, οι οποίοι λειτουργούν παράλληλα και συνδέονται με κάθε κομμάτι του υπολογιστή (επεξεργαστής, μνήμη και περιφερειακά). Με αυτόν τον τρόπο, κάθε μέρος του υπολογιστή μπορεί να επικοινωνεί με οποιοδήποτε άλλο χρησιμοποιώντας μία διαδρομή μέσα στον σύνθετο δίαυλο, ενώ πάνω από μία διασυνδέσεις μπορούν να είναι ταυτοχρόνως ενεργές. Όπως φαίνεται και στο Σχήμα 1.8, σε έναν ραβδεπαφικό δίαυλο 4x5 μπορούν να πραγματοποιηθούν ταυτόχρονα 4 διεργασίες μεταφοράς δεδομένων.



Σχήμα 1.8
Αρχιτεκτονική ραβδεπαφικού σύνθετου διαύλου (4x5)

Φυσικά, παρότι η απεικόνιση ενός ραβδεπαφικού διαύλου είναι εύκολη, η κατασκευή του είναι ιδιαίτερα πολύπλοκη και με αρκετά μεγάλο κόστος. Το γεγονός αυτό οφείλεται στο ότι ένας σύνθετος δίαυλος δεν συνδέει απλά τα μέρη που επιθυμούν να επικοινωνούν, αλλά πρέπει επίσης να διαιτητεύει μεταξύ δύο ή περισσοτέρων επεξεργαστών που επιθυμούν να έχουν πρόσβαση στην ίδια περιοχή μνήμης ή το ίδιο περιφερειακό. Στην περίπτωση που ένα στοιχείο υλισμικού είναι ιδιαίτερα «δημοφιλές», ο σύνθετος δίαυλος καλείται να αποφασίσει ποιος θα έχει πρόσβαση και ποιος όχι. Έτσι, παρά το ότι αποτελούν την αιχμή στον τομέα των επιδόσεων, το γεγονός ότι το κόστος κατασκευής τους αυξάνει δραματικά όσο αυξάνεται ο αριθμός των θυρών που χρησιμοποιούν έχει ως αποτέλεσμα οι σύνθετοι δίαυλοι επικοινωνίας να χρησιμοποιούνται μόνο σε μηχανήματα ιδιαίτερα υψηλών επιδόσεων και κόστους.

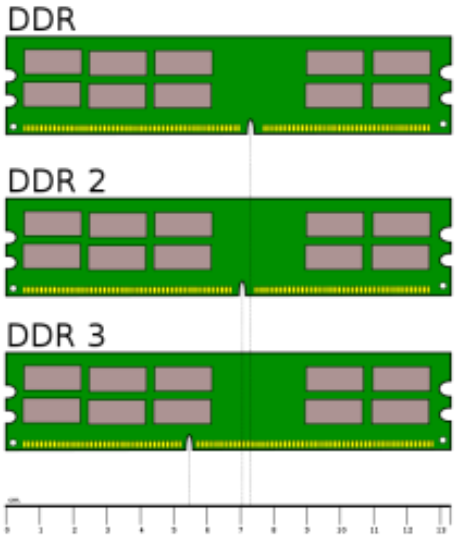
1.5 Μνήμη

Δεδομένης της σημασίας των επιδόσεων του υποσυστήματος μνήμης ενός υπολογιστή, πολλές τεχνικές έχουν χρησιμοποιηθεί για τη βελτίωση αυτού. Τα δύο χαρακτηριστικά του συστήματος μνήμης που παίζουν τον σημαντικότερο ρόλο στην απόδοσή του είναι το εύρος επικοινωνίας με τους επεξεργαστές (memory bandwidth) και ο λανθάνων χρόνος αντίδρασης της μνήμης (memory latency). Οι αλλαγές στον

σχεδιασμό του συστήματος μνήμης ενός υπολογιστή συνήθως βελτιώνουν ένα εκ των δύο χαρακτηριστικών σε βάρος του άλλου, ενώ σε ελάχιστες περιπτώσεις μπορούμε να έχουμε βελτιώσεις με θετικό αντίκτυπο τόσο στο εύρος επικοινωνίας με τους επεξεργαστές όσο και στον λανθάνων χρόνο αντίδρασης της μνήμης. Το εύρος επικοινωνίας εστιάζεται γενικά στον ταχύτερο δυνατό ρυθμό μεταφοράς δεδομένων υπό σταθερή κατάσταση από το υποσύστημα της μνήμης. Ο ρυθμός αυτός συνήθως μετριέται ενώ εκτελείται ένας μεγάλος unit-stride βρόχος ανάγνωσης και εγγραφής στη μνήμη. Ο λανθάνων χρόνος είναι ουσιαστικά το μέτρο της απόδοσης του συστήματος μνήμης στην χειρότερη περίπτωση, όπου καλείται να μετακινήσει μια μικρή ποσότητα δεδομένων 32 ή 64-bit. Φυσικά, στον τομέα της υπολογιστικής ανάλυσης υψηλών επιδόσεων και τα δύο αυτά μεγέθη είναι εξίσου σημαντικά αφού αποτελούν σημαντικό μέρος των περισσότερων εφαρμογών υψηλής απόδοσης.

1.5.1 Πρότυπα μνήμης DDR SDRAM (Double Data Rate Synchronous Dynamic Random-Access Memory)

Από το 1996 έως το 2000, η JEDEC (Joint Electron Devices Engineering Council) ανέπτυξε το πρότυπο ολοκληρωμένου κυκλώματος μνήμης DDR. Οι μνήμες DDR SDRAM ήρθαν για να αντικαταστήσουν τις παλαιότερες SDR SDRAM (Single Data Rate Synchronous Dynamic Random-Access Memory) και να καλύψουν τις συνεχώς αυξανόμενες απαιτήσεις για μεγαλύτερο εύρος επικοινωνίας των επεξεργαστών με τις μνήμες αλλά και για μικρότερους λανθάνοντες χρόνους. Η μεγάλη διαφορά του προτύπου DDR σε σύγκριση με το SDR έγκειται στο γεγονός ότι σε έναν κύκλο ρολογιού εκτελείται μία εντολή που διαβάζει και γράφει τα διπλάσια δεδομένα από και προς την μνήμη, καθώς μεταφέρονται δεδομένα και κατά την ακμή ανόδου και κατά την ακμή καθόδου του σήματος του ρολογιού του συστήματος. Κατά αυτόν τον τρόπο επιτυγχάνεται σχεδόν διπλασιασμός του ρυθμού μεταφοράς κρατώντας σταθερή την συχνότητα λειτουργίας του διαύλου επικοινωνίας του



Σχήμα 1.9
Τα υπάρχοντα πρότυπα DDR SDRAM

επεξεργαστή με την μνήμη και άρα τον λανθάνων χρόνο της μνήμης σε χαμηλά επίπεδα.

Πιο αναλυτικά, ο μέγιστος ρυθμός μεταφοράς δεδομένων για μία μνήμη SDR με συχνότητα λειτουργίας 100MHz προκύπτει από την σχέση :

$$\frac{100MHz \cdot 64bit}{8\,bit/byte} = 800\,MB/s$$

Αντίστοιχα, για μία μνήμη DDR αντίστοιχης συχνότητας :

$$\frac{100MHz \cdot 2 \cdot 64bit}{8\,bit/byte} = 1600\,MB/s$$

όπου, ο πολλαπλασιαστής «2» εισάγεται στην σχέση υπολογισμού του μέγιστου ρυθμού μεταφοράς δεδομένων λόγω της διπλής πρόσβασης στην μνήμη σε κάθε κύκλο ρολογιού.

Στον πίνακα που ακολουθεί παρατίθενται τα υπάρχοντα πρότυπα DDR SDRAM που χρησιμοποιούνται σε επεξεργαστές :

Πρότυπο	Ταχύτητα διαύλου (MHz)	Εσωτερική συχνότητα (MHz)	Ρυθμός μεταφοράς (MT/s)	Συνήθης τάση λειτουργίας (V)
DDR	100-200	100-200	200-400	2.5/2.6
DDR2	200-533	100-266	400-1066	1.8
DDR3	400-1066	100-266	200-2133	1.5

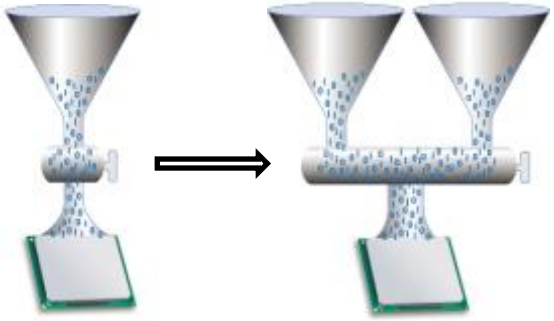
1.5.2 Αρχιτεκτονική πολλαπλών καναλιών μνήμης

Η αρχιτεκτονική πολλαπλών καναλιών αποτελεί μια τεχνολογία η οποία χρησιμοποιείται για αν αυξηθεί ο ρυθμός μεταφοράς δεδομένων μεταξύ της RAM και του ελεγκτή μνήμης του επεξεργαστή. Η εν λόγω αύξηση επιτυγχάνεται με την προσθήκη επιπλέον καναλιών επικοινωνίας μεταξύ ελεγκτή και μνήμης. Θεωρητικά, ο ρυθμός μεταφοράς δεδομένων είναι πολλαπλάσιος του αριθμού καναλιών που χρησιμοποιούνται. Η πιο απλή μορφή αρχιτεκτονικής πολλαπλών καναλιών, το διπλό κανάλι μνήμης, χρησιμοποιεί δύο κανάλια, ενώ σε εξειδικευμένους υπολογιστές ιδιαίτερα υψηλών επιδόσεων έχουν χρησιμοποιηθεί chipset που μπορούν να υποστηρίξουν μέχρι και οχτώ κανάλια. Στους υπολογιστές ευρείας χρήσης, χρησιμοποιούνται οι τεχνολογίες διπλού, τριπλού και σε περιορισμένες περιπτώσεις τετραπλού καναλιού.

1.5.2.1 Σκοπός των πολλαπλών καναλιών μνήμης

Η τεχνολογία πολλαπλών καναλιών μνήμης δημιουργήθηκε με σκοπό να επιλύσει θέματα συμφόρησης που προέκυπταν κατά την επικοινωνία των επεξεργαστών με την μνήμη του συστήματος. Οι συνεχώς αυξανόμενες επιδόσεις των επεξεργαστών απαιτούν εξελίξεις στα «δευτερεύοντα» τμήματα ενός υπολογιστή, ώστε αυτά να συμβαδίζουν μεταξύ τους. Στην περίπτωση των πολλαπλών καναλιών μνήμης ο στόχος ήταν η βελτίωση των επιδόσεων του ελεγκτή μνήμης, ο οποίος ρυθμίζει την ροή δεδομένων μεταξύ επεξεργαστών και μνήμης RAM. Ο ελεγκτής μνήμης καθορίζει τον τύπο και την ταχύτητα της RAM, όπως επίσης το μέγιστο μέγεθος που μπορεί να έχει κάθε άρθρωμα μνήμης και το σύνολο μνήμης που μπορεί να χρησιμοποιηθεί από έναν υπολογιστή. Όταν ο ελεγκτής δεν συμβαδίζει με την ταχύτητα του επεξεργαστή, δημιουργούνται συμφορήσεις, αφήνοντας έναν ή περισσότερους επεξεργαστές χωρίς δεδομένα προς επεξεργασία. Υπό συνθήκες απλού καναλιού μνήμης, οποιοσδήποτε επεξεργαστής με δίαυλο ταχύτερο από την συχνότητα της μνήμης καθίσταντο ευπαθής σε φαινόμενα συμφόρησης, φαινόμενο που γίνεται πιο έντονο όσο ο αριθμός των επεξεργαστών αυξάνεται.

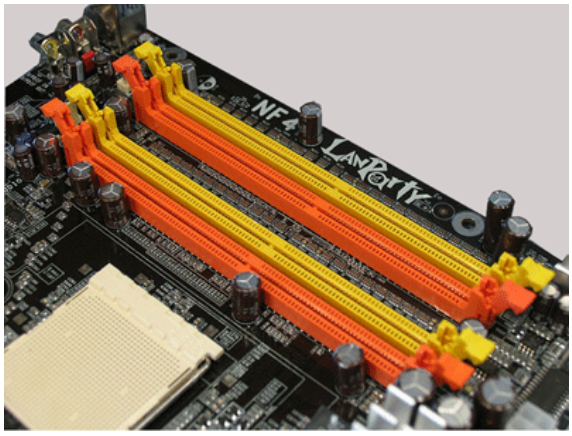
Η διαμόρφωση πολλαπλού καναλιού περιορίζει το πρόβλημα των συμφορήσεων στον ελεγκτή μνήμης, πολλαπλασιάζοντας το διαθέσιμο εύρος επικοινωνίας με την μνήμη ανάλογα με τον αριθμό των καναλιών που χρησιμοποιούνται. Κατά αυτόν τον τρόπο, οι ελεγκτές μνήμης μπορούν να ανταπεξέλθουν στις απαιτήσεις των ταχύτερων επεξεργαστών της αγοράς αλλάζοντας ουσιαστικά τον τρόπο με τον οποίο διαχειρίζονται την ήδη υπάρχουσα τεχνολογία μνήμης και όχι με αύξηση της ταχύτητας λειτουργίας των αρθρωμάτων μνήμης.



Σχήμα 1.10
Η προσθήκη ενός επιπλέον καναλιού επικοινωνίας με την μνήμη διπλασιάζει το διαθέσιμο εύρος επικοινωνίας με αυτήν και περιορίζει τα φαινόμενα συμφόρησης.

1.5.2.2 Αρχιτεκτονική διπλού καναλιού μνήμης

Οι ελεγκτές μνήμης με ενσωματωμένο διπλό κανάλι μνήμης, χρησιμοποιούν δύο κανάλια δεδομένων των 64bit έκαστο. Στην περίπτωση που η μητρική κάποιου υπολογιστικού συστήματος είναι εξοπλισμένη με chipset που υποστηρίζει δύο κανάλια επικοινωνίας με την RAM, για την χρήση τους απαιτούνται δύο ή περισσότερα αρθρώματα (modules) μνήμης τύπου DDR, DDR2 ή DDR3 SDRAM. Τα αρθρώματα τοποθετούνται ανά δύο σε υποδοχές οι οποίες έχουν κοινό χρωματισμό και αντιπροσωπεύουν τα διαφορετικά κανάλια επικοινωνίας. Τα αρθρώματα που τοποθετούνται σε υποδοχές με ίδιο χρωματισμό (Σχήμα 1.11), έχουν την δυνατότητα της ταυτόχρονης επικοινωνίας με τον ελεγκτή μνήμης, αυξάνοντας έτσι το συνολικό εύρος επικοινωνίας του επεξεργαστή με την RAM.



Σχήμα 1.11
Οι υποδοχές μνήμης με ίδιο χρωματισμό υποδηλώνουν τα σημεία που τοποθετούνται ζευγάρια αρθρωμάτων μνήμης για να δουλέψουν σε διαμόρφωση διπλού καναλιού.

Στις περισσότερες περιπτώσεις (ανάλογα με την μητρική που χρησιμοποιείται) δεν είναι αναγκαίο να τοποθετηθούν πανομοιότυπα αρθρώματα μνήμης στις υποδοχές ίδιου χρώματος, ώστε να λειτουργήσουν τα δύο κανάλια. Στην περίπτωση που θα χρησιμοποιηθούν αρθρώματα με διαφορετικές συχνότητες λειτουργίας, η μητρική θα επιλέξει από μόνη της να δουλέψει ολόκληρη την συστοιχία μνήμης στην συχνότητα του πιο αργού αρθρώματος, ώστε να επιτευχθεί σταθερή λειτουργία του συστήματος. Επίσης, μπορεί να χρησιμοποιηθεί μία μίξη από single-sided και double-sided μνήμες, διατρέχοντας όμως ένα ρίσκο, ανάλογα με τις δυνατότητες του ελεγκτή μνήμης, να υπάρξουν πιθανές αστάθειες στην λειτουργία του συστήματος, ιδιαίτερα κατά την εκτέλεση μνημοβόρων εφαρμογών.

Παρόλη την ελαστικότητα στην λειτουργία της αρχιτεκτονικής διπλού καναλιού μνήμης, εάν ο χρήστης επιθυμεί να έχει βέλτιστες επιδόσεις από το υπολογιστικό του σύστημα, θα πρέπει στις υποδοχές μνήμης ίδιου χρώματος να τοποθετήσει πανομοιότυπα αρθρώματα μνήμης. Αυτός είναι και ο λόγος που πλέον όλες οι

κατασκευάστριες μνήμης προσφέρουν «πακέτα» που αποτελούνται από ζευγάρια πανομοιότυπων αρθρωμάτων για χρήση σε διπλό κανάλι.

1.5.2.3 Ganged vs Unganged Dual-Channel DDR

Το διπλό κανάλι μνήμης αναπτύχθηκε αρχικά με σκοπό να αυξηθεί το εύρος επικοινωνίας του επεξεργαστή με την μνήμη, αθροίζοντας τα δύο ξεχωριστά κανάλια των 64bit σε ένα εικονικό κανάλι με εύρος 128bit (ganged mode). Η εξέλιξη όμως των επεξεργαστών και η στροφή της βιομηχανίας προς τα πολυπύρηνα CPU ώθησε στην σταδιακή εγκατάλειψη αυτής της μορφής διπλού καναλιού και στην υιοθέτηση της «απλής» μορφής (unganged mode), όπου τα δύο κανάλια των 64bit χρησιμοποιούνται ξεχωριστά. Κατά αυτόν τον τρόπο, σε συστήματα με πολλαπλά CPU, οι διαφορετικοί πυρήνες μπορούν να έχουν ταυτόχρονη πρόσβαση σε διαφορετικά αρθρώματα μνήμης.

1.5.2.4 Αρχιτεκτονική τριπλού καναλιού μνήμης

Προς το παρόν και σε υπολογιστές «ευρείας χρήσης», αρχιτεκτονική τριών καναλιών μνήμης χρησιμοποιείται μόνο σε συστήματα εξοπλισμένα με επεξεργαστές που τοποθετούνται σε πλατφόρμα (socket) LGA1366 , σε συνδυασμό με μνήμες DDR3 SDRAM. Κατώτερα CPU της Intel (Core i7-800, Core i5 και Core i3), όπως επίσης και όλοι οι επεξεργαστές της AMD χρησιμοποιούν μέχρι δύο κανάλια μνήμης σε συνδυασμό με μνήμες DDR3 ή DDR2 SDRAM. Πιο αναλυτικά οι επεξεργαστές που ενσωματώνουν ελεγκτή μνήμης με αρχιτεκτονική τριπλού καναλιού είναι :

<table>
<tr><td>Intel Core i7</td><td colspan="3">Intel Xeon</td></tr>
<tr><td rowspan="2">9xx Bloomfield, Gulftown</td><td>E55xx Nehalem-EP</td><td>L55xx Nehalem-EP</td><td rowspan="3">Wxxxx Bloomfield, Nehalem-EP, Westmere-EP</td></tr>
<tr><td>E56xx Westmere-EP</td><td>L5609 Westmere-EP</td></tr>
<tr><td rowspan="3">9x0X Gulftown</td><td>ECxxxx Jasper Forest</td><td>L5630 Westmere-EP</td></tr>
<tr><td></td><td>L5640 Westmere-EP</td><td>X55xx Nehalem-EP</td></tr>
<tr><td></td><td>LC55x8 Jasper Forest</td><td>X56xx Westmere-EP</td></tr>
</table>

Σύμφωνα με την Intel, ένα σύστημα βασισμένο σε Core i7 με μνήμες DDR3 συχνότητας λειτουργίας στα 1066MHz, μπορεί να αγγίξει ρυθμούς μεταφοράς δεδομένων της τάξης των 25.6GB/s, όταν λειτουργεί σε διαμόρφωση τριπλού καναλιού μνήμης.

Η αρχιτεκτονική αυτή μπορεί να χρησιμοποιηθεί μόνο όταν τοποθετηθούν στην μητρική τρία (ή πολλαπλάσια του τρία) αρθρώματα μνήμης της ίδιας ταχύτητας και

χωρητικότητας. Σε περίπτωση που τοποθετηθούν δύο (ή πολλαπλάσια του δύο) αρθρώματα, το σύστημα θα λειτουργήσει σε διαμόρφωση διπλού καναλιού.

1.5.2.5 Αρχιτεκτονική τετραπλού καναλιού μνήμης

Η εν λόγω αρχιτεκτονική χρησιμοποιείται μόνο από επεξεργαστές οι οποίοι τοποθετούνται σε πλατφόρμα LGA 2011. Πιο αναλυτικά οι επεξεργαστές που ενσωματώνουν ελεγκτή μνήμης με αρχιτεκτονική τετραπλού καναλιού είναι :

Intel Core i7	Intel Xeon
3960X	E5-16xx
3930K	E5-26xx
3820	

Για να λειτουργήσει μία μητρική σε διαμόρφωση τετραπλού καναλιού μνήμης, θα πρέπει να τοποθετηθούν τέσσερα ή πολλαπλάσια του τέσσερα αρθρώματα μνήμης της ίδιας ταχύτητας και χωρητικότητας. Σε οποιαδήποτε άλλη περίπτωση, το σύστημα θα δουλέψει σε διαμόρφωση διπλού ή τριπλού καναλιού.

1.6 Παράρτημα

[1] FLOPS (FLoating point OPerations per Second)

Στην επιστήμη των υπολογιστών και ιδιαίτερα στον τομέα της επιστημονικής έρευνας όπου πραγματοποιείται εκτενής χρήση υπολογισμών κινητής υποδιαστολής, τα FLOPS αποτελούν το μέτρο σύγκρισης στις επιδόσεις ενός υπολογιστικού συστήματος. Σε σχετική αναλογία με το παλαιότερο μέτρο σύγκρισης των εντολών ανά δευτερόλεπτο (instructions per second), τα FLOPS λαμβάνουν υπόψη τους ανά δευτερόλεπτο υπολογισμούς κινητής υποδιαστολής που μπορεί να πραγματοποιήσει ένας επεξεργαστής. Τα πιο συνήθη τεστ αναφοράς (benchmarks) που χρησιμοποιούνται για την αξιολόγηση των επεξεργαστών περιέχονται στις βιβλιοθήκες γραμμικής άλγεβρας LINPACK, ενώ αξίζει να σημειωθεί ότι ο αριθμός των FLOPS που μπορεί να αποδώσει ένας επεξεργαστής εξαρτώνται από την ακρίβεια των υπολογισμό κινητής υποδιαστολής (single ή double precision).

[2] Τοπικότητα της πρόσβασης μνήμης

Όλα τα προγράμματα υπολογιστών παρουσιάζουν την ιδιότητα της τοπικότητας στην πρόσβαση μνήμης. Σύμφωνα με την ιδιότητα αυτή, οι θέσεις μνήμης που προσπελαύνονται από τον επεξεργαστή στο άμεσο παρελθόν έχουν την μεγαλύτερη πιθανότητα να προσπελαστούν και στο άμεσο μέλλον.

[3] Unit Stride

Στον προγραμματισμό ηλεκτρονικών υπολογιστών, ο βηματισμός μίας αράδας (array stride) αναφέρεται στον αριθμό θέσεων μνήμης (memory locations) που παρεμβάλλονται μεταξύ διαδοχικών στοιχείων της αράδας. Το παραπάνω μετράται σε bytes ή στην μονάδα μεγέθους των στοιχείων της αράδας.

Μία αράδα με βηματισμό 1, έχει στοιχεία τα οποία είναι συνεχόμενα στην μνήμη. Τέτοιες αράδες συχνά αναφέρεται ότι διαθέτουν μοναδιαίο βηματισμό (unit stride). Οι αράδες unit stride είναι γενικά πιο αποδοτικές από τις non-unit stride, λόγω των επιδράσεων της κρυφής μνήμης.

Κεφάλαιο 2ο

2.1 Εισαγωγή

Το πακέτο PARDISO (Parallel sparse direct and multi-recursive solvers) είναι ένα σύνολο μαθηματικών βιβλιοθηκών που αποτελείται από ρουτίνες OpenMP [1] και BLAS [2], για την απευθείας παράλληλη επίλυση μεγάλων, αραιών, συμμετρικών και μη-συμμετρικών γραμμικών συστημάτων εξισώσεων σε πολυπύρηνους επεξεργαστές με διαμοιραζόμενη μνήμη.

Ο PARDISO υπολογίζει την λύση ενός συστήματος αραιών γραμμικών εξισώσεων με πολλαπλά δεξιά μέλη,

$$AX = B$$

χρησιμοποιώντας παράλληλες παραγοντοποιήσεις LU [3], LDL^T ή LL^T [4], όπου τα A, X και B είναι πίνακες n επί n, n επί n και n επί nrhs (number of right-hand sides) αντίστοιχα. Ο PARDISO υποστηρίζει μία ευρεία οικογένεια αραιών πινάκων και υπολογίζει την λύση πραγματικών ή μιγαδικών, συμμετρικών, διαρθρωτικά συμμετρικών ή μη-συμμετρικών, αόριστων ή Ερμιτιανών αραιών γραμμικών συστημάτων εξισώσεων σε πολυπύρηνες αρχιτεκτονικές με διαμοιραζόμενη μνήμη.

Αραιοί πίνακες που μπορούν να επιλυθούν με την χρήση του PARDISO :

- Συμμετρικοί
  - Πραγματικοί
    - Αόριστοι
    - Θετικά ορισμένοι
  - Ερμιτιανοί
    - Αόριστοι
    - Θετικά ορισμένοι
  - Μιγαδικοί
- Μη συμμετρικοί
  - Πραγματικοί
  - Μιγαδικοί

2.2 Υποστηριζόμενοι πίνακες και βήματα επίλυσης

Το πακέτο PARDISO πραγματοποιεί τα παρακάτω βήματα, ανάλογα με την δομή του πίνακα A :

2.2.1 Συμμετρικοί πίνακες

Συμμετρικός αραιός πίνακας είναι ο αραιός πίνακας για τον οποίο ισχύει ότι $a(i,j) = a(j,i)$, για κάθε i και j :

$$A = \begin{bmatrix} 7 & 1 & 3 & 0.5 & 2 \\ 1 & 0.5 & 0 & 0 & 0 \\ 3 & 0 & 3 & 0 & 0 \\ 0.5 & 0 & 0 & 0.6 & 0 \\ 2 & 0 & 0 & 0 & 12 \end{bmatrix}$$

Στην περίπτωση ενός τέτοιου πίνακα, ο επιλύτης αρχικά υπολογίζει έναν συμμετρικό πίνακα ομοιότητας P ώστε να ελαχιστοποιηθούν φαινόμενα fill-in [5], βασιζόμενος είτε σε αλγόριθμο minimum degree είτε σε αλγόριθμο nested dissection από το πακέτο METIS [6]. Ακολουθεί η παράλληλη αριθμητική παραγοντοποίηση Cholesky $PAP^T = LL^T$ ή $PAP^T = LDL^T$ για συμμετρικούς μη ορισμένους πίνακες. Ο επιλύτης χρησιμοποιεί διαγώνιες περιστροφές ή 1x1 και 2x2 περιστροφές Bunch-Kaufman για συμμετρικούς μη ορισμένους πίνακες και μία προσέγγιση του X υπολογίζεται από μπρος-πίσω αντικατάσταση και επαναληπτική βελτίωση.

Ο πίνακας των συντελεστών διαταράσσεται όποτε δεν είναι δυνατό να βρεθούν αποδεκτές 1x1 και 2x2 περιστροφές μέσα σε ένα διαγώνιο μπλοκ υπερκόμβων. Σε αυτήν την περίπτωση είναι πιθανό να χρειαστούν ένα ή δύο περάσματα επαναληπτικής βελτίωσης ώστε να διορθωθούν οι επιδράσεις της διαταραχής. Αυτή η περιοριστική έννοια των περιστροφών με επαναληπτική βελτίωση είναι αποτελεσματική σε άκρως αόριστα συμμετρικά συστήματα. Επιπλέον, η ακρίβεια αυτής της μεθόδου είναι, για μεγάλο εύρος πινάκων από διάφορους τομείς εφαρμογών, το ίδιο ακριβής με μεθόδους άμεσης παραγοντοποίησης που χρησιμοποιούν τεχνικές με πλήρεις αραιές περιστροφές. Μια άλλη δυνατότητα βελτίωσης της ακρίβειας των περιστροφών είναι η χρήση αλγόριθμων συμμετρικά σταθμισμένων αντιπαραβολών. Οι μέθοδοι αυτές αναγνωρίζουν μεγάλες καταχωρίσεις στον A οι οποίες, αν μετατεθούν κοντά στην διαγώνιο, επιτρέπουν στην διαδικασία παραγοντοποίησης να αναγνωρίσει περισσότερο αποδεκτές περιστροφές και να προβεί σε λιγότερες διαταραχές των περιστροφών. Οι μέθοδοι βασίζονται σε μέγιστα σταθμισμένες αντιπαραβολές και βελτιώνουν την ποιότητα του συντελεστή

με πιο συμπληρωματικό τρόπο σε σχέση με τη χρήση τεχνικών με πιο πλήρεις περιστροφές.

2.2.2 Δομικά συμμετρικοί πίνακες

Δομικά συμμετρικός αραιός πίνακας είναι ο αραιός πίνακας που έχει μη-μηδενικά στοιχεία για τα οποία ισχύει ότι αν $a(i,j) \neq 0$, τότε $a(j,i) \neq 0$ για κάθε i και j :

$$A = \begin{bmatrix} 3 & 0 & 4 & 0 & 0 \\ 0 & 6 & 0 & 12 & 0 \\ 1 & 0 & 9 & 0 & 0 \\ 0 & 3 & 0 & 11 & 0 \\ 0 & 0 & 0 & 0 & 18 \end{bmatrix}$$

Σε αυτήν την περίπτωση ο επιλύτης αρχικά υπολογίζει έναν συμμετρικό πίνακα ομοιότητας P ώστε να ελαχιστοποιηθούν φαινόμενα fill-in, διαδικασία η οποία ακολουθείται από παράλληλη αριθμητική παραγοντοποίηση τύπου $PAP^T = QLU^T$. Ο επιλύτης χρησιμοποιεί μερικές περιστροφές στους υπερκόμβους και μία προσέγγιση του X υπολογίζεται από μπρος-πίσω αντικαταστάσεις και επαναληπτικές βελτιώσεις.

2.2.3 Μη συμμετρικοί πίνακες

Μη συμμετρικός αραιός πίνακας είναι ο αραιός πίνακας για τον οποίο δεν ισχύει ότι $a(i,j) = a(j,i)$, για κάθε i και j. Οι πίνακες αυτοί δεν ακολουθούν κάποιο συγκεκριμένο πρότυπο δομής :

$$A = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 7 & 1 & 0 & 0 \\ 3 & 0 & 12 & 0 & 0 \\ 6 & 0 & 0 & 32 & 0 \\ 0 & 0 & 0 & 5 & 8 \end{bmatrix}$$

Στην περίπτωση μη-συμμετρικού πίνακα, ο επιλύτης υπολογίζει αρχικά έναν συμμετρικό πίνακα ομοιότητας P_{MPS} και τους κλιμακωτούς πίνακες D_r και D_c με στόχο την τοποθέτηση μεγάλων καταχωρήσεων στην διαγώνιο, κίνηση η οποία ενισχύει σημαντικά την αξιοπιστία της διαδικασίας αριθμητικής παραγοντοποίησης. Στο επόμενο βήμα ο επιλύτης υπολογίζει έναν συμμετρικό πίνακα ομοιότητας P, ώστε να ελαχιστοποιηθούν φαινόμενα fill-in, βασιζόμενος στον πίνακα $P_{MPS}A + (P_{MPS}A)^T$, διαδικασία η οποία ακολουθείται από την παράλληλη αριθμητική παραγοντοποίηση $QLUR = PP_{MPS}D_rAD_cP$, με τους πίνακες περιστροφών υπερκόμβων Q, R και $P = P(P_{MPS})$ για να διατηρηθούν επαρκώς μεγάλα κομμάτια των κύκλων (cycles) του P_{MPS} σε ένα διαγώνιο μπλοκ. Όταν ο αλγόριθμος παραγοντοποίησης φτάσει στο σημείο όπου δεν είναι δυνατόν να παραγοντοποιήσει

τους υπερκόμβους με την παραπάνω στρατηγική περιστροφών, χρησιμοποιεί στρατηγική στατικών περιστροφών.

2.3 Εισαγωγή δεδομένων στον PARDISO

2.3.1 Γενικά

Είναι γενικά πολύ πιο αποδοτικό το να αποθηκεύουμε μόνο τα μη-μηδενικά στοιχεία ενός αραιού πίνακα. Υπάρχουν διάφορες μεθοδολογίες αποθήκευσης αραιών πινάκων, αλλά οι περισσότερες μοιράζονται την ίδια βασική τεχνική. Το βασικό αυτό στοιχείο είναι η αποθήκευση όλων των μη μηδενικών στοιχείων σε μία μονοδιάστατη αράδα και η παροχή μιας σειράς δευτερευουσών μονοδιάστατων αράδων, οι οποίες περιέχουν στοιχεία που υποδεικνύουν την τοποθεσία του κάθε μη-μηδενικού στοιχείου στον αρχικό αραιό πίνακα.

2.3.2 Διατάξεις αποθήκευσης για Direct Sparse Solvers

Η αποθήκευση των μη-μηδενικών στοιχείων ενός αραιού πίνακα σε μία μονοδιάστατη αράδα πραγματοποιείται είτε με σάρωση του πίνακα κατά στήλη (column-major format) είτε με σάρωση του πίνακα κατά γραμμή (row-major format) και την εγγραφή των μη-μηδενικών στοιχείων σε μία μονοδιάστατη αράδα με την σειρά που αυτά εμφανίζονται κατά την σάρωση.

Για συμμετρικούς πίνακες είναι αναγκαία η αποθήκευση μόνο των μη-μηδενικών στοιχείων που βρίσκονται από την διαγώνιο και πάνω (upper triangular format) ή μόνο των μη-μηδενικών στοιχείων που βρίσκονται από την διαγώνιο και κάτω (lower triangular format).

Ο PARDISO χρησιμοποιεί άνω τριγωνική row-major διάταξη (CSR format – Compressed Sparse Row format) : ο πίνακας συμπιέζεται γραμμή-γραμμή και για συμμετρικούς πίνακες αποθηκεύονται μόνο τα μη-μηδενικά στοιχεία που βρίσκονται από την διαγώνιο και πάνω. Η διάταξη αυτή καθορίζεται από τρεις μονοδιάστατες αράδες : *τιμές*, *στήλες* και *δείκτης γραμμής*. Ο παρακάτω πίνακας περιγράφει την λειτουργία των προαναφερθέντων αράδων σχετικά με τις τιμές των μη-μηδενικών στοιχείων και της θέσης τους στον αραιό πίνακα :

Τιμές	Μία πραγματική ή μιγαδική μονοδιάστατη αράδα η οποία περιέχει τα μη-μηδενικά στοιχεία του αραιού πίνακα. Τα μη-μηδενικά στοιχεία χαρτογραφούνται στην αράδα με χρήση της άνω τριγωνικής row-major μεθόδου αποθήκευσης που περιγράφεται παραπάνω.
Στήλες	Η τιμή του στοιχείου i της ακέραιης αυτής αράδας είναι ο αριθμός της στήλης η οποία περιέχει το i στοιχείο της αράδας τιμών.
Δείκτης γραμμής	Η τιμή του στοιχείου j της ακέραιης αυτής αράδας καταδεικνύει το στοιχείο της αράδας τιμών που είναι το πρώτο μη-μηδενικό στοιχείο της γραμμής j του αραιού πίνακα.

Ο αριθμός των στοιχείων από τα οποία αποτελούνται οι αράδες τιμών και στηλών θα ισούται με τον αριθμό των μη-μηδενικών στοιχείων του αραιού πίνακα.

Καθώς η αράδα του δείκτη γραμμής δίνει την τοποθεσία του πρώτου μη-μηδενικού στοιχείου σε μία γραμμή, και τα μη-μηδενικά στοιχεία αποθηκεύονται διαδοχικά, ο αριθμός των μη-μηδενικών στοιχείων της γραμμής i θα ισούται με την διαφορά των τιμών του i και του i+1. Για να διατηρηθεί η προαναφερθείσα σχέση και στην τελευταία γραμμή του αραιού πίνακα, προστίθεται ένα επιπλέον στοιχείο (dummy entry) στο τέλος της αράδας του δείκτη γραμμής. Η τιμή του στοιχείου αυτού ισούται με τον αριθμό των μη-μηδενικών στοιχείων συν ένα. Το γεγονός αυτό κάνει τον αριθμό των στοιχείων που αποτελούν την αράδα του δείκτη γραμμής κατά ένα μεγαλύτερο από τον αριθμό των σειρών του αραιού πίνακα.

2.3.3 Περιορισμοί στην διάταξη αποθήκευσης

Η διάταξη αποθήκευσης αραιού πίνακα για τον PARDISO πρέπει να ανταποκρίνεται σε δύο σημαντικούς περιορισμούς :

- Οι μη-μηδενικές τιμές σε μία γραμμή πρέπει να τοποθετούνται στην αράδα τιμών με τη σειρά που προκύπτουν στην γραμμή από τα αριστερά προς τα δεξιά.
- Κανένα στοιχείο της διαγωνίου δεν επιτρέπεται να παραλειφθεί από την αράδα τιμών στην περίπτωση συμμετρικού ή δομικά συμμετρικού πίνακα.

Ο δεύτερος περιορισμός συνεπάγεται ότι στην περίπτωση όπου ένας συμμετρικός ή δομικά συμμετρικός πίνακας έχει μηδενικά στοιχεία στην διαγώνιο, αυτά θα πρέπει να συμπεριληφθούν στην αράδα τιμών σαν να ήταν μη-μηδενικά στοιχεία.

Έστω ο μη-συμμετρικός πίνακας A :

$$A = \begin{bmatrix} 7 & 0 & 1 & 0 & 0 & 2 & 7 & 0 \\ 0 & -4 & 8 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 0 & 7 & 0 & 0 & 9 & 0 \\ 0 & -4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 7 & 0 & 0 & 3 & 0 & 5 \\ 0 & 17 & 0 & 0 & 0 & 0 & 11 & 0 \\ 0 & 0 & -3 & 0 & 0 & 0 & 2 & 5 \end{bmatrix}$$

και ο συμμετρικός πίνακας B :

$$B = \begin{bmatrix} 7 & 0 & 1 & 0 & 0 & 2 & 7 & 0 \\ 0 & -4 & 8 & 0 & 2 & 0 & 0 & 0 \\ 1 & 8 & 1 & 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 0 & 7 & 0 & 0 & 9 & 0 \\ 0 & 2 & 0 & 0 & 5 & -1 & 5 & 0 \\ 2 & 0 & 0 & 0 & -1 & 0 & 0 & 5 \\ 7 & 0 & 0 & 9 & 5 & 0 & 11 & 0 \\ 0 & 0 & 5 & 0 & 0 & 5 & 0 & 5 \end{bmatrix}$$

Κατά την αποθήκευση του παραπάνω πίνακα σε CSR format θα προκύψουν οι παρακάτω αράδες :

k	Μη-συμμετρικός			Συμμετρικός		
	iA(k)	jA(k)	A(k)	iB(k)	jB(k)	B(k)
1	1	1	7	1	1	7
2	5	3	1	5	3	1
3	8	6	2	8	6	2
4	10	7	7	10	7	7
5	12	2	-4	12	2	-4
6	13	3	8	15	3	8
7	16	5	2	17	5	2
8	18	3	1	18	3	1
9	21	8	5	19	8	5
10		4	7		4	7

11		7	9		7	9
12		2	-4		5	5
13		3	7		6	-1
14		6	3		7	5
15		8	5		6	0
16		2	17		8	5
17		7	11		7	11
18		3	-3		8	5
19		7	2			
20		8	5			

- Σημείωση

Ο PARDISO μπορεί να επεξεργαστεί και δομικά συμμετρικά συστήματα εξισώσεων. Αξίζει να σημειωθεί ότι, από την οπτική του επιλύτη, τα «μη-μηδενικά» στοιχεία ενός πίνακα είναι κάθε στοιχείο που είναι αποθηκευμένο στην αράδα τιμών, ακόμα κι αν η τιμή αυτού είναι μηδέν. Με αυτό το σκεπτικό, κάθε μη-συμμετρικός πίνακας μπορεί να μετατραπεί σε δομικά συμμετρικό πίνακα αν προστεθούν στρατηγικά κάποια μηδενικά στην αράδα τιμών. Για παράδειγμα, ο παρακάτω μη-συμμετρικός πίνακας A μπορεί να μετατραπεί σε δομικά συμμετρικό αν προσθέσουμε δύο εικονικά μη-μηδενικά στοιχεία :

$$A = \begin{bmatrix} 1 & -1 & 0 & -3 & 0 \\ -2 & 5 & 0 & 0 & 0 \\ 0 & 0 & 4 & 6 & 4 \\ -4 & 0 & 2 & 7 & 0 \\ 0 & 8 & 0 & 0 & -5 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & -1 & 0 & -3 & 0 \\ -2 & 5 & 0 & 0 & * \\ 0 & 0 & 4 & 6 & 4 \\ -4 & 0 & 2 & 7 & 0 \\ 0 & 8 & * & 0 & -5 \end{bmatrix}$$

Τα στοιχεία που σημειώνονται με μηδέν είναι τα μηδενικά στοιχεία του πίνακα, ενώ τα στοιχεία που σημειώνονται με αστερίσκο είναι τα μηδενικά στοιχεία που εκλαμβάνονται ως μη-μηδενικά (εικονικά μη-μηδενικά στοιχεία) και αποθηκεύονται στην αράδα τιμών.

Έτσι, θα προκύψουν οι παρακάτω αράδες εισαγωγής δεδομένων :

K	Μη-συμμετρικός			Δομικά συμμετρικός		
	iA(k)	jA(k)	A(k)	iA(k)	jA(k)	A(k)
1	1	1	1	1	1	1
2	4	2	-1	4	2	-1
3	6	4	-3	7	4	-3
4	9	1	-2	10	1	-2
5	12	2	5	13	2	5
6	14	3	4	16	5	0
7		4	6		3	4
8		5	4		4	6
9		1	-4		5	4
10		3	2		1	-4
11		4	7		3	2
12		2	8		4	7
13		5	-5		2	8
14					3	0
15					5	-5

2.4 Παράρτημα

[1] OpenMP API (Open Multi-Processing Application Program Interface)

Η διεπαφή προγραμματισμού εφαρμογών OpenMP υποστηρίζει προγραμματισμό σε C, C++ και Fortran, για πολυπύρηνα συστήματα διαμοιραζόμενης μνήμης σε όλες τις αρχιτεκτονικές λειτουργικών συστημάτων (συμπεριλαμβάνονται πλατφόρμες σε Unix και Windows NT). Η μορφολογία και το περιεχόμενο της διεπαφής καθορίζεται από κοινού από μία συνομοταξία (OpenMP Architecture Review Board) που αποτελείται από τους μεγαλύτερους κατασκευαστές λογισμικού και υλισμικού. Το OpenMP είναι ένα φορητό και επεκτάσιμο μοντέλο προγραμματισμού, το οποίο προσφέρει στους προγραμματιστές πολυπύρηνων συστημάτων διαμοιραζόμενης μνήμης ένα απλό και ευέλικτο περιβάλλον για ανάπτυξη παράλληλων εφαρμογών που απευθύνονται σε συστήματα που κυμαίνονται από απλά desktop μέχρι υπερυπολογιστές.

Η συνομοταξία ανάπτυξης (ARB) της διεπαφής OpenMP αποτελείται από τα παρακάτω μέλη :

Μόνιμα μέλη	Βοηθητικά μέλη
AMD	ANL
Cray	ASC/LLNL
Fujitsu	cOMPunity
HP	EPCC
IBM	LANL
Intel	NASA
NEC	ORNL
The Portland Group, Inc.	RWTH Aachen University
Oracle Corporation	
Microsoft	Texas Advanced Computing Center
Texas Instruments	
CAPS-Entreprise	
nVidia	

[2] BLAS (Basic Linear Algebra Subprograms)

Η βιβλιοθήκη BLAS αποτελείται από ρουτίνες που προσφέρουν στάνταρ δομικά στοιχεία για την εκτέλεση βασικών πράξεων μεταξύ διανυσμάτων και πινάκων. Αναπτύσσεται διαρκώς από το 1979 και είναι διαθέσιμη για ένα τεράστιο εύρος αρχιτεκτονικών συστημάτων, ενώ ανάλογα με το υλισμικό του συστήματος χρησιμοποιούνται συγκεκριμένες υλοποιήσεις των βιβλιοθηκών ώστε να επιτυγχάνεται μεγαλύτερη απόδοση.

Η βιβλιοθήκη χωρίζεται στα παρακάτω επίπεδα ανάλογα με την τάξη των δεδομένων και την πολυπλοκότητα των πράξεων :

Επίπεδο 1ο (Level 1 BLAS)

Οι ρουτίνες του πρώτου επιπέδου της βιβλιοθήκης χρησιμοποιούνται για πράξεις βαθμωτού με διάνυσμα και διανύσματος με διάνυσμα.

Παραδείγματα :

saxpy → $y_i = ax_i + y_i$

sscal → $y = ax$

sdot → $s = s + \sum_{i=1}^{n} x_i\, y_i$

Επίπεδο 2ο (Level 2 BLAS)

Οι ρουτίνες του δεύτερου επιπέδου της βιβλιοθήκης χρησιμοποιούνται για πράξεις πίνακα με διάνυσμα.

Παραδείγματα :

sgemv → $y = y + Ax$, όπου A πίνακας mxn

sger → $A = A + yx^T$

strsv → $Tx = b$, επίλυση τριγωνικού συστήματος

Επίπεδο 3ο (Level 3 BLAS)

Οι ρουτίνες του τρίτου επιπέδου της βιβλιοθήκης χρησιμοποιούνται για πράξεις πίνακα με πίνακα.

Παραδείγματα :

sgemm → $C = C + AB$, πολλαπλασιασμός πινάκων

strsm → $TX = B$, επίλυση τριγωνικού συστήματος με X nxn

[3] Παραγοντοποίηση LU

Στην γραμμική άλγεβρα, παραγοντοποίηση LU είναι η διαδικασία διάσπασης ενός πίνακα στο γινόμενο ενός άνω με ένα κάτω τριγωνικό πίνακα, ενώ σε κάποιες περιπτώσεις στο γινόμενο περιέχεται και ένας πίνακας μετάθεσης.

Αν A είναι ένας τετραγωνικός πίνακας nxn, η παραγοντοποίηση έχει τη μορφή :

$$A = LU$$

όπου L και U είναι αντίστοιχα ένας κάτω και άνω τριγωνικός πίνακας nxn.

Ένας αντιστρέψιμος nxn πίνακας, μπορεί να παραγοντοποιηθεί στην μορφή LU αν και μόνο αν όλες οι πρωτεύουσες ελάσσονες ορίζουσές του είναι μη μηδενικές. Στην περίπτωση αυτή, η παραγοντοποίηση είναι μοναδική αν η διαγώνιος τους ενός τριγωνικού πίνακα αποτελείται από μονάδες.

Για πίνακα nxn, η παραγοντοποίηση LU παίρνει αναλυτικά την μορφή :

$$\begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} & \dots & \alpha_{1n} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} & \dots & \alpha_{2n} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} & \dots & \alpha_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \alpha_{n1} & \alpha_{n2} & \alpha_{n3} & \dots & \alpha_{nn} \end{bmatrix} = \begin{bmatrix} \ell_{11} & 0 & 0 & \dots & 0 \\ \ell_{21} & \ell_{22} & 0 & \dots & 0 \\ \ell_{31} & \ell_{32} & \ell_{33} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \ell_{n1} & \ell_{n2} & \ell_{n3} & \dots & \ell_{nn} \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} & \dots & u_{1n} \\ 0 & u_{22} & u_{23} & \dots & u_{2n} \\ 0 & 0 & u_{33} & \dots & u_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & u_{nn} \end{bmatrix}$$

Ο υπολογισμός της παραγοντοποίησης πραγματοποιείται αν εξισώσουμε τους συντελεστές της παραπάνω σχέσης και θεωρώντας ότι τα στοιχεία της διαγωνίου του πίνακα L είναι ίσα με την μονάδα. Έτσι προκύπτουν τα παρακάτω :

$$\ell_{ij} = \begin{cases} 0 & \gamma\iota\alpha \quad i < j \\ 1 & \gamma\iota\alpha \quad i = j \\ \frac{a_{ij}^{(j)}}{a_{jj}^{(j)}} & \gamma\iota\alpha \quad i > j \end{cases} \qquad \text{και} \qquad u_{jk} = \begin{cases} 0 & \gamma\iota\alpha \quad j > k \\ a_{jk}^{(j)} & \gamma\iota\alpha \quad j \leq k \end{cases}$$

Το σύστημα λύνεται με επίλυση της $Ly = b$ ως προς y και στην συνέχεια με επίλυση της $Ux = y$ ως προς x. Επειδή και στις δύο σχέσεις οι πίνακες είναι τριγωνικοί, μπορούν να λυθούν χωρίς χρήση απαλοιφής Gauss* αλλά απευθείας με χρήση μπρος-πίσω αντικαταστάσεων.

* Απαλοιφή Gauss χρειάζεται για τον υπολογισμό της παραγοντοποίησης LU. Για αυτόν τον λόγο, η παραπάνω παραγοντοποίηση είναι αποδοτική σε περιπτώσεις όπου εξετάζουμε συστήματα με ίδιο A, αλλά διαφορετικά δεξιά μέλη (b).

[4] Παραγοντοποίηση Cholesky

Κάθε τετραγωνικός πίνακας με μη μηδενικές πρωτεύουσες ελάσσονες ορίζουσές, μπορεί να παραγοντοποιηθεί σε έναν κάτω τριγωνικό πίνακα L και έναν άνω τριγωνικό πίνακα U (Παραγοντοποίηση LU). Στην περίπτωση που ο πίνακας A είναι συμμετρικός και θετικά ορισμένος, η παραγοντοποίηση μπορεί να γίνει έτσι ώστε ο U να είναι ο συζυγής ανάστροφος του L, δηλαδή $U = L^T$, και η παραγοντοποίηση παίρνει την μορφή :

$$A = LL^T$$

Ισχύει επίσης και το αντίστροφο. Δηλαδή, αν ένας πίνακας A μπορεί να παραγοντοποιηθεί στο γινόμενο κάποιου κάτω τριγωνικού πίνακα (L) με τον συζυγή ανάστροφό του (L^T), τότε ο A θα είναι συμμετρικός και θετικά ορισμένος.

Η παραπάνω σχέση αποτελεί την παραγοντοποίηση Cholesky (André-Louis Cholesky), η οποία, όπου είναι εφαρμόσιμη, εκτελείται δύο φορές πιο γρήγορα από την παραγοντοποίηση LU. Συγκριτικά, μία παραγοντοποίηση LU απαιτεί περίπου $2n^3/3$ FLOPS, όπου n το μέγεθος του πίνακα A, ενώ μία παραγοντοποίηση Cholesky απαιτεί $n^3/3$ FLOPS.

Για τον υπολογισμό της παραγοντοποίησης θα έχουμε :

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{bmatrix} = \begin{bmatrix} \ell_{11} & 0 & 0 & \dots & 0 \\ \ell_{21} & \ell_{22} & 0 & \dots & 0 \\ \ell_{31} & \ell_{32} & \ell_{33} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \ell_{n1} & \ell_{n2} & \ell_{n3} & \dots & \ell_{nn} \end{bmatrix} \begin{bmatrix} \ell_{11} & \ell_{21} & \ell_{31} & \dots & \ell_{n1} \\ 0 & \ell_{22} & \ell_{32} & \dots & \ell_{n2} \\ 0 & 0 & \ell_{33} & \dots & \ell_{n3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \ell_{nn} \end{bmatrix}$$

Από την παραπάνω σχέση, προκύπτουν :

$$a_{11} = \ell_{11}^2 \rightarrow \ell_{11} = \sqrt{a_{11}}$$

$$a_{21} = \ell_{21}\ell_{11} \rightarrow \ell_{21} = \frac{a_{21}}{\ell_{11}}, \ell_{31} = \frac{a_{31}}{\ell_{11}}, \dots, \ell_{n1} = \frac{a_{n1}}{\ell_{11}}$$

$$a_{22} = \ell_{21}^2 + \ell_{22}^2 \rightarrow \ell_{22} = \sqrt{(a_{22} - \ell_{21}^2)}$$

$$a_{32} = \ell_{31}\ell_{21} + \ell_{32}\ell_{22} \rightarrow \ell_{32} = \frac{(a_{32} - \ell_{31}\ell_{21})}{\ell_{22}}, \dots$$

Ενώ γενικά θα ισχύει :

$$\ell_{jj} = \sqrt{a_{jj} - \sum_{k=1}^{j-1} \ell_{jk}}$$

$$\ell_{ij} = \frac{1}{\ell_{jj}} \left(a_{ij} - \sum_{k=1}^{j-1} \ell_{ik}\ell_{jk} \right), \gamma\iota\alpha\ i > j$$

Η λύση του συστήματος $Ax = b$ προκύπτει σε δύο βήματα. Αρχικά λύνουμε το σύστημα $Ly = b$ ως προς y και στη συνέχεια το σύστημα $L^T x = y$ από όπου προκύπτει το x.

Για να αποφύγουμε την λήψη τετραγωνικών ριζών (παραγοντοποίηση LDL^T) θα έχουμε :

$$A = LDL^T$$

Για πίνακα A μεγέθους nxn :

$$A = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ \ell_{21} & 1 & 0 & \dots & 0 \\ \ell_{31} & \ell_{32} & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \ell_{n1} & \ell_{n2} & \ell_{n3} & \dots & 1 \end{bmatrix} \begin{bmatrix} d_1 & 0 & 0 & \dots & 0 \\ 0 & d_2 & 0 & \dots & 0 \\ 0 & 0 & d_3 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & d_n \end{bmatrix} \begin{bmatrix} 1 & \ell_{21} & \ell_{31} & \dots & \ell_{n1} \\ 0 & 1 & \ell_{32} & \dots & \ell_{n2} \\ 0 & 0 & 1 & \dots & \ell_{n3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix}$$

Η παραπάνω μορφή μας βοηθά να αποφύγουμε τις τετραγωνικές ρίζες. Αν ο πίνακας A είναι θετικά ορισμένος, τότε τα στοιχεία της διαγωνίου του πίνακα D είναι όλα θετικά.

Αν ο πίνακας A είναι πραγματικός, προκύπτουν τα παρακάτω για τα στοιχεία των πινάκων D και L :

$$d_j = a_{jj} - \sum_{k=1}^{j-1} \ell_{jk}^2 d_k$$

$$\ell_{ij} = \frac{1}{d_j} \left(a_{ij} - \sum_{k=1}^{j-1} \ell_{ik}\ell_{jk}d_k \right), \gamma\iota\alpha\ i > j$$

Ενώ, αν ο πίνακας A είναι Ερμιτιανός προκύπτουν :

$$d_j = a_{jj} - \sum_{k=1}^{j-1} \ell_{jk} \ell_{jk}^T d_k$$

$$\ell_{ij} = \frac{1}{d_j}\left(a_{ij} - \sum_{k=1}^{j-1} \ell_{ik} \ell_{jk}^T d_k \right), για\ i > j$$

Οι παραγοντοποιήσεις LL^T και LDL^T (ο L είναι διαφορετικός σε κάθε περίπτωση) μπορούν εύκολα να συσχετιστούν ως εξής :

$$A = LDL^T = LD^{\frac{1}{2}} D^{\frac{1}{2}} L^T = LD^{\frac{1}{2}} \left(D^{\frac{1}{2}}\right)^T L^T = LD^{\frac{1}{2}} \left(LD^{\frac{1}{2}}\right)^T$$

Η τελευταία σχέση αποτελεί παραγοντοποίηση ενός πίνακα A στο γινόμενο ενός κάτω τριγωνικού πίνακα επί τον ανάστροφό του, όπως ακριβώς συμβαίνει και στην παραγοντοποίηση LL^T.

[5] Ελαχιστοποίηση φαινομένων fill-in

Φαινόμενα fill-in σε ένα πίνακα παρατηρούνται όταν, κατά την εκτέλεση ενός αλγόριθμου, κάποιες από τις μηδενικές καταχωρήσεις του αλλάζουν σε μη-μηδενικές τιμές. Για να ελαχιστοποιηθούν οι απαιτήσεις σε μνήμη, αλλά και οι αριθμητικές πράξεις που πραγματοποιούνται κατά την εκτέλεση του αλγορίθμου, είναι χρήσιμο να ελαχιστοποιούνται τα φαινόμενα fill-in εναλλάσσοντας γραμμές και στήλες στον πίνακα. Η συμβολική παραγοντοποίηση Cholesky μπορεί να χρησιμοποιηθεί για τον υπολογισμό της χειρότερης δυνατής περίπτωσης fill-in πριν την εκτέλεση της πραγματικής παραγοντοποίησης.

[6] METIS (Graph partitioning, Mesh partitioning, Matrix reordering)

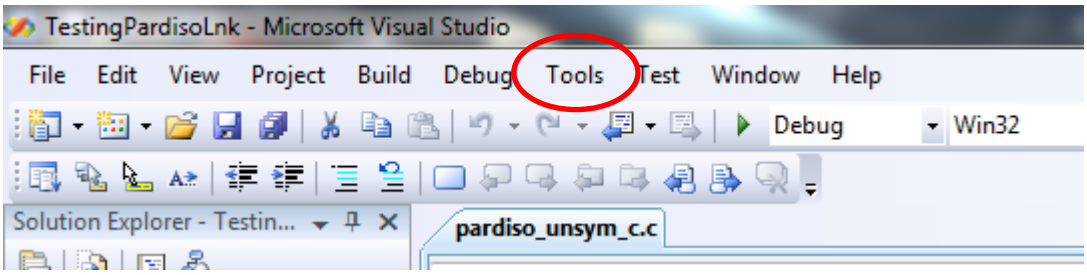
Το προγραμματιστικό πακέτο METIS περιλαμβάνει μια σειρά προγραμμάτων σχεδιασμένα για να πραγματοποιούν διαμερισμούς διαγραμμάτων και πλεγμάτων πεπερασμένων στοιχείων, όπως επίσης και αλγόριθμους που ελαχιστοποιούν τα φαινόμενα fill-in σε αραιούς πίνακες.

Κεφάλαιο 3ο

3.1 Σύνδεση του PARDISO με το Visual Studio

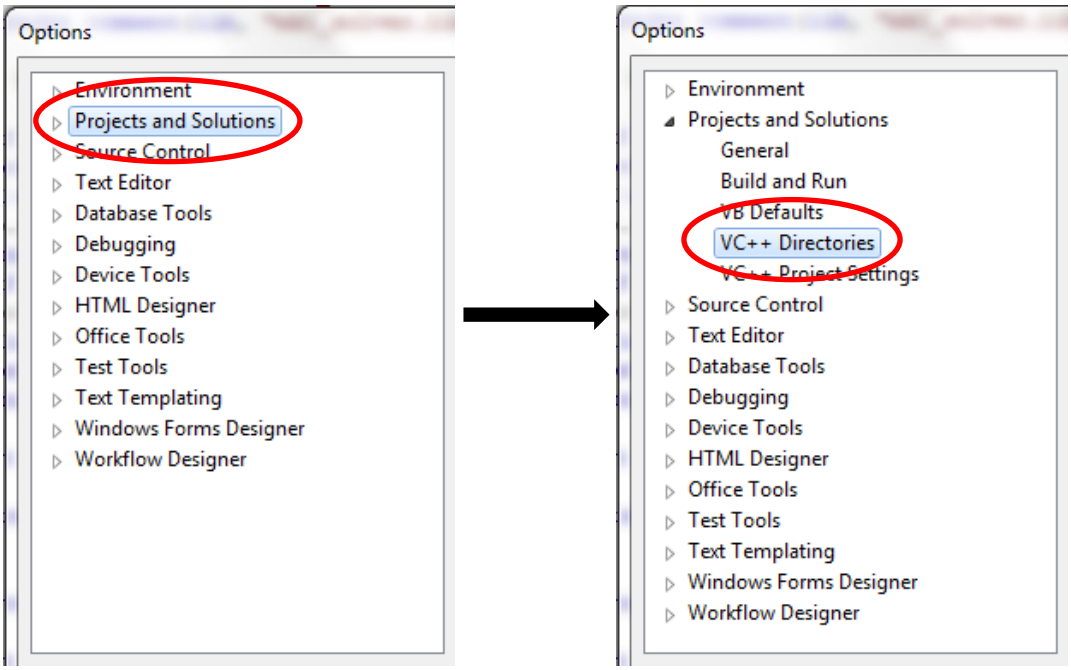
Για να συνδέσουμε τις βιβλιοθήκες του PARDISO με το οπτικό περιβάλλον προγραμματισμού Visual Studio, πραγματοποιούμε τα παρακάτω βήματα :

Βήμα 1ο



Στο tab "*Tools*" του Visual Studio επιλέγουμε "*Options*".

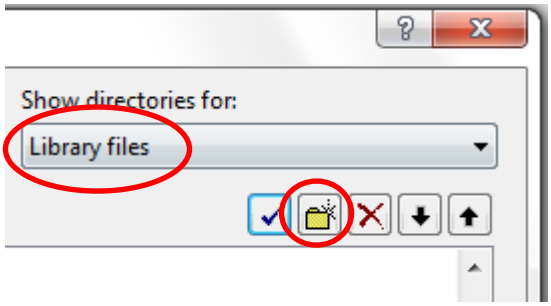
Βήμα 2ο



Στην αριστερή πλευρά του παραθύρου που ανοίγει, «ξεδιπλώνουμε» την επιλογή "*Projects and Solutions*" και επιλέγουμε "*VC++ Directories*".

Βήμα 3ο

Στην δεξιά πλευρά του παραθύρου που έχει ανοίξει, επιλέγουμε "*Show directories for: Library files*" και με το εικονίδιο "*New Line*" εισάγουμε τις τοποθεσίες στον σκληρό δίσκο όπου βρίσκονται οι απαραίτητες βιβλιοθήκες για την λειτουργία του PARDISO.



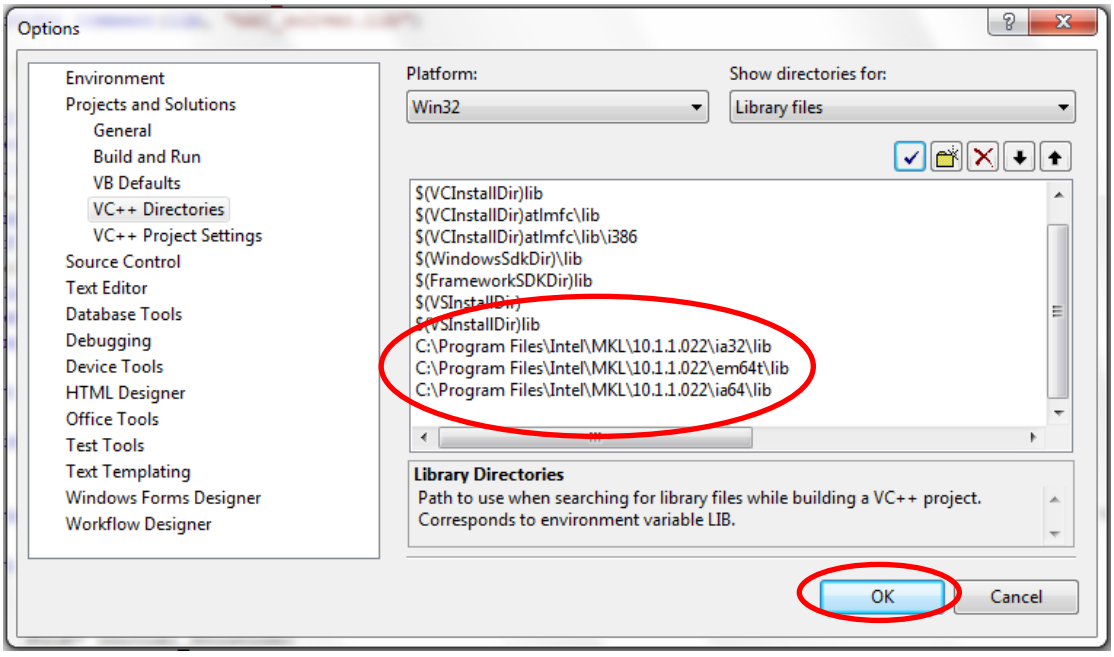
Στην περίπτωσή μας, οι βιβλιοθήκες που χρησιμοποιήθηκαν περιλαμβάνονται στο πακέτο MKL (Math Kernel Library) της Intel και οι τοποθεσίες τους στον σκληρό ήταν οι εξής :

C:\Program Files\Intel\MKL\10.1.1.022\ia32\lib

C:\Program Files\Intel\MKL\10.1.1.022\em64t\lib

C:\Program Files\Intel\MKL\10.1.1.022\ia64\lib

Βήμα 4ο



Αφού περάσουμε όλες τις τοποθεσίες για τις απαραίτητες βιβλιοθήκες πατάμε "*Ok*", ώστε να καταχωρηθούν στις επιλογές του Visual Studio.

3.2 Εκτέλεση του PARDISO

Για να καλέσουμε τον PARDISO σε ένα πρόγραμμα γραμμένο σε περιβάλλον C++, αρκεί να χρησιμοποιήσουμε την εντολή :

```
PARDISO (pt, &maxfct, &mnum, &mtype, &phase, &n, a, ia, ja, &perm,
         &nrhs, iparm, &msglvl, &b, &x, &error);
```

Στους πίνακες που ακολουθούν, παρατίθενται όλες οι παράμετροι λειτουργίας που περιέχονται στην παραπάνω εντολή και πρέπει να ρυθμίσει ο χρήστης για να τρέξει τον PARDISO. Επίσης, διατίθεται μια σύντομη περιγραφή τους και η κατηγοριοποίησή τους σε παραμέτρους εισόδου (**I**nput) και εξόδου (**O**utput). Οι τιμές των μεταβλητών που σημειώνονται με αστερίσκο, αποτελούν τις προεπιλεγμένες τιμές που χρησιμοποιούνται από τον επιλύτη.

Πίνακας 3.1
Επισκόπηση των παραμέτρων λειτουργίας του PARDISO

Όνομα	Τύπος /Τιμή	Περιγραφή	Εισαγωγή /Εξαγωγή
pt(64)	Int	Εσωτερικός pointer μνήμης	I/O
maxfct	Int	Αριθμός αριθμητικών παραγοντοποιήσεων στην μνήμη	I
mnum	Int	Πραγματικός πίνακας προς παραγοντοποίηση	I
mtype	Int	Τύπος πίνακα	I
	1	Πραγματικός και δομικά συμμετρικός	
	2	Πραγματικός και θετικά ορισμένος συμμετρικός	
	-2	Πραγματικός και αόριστος συμμετρικός	
	11	Πραγματικός και μη-συμμετρικός	
	3	Μιγαδικός και δομικά συμμετρικός	
	4	Μιγαδικός και θετικά ορισμένος Ερμιτιανός	
	-4	Μιγαδικός και αόριστος Ερμιτιανός	
	6	Μιγαδικός και συμμετρικός	
	13	Μιγαδικός και μη-συμμετρικός	
phase	Int	Φάση εκτέλεσης του επιλύτη	I
	11	Ανάλυση	
	12	Ανάλυση, αριθμητική παραγοντοποίηση	
	13	Ανάλυση, αριθμητική παραγοντοποίηση, επίλυση	
	22	Αριθμητική παραγοντοποίηση	
	23	Αριθμητική παραγοντοποίηση, επίλυση	
	33	Επίλυση, επαναληπτική βελτίωση	
	-1	Απελευθέρωση της μνήμης για όλους τους πίνακες	
	0	Απελευθέρωση της μνήμης για τον πίνακα mnum	
n	Int	Αριθμός εξισώσεων	I
a(*)	R/C	Μη-μηδενικά στοιχεία του πίνακα συντελεστών A	I
ia(n+1)	Int	Αράδα δείκτης γραμμής για δομή CSR	I
ja(*)	Int	Αράδα στήλης για δομή CSR	I

Όνομα	Τύπος /Τιμή	Περιγραφή	Εισαγωγή /Εξαγωγή
perm(n)	Int	Περιέχει το διάνυσμα ομοιότητας μεγέθους n	I
nrhs	Int	Αριθμός δεξιών μελών	I
iparm(64)	Int	Παράμετροι ελέγχου επιλύτη	I/O
msglvl	Int	Επίπεδο στατιστικών πληροφοριών	I
	0	Ο PARDISO δεν δίνει καμία πληροφορία	
	1	Ο PARDISO τροφοδοτεί τον χρήστη με στατιστικά στοιχεία	
b(n*nrhs)	R/C	Διανύσματα δεξιών μελών	I/O
x(n*nrhs)	R/C	Διανύσματα λύσης	O
error	Int	Δείκτης σφαλμάτων	O

Πίνακας 3.2
Επισκόπηση των τιμών εισόδου της αράδας IPARM

Όνομα	Περιγραφή	
IPARM(1)	Χρήση προεπιλεγμένων τιμών	
	0	Θέτει σε όλες τις παραμέτρους, εκτός της IPARM(3), τις προεπιλεγμένες τιμές τους (τιμές με αστερίσκο)
IPARM(2)	Χρήση αναδιάταξης METIS	
	0	Δεν χρησιμοποιείται METIS
	2*	Χρήση αναδιάταξης METIS με αλγόριθμο nested dissection
IPARM(3)	Αριθμός επεξεργαστών	
	P	Αριθμός νημάτων OpenMP (μεταβλητή OMP_NUM_THREADS)
IPARM(5)	Χρήση πίνακα ομοιότητας καθορισμένου από τον χρήστη	
	0*	Δεν χρησιμοποιείται
	1	Χρήση που καθορίζεται στην μεταβλητή PERM
IPARM(6)	Λύση στο X / B	
	0*	Εγγραφή της λύσης στο X
	1	Εγγραφή της λύσης στο B
IPARM(8)	Μέγιστος αριθμός βημάτων επαναληπτικής βελτίωσης	
	k=0*	Εκτέλεση το πολύ k βημάτων αριθμητικής επαναληπτικής βελτίωσης για όλους τους πίνακες
IPARM(10)	eps pivot (perturbation 10^{-k})	
	13*	Προεπιλεγμένη τιμή για μη-συμμετρικούς πίνακες
	8*	Προεπιλεγμένη τιμή για συμμετρικούς, αόριστους πίνακες
IPARM(11)	Χρήση (μη-) συμμετρικών διανυσμάτων κλιμάκωσης	
	0	Δεν χρησιμοποιούνται
	1*	Χρησιμοποιούνται (μη-συμμετρικοί πίνακες)
	0*	Δεν χρησιμοποιούνται (συμμετρικοί πίνακες)
IPARM(12)	Λύση πίνακα μετατόπισης	
	0*	Εκτέλεση κανονικής επίλυσης
	1	Εκτέλεση επίλυσης με πίνακα μετατόπισης
IPARM(13)	Βελτιωμένη ακρίβεια με χρήση (μη-) συμμετρικών ταιριασμάτων	
	0	Δεν χρησιμοποιείται
	1*	Χρησιμοποιείται (μη-συμμετρικοί πίνακες)
	2	Χρήση ιδιαίτερα εύρωστης μεθόδου για συμμετρικούς αόριστους πίνακες
	0*	Δεν χρησιμοποιείται (συμμετρικοί πίνακες)

Όνομα	Περιγραφή	
IPARM(18)	Αριθμός μη-μηδενικών στοιχείων στην LU	
	0	Να μην καθοριστεί
	-1*	Θα καθοριστεί μόνο αν η τιμή της μεταβλητής είναι -1
IPARM(19)	MFLOPS για την παραγοντοποίηση LU	
	0*	Να μην καθοριστεί
	-1	Θα καθοριστεί μόνο αν η τιμή της μεταβλητής είναι -1 (αυξάνει τον χρόνο υπολογισμών)
IPARM(21)	Περιστροφές για συμμετρικούς αόριστους πίνακες	
	0	Διαγώνιες περιστροφές 1x1
	1*	Διαγώνιες περιστροφές 1x1 και 2x2 (Bunch-Kaufman)
IPARM(24)	Παράλληλη αριθμητική παραγοντοποίηση	
	0	Εκτέλεση παράλληλης χρονοδρομολόγησης ενός επιπέδου
	1*	Εκτέλεση παράλληλης χρονοδρομολόγησης δύο επιπέδων
IPARM(25)	Παράλληλη μπρος / πίσω επίλυση	
	0	Εκτέλεση σειριακής επίλυσης
	1*	Εκτέλεση παράλληλης επίλυσης
IPARM(26)	Μερική μπρος / πίσω επίλυση	
	0*	Εκτέλεση μπρος / πίσω επίλυσης με τους L και U
	1	Εκτέλεση επίλυσης προς τα εμπρός με τους L ή U^T
	2	Εκτέλεση προς τα πίσω επίλυσης με τους U ή L^T
IPARM(28)	Παράλληλη αναδιάταξη METIS	
	0*	Εκτέλεση σειριακής αναδιάταξης METIS
	1	Εκτέλεση παράλληλης αναδιάταξης METIS
IPARM(29)	Ακρίβεια 32bit / 64bit IEEE	
	0*	Χρήση ακρίβειας 64bit IEEE
	1	Χρήση ακρίβειας 32bit IEEE
IPARM(30)	Control size of supernodes	
	0*	Χρήση προεπιλεγμένης διαμόρφωσης
	1	Χρήση διαμόρφωσης χρήστη
IPARM(31)	Μερική επίλυση για αραιά δεξιά μέλη και αραιές λύσεις	
	0*	Υπολογισμός όλων των μερών στο διάνυσμα λύσης
	1	Υπολογισμός μερικών επιλεγμένων μελών στο διάνυσμα λύσης
IPARM(32)	Χρήση του αναδρομικού, επαναληπτικού, γραμμικού επιλύτη	
	0*	Χρήση του αραιού, άμεσου επιλύτη
	1	Χρήση του αναδρομικού, επαναληπτικού επιλύτη
IPARM(60)	Διαμόρφωση του PARDISO	
	0*	In-Core
	2	Out-Of-Core : Χρησιμοποιείται για επίλυση πολύ μεγάλων συστημάτων, όπου η RAM δεν επαρκεί για την αποθήκευση των απαραίτητων πινάκων και πραγματοποιείται χρήση του σκληρού δίσκου ως μνήμη.

Πίνακας 3.3 Επισκόπηση των τιμών εξόδου της αράδας IPARM		
Όνομα	Περιγραφή	
IPARM(7)	Αριθμός βημάτων επαναληπτικής βελτίωσης	
IPARM(15)	Μέγιστη κατανάλωση μνήμης σε KB κατά τη διάρκεια της ανάλυσης και της ελαχιστοποίησης φαινομένων fill-in	
IPARM(16)	Μόνιμη κατανάλωση μνήμης σε KB που απαιτείται από τον επιλύτη για να πραγματοποιήσει τις φάσεις παραγοντοποίησης και επίλυσης	
IPARM(17)	Μέγιστη κατανάλωση μνήμης σε KB κατά την φάση της παραγοντοποίησης και επίλυσης	
IPARM(18)	Αριθμός μη-μηδενικών στοιχείων στην LU	
	0 -1*	Να μην καθοριστεί Θα καθοριστεί μόνο αν η τιμή της μεταβλητής είναι -1
IPARM(19)	MFLOPS για την παραγοντοποίηση LU	
	0* -1	Να μην καθοριστεί Θα καθοριστεί μόνο αν η τιμή της μεταβλητής είναι -1 (αυξάνει τον χρόνο υπολογισμών)
IPARM(20)	Αριθμός επαναλήψεων CG	
IPARM(22)	Αριθμός θετικών ιδιοτιμών για συμμετρικούς, αόριστους πίνακες	
IPARM(23)	Αριθμός αρνητικών ιδιοτιμών για συμμετρικούς, αόριστους πίνακες	

Πίνακας 3.4 Κωδικοί σφαλμάτων του PARDISO	
Σφάλμα	Περιγραφή
0	Κανένα σφάλμα
-1	Ασυνεπή δεδομένα
-2	Μη επαρκής μνήμη
-3	Πρόβλημα αναδιάταξης
-4	Πρόβλημα στην αριθμητική παραγοντοποίηση ή στην επαναληπτική βελτίωση
-5	Μη κατηγοριοποιημένο εσωτερικό σφάλμα
-6	Σφάλμα προδιάταξης (μόνο σε πίνακες τύπου 11 και 13)
-7	Πρόβλημα διαγώνιου πίνακα
-8	Πρόβλημα υπερχείλισης integer 32bit

3.3 Παράρτημα

Ακολουθεί ο κώδικας σε C++ που χρησιμοποιήθηκε για την εκτέλεση του PARDISO :

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

#pragma comment(lib, "libguide.lib")
#pragma comment(lib, "mkl_lapack.lib")
#pragma comment(lib, "mkl_solver.lib")

/* PARDISO prototype. */

#if defined(_WIN32) || defined(_WIN64)
#define pardiso_ PARDISO
#else
#define PARDISO pardiso_
#endif
#if defined(MKL_ILP64)
#define int long long
#else
#define int int
#endif

int compare (const void * a, const void * b)
{
  return ( *(int*)a - *(int*)b );
}

extern int PARDISO(void *, int *, int *, int *, int *, int *,double
                   *, int *, int *, int *, int *, int *,int *, double
                   *, double *, int *);

int main( void ) {

      FILE* ForInput;
      FILE* Initial_Solution;
      FILE* Status;
      FILE* myFile;
      FILE* dataElementsFile;
      FILE* dataFile;
      FILE* jacFile;
      FILE* ijacFile;
      FILE* jjacFile;
      FILE* bFile;

      int size1 = 0;
      int size2 = 0;

      double *jac; // Here is the A matrix in a system Ax = b
      int *ijac;
      int *jjac;
      double *b; // Here is the b array
      double *u;
      int *temp;
```

```
	//-----Input for the code to start-------//

	int nex = 175; //Number of elements in the x-direction
	int ney = 175; //Number of elements in the y-direction

	int np = 0;
	int i = 0;
	int j = 0;
	int count = 0;
	double Re = 1;

	void *pt[64];

	/* Pardiso control parameters. */

	int iparm[64];
	int mtype = 11; // Matrix type.
	int nrhs = 1; // Number of right hand sides.
	int phase = 13;
	int maxfct = 1; // Maximum number of numerical factorizations.
	int mnum = 1; // Which factorization to use.
	int msglvl = 1; // Print statistical information.
	int error = 0; // Initialize error flag.

	/* Auxiliary variables. */

	double ddum; // Double dummy.
	int idum; // Integer dummy.

	fopen_s(&Status,"status.dat","w");
	fprintf_s(Status,"%d\n",0);

	fclose(Status);

	fopen_s(&ForInput,"Input.dat","w");
	fprintf_s(ForInput,"%d\n",nex);
	fprintf_s(ForInput,"%d\n",ney);
	fprintf_s(ForInput,"%lf\n",Re);

	//---------------------------------------//

	np = (nex+1)*(ney+1) + 2*(2*nex+1)*(2*ney+1); // Total nummber
of degrees of freedom.

	u = malloc(np*sizeof(double));

	for (i=0; i<np; i++)
		u[i] = 0;

	fopen_s(&Initial_Solution,"u.dat","w");

	for (i=0; i<np; i++)
		fprintf_s(Initial_Solution,"%lf\n",u[i]);

	fclose(ForInput);
	fclose(Initial_Solution);

	system("RPM2_COMPAQ.exe");

	fopen_s(&dataElementsFile,"input.dat","r");
	fopen_s(&dataFile,"size_me.dat","r");
```

```
        fopen_s(&jacFile,"jac.dat","rb");
        fopen_s(&ijacFile,"ijac.dat","rb");
        fopen_s(&jjacFile,"jjac.dat","rb");
        fopen_s(&bFile,"b.dat","rb");

        fscanf_s(dataFile,"%d %d\n",&size1,&size2);

        jac = malloc(size1*sizeof(double));
        ijac = malloc((np+1)*sizeof(int));
        jjac = malloc(size1*sizeof(int));
        b = malloc(np*sizeof(double));

        fread(jac,sizeof(double),size1,jacFile);
        fread(ijac,sizeof(int),(np+1),ijacFile);
        fread(jjac,sizeof(int),size1,jjacFile);
        fread(b,sizeof(double),np,bFile);

        fclose(dataFile);
        fclose(jacFile);
        fclose(ijacFile);
        fclose(jjacFile);
        fclose(dataElementsFile);
        fclose(bFile);

        temp = (int*)malloc((ijac[1]-ijac[0])*sizeof(int));

        i = 0;
        for (j=0; j<(ijac[i+1]-ijac[i]); j++)
              temp[j] = 0;

        for (i=0; i<np; i++)
        {
              temp = realloc(temp,(ijac[i+1]-ijac[i])* sizeof(int));

              for (j=0; j<(ijac[i+1]-ijac[i]); j++)
                    temp[j] = jjac[ijac[i]+j-1];

              qsort(temp, (ijac[i+1]-ijac[i]), sizeof(int), compare);

              for (j=0; j<ijac[i+1]-ijac[i]; j++)
                    jjac[j+ijac[i]-1] = temp[j];
        }

        ijac[np] = size1 + 1;

        for (i=0; i<size1; i++)
              printf("%d %d\n",i, jjac[i]);

/* ---------------------------------------------------------------------- */
/* Setup Pardiso control parameters. */
/* ---------------------------------------------------------------------- */

        for (i = 1; i < 64; i++) {
              iparm[i] = 0;
        }
        iparm[1] = 0; // 0 -> Solver default parameters
        iparm[2] = 2; // Fill-in reordering algorithm
        iparm[3] = 4; // Number of CPUs
        iparm[4] = 0; // No iterative-direct algorithm
        iparm[5] = 0; // No user fill-in reducing permutation
        iparm[6] = 0; // Write solution into x
```

```
	iparm[7] = 0; // Output : Number of iterative refinement steps
	iparm[8] = 2; // Max numbers of iterative refinement steps
	iparm[9] = 0; // Not in use
	iparm[10] = 13; // Perturb the pivot elements with 1E-13
	iparm[11] = 1; // Use nonsymmetric permutation and scaling MPS
	iparm[12] = 0; // Not in use
	iparm[13] = 1; // Maximum weighted matching algorithm
	iparm[14] = 0; // Output : Number of perturbed pivots
	iparm[15] = 0; // Output : Peak memory in KB during analysis
	iparm[16] = 0; // Output : Permanent memory in KB
	iparm[17] = 0; // Output : Peak double precision memory in KB
	iparm[18] = -1; // Output : Number of nonzeros in the factor LU
	iparm[19] = -1; // Output : Mflops for LU factorization
	iparm[20] = 0; // Output : Numbers of CG Iterations
	iparm[27] = 1; // Input Checker
	iparm[60] = 0; // 0 -> In-Core mode

/* ---------------------------------------------------------------- */
/* Initialize the internal solver memory pointer. This is only */
/* necessary for the FIRST call of the PARDISO solver. */
/* ---------------------------------------------------------------- */

	for (i = 0; i < 64; i++) {
		pt[i] = 0;
	}

/* ---------------------------------------------------------------- */
/* Reordering and Symbolic Factorization. This step also allocates */
/* all memory that is necessary for the factorization. */
/* ---------------------------------------------------------------- */

	PARDISO (pt, &maxfct, &mnum, &mtype, &phase,
		&np, jac, ijac, jjac, &idum, &nrhs,
		iparm, &msglvl, &b, &u, &error);
	if (error != 0) {
		printf("\nERROR during symbolic factorization: %d",
error);
		exit(1);
	}
	printf("\nReordering completed ... ");
	printf("\nPeak memory in KB during analysis = %d", iparm[15]);
	printf("\nPermanent memory in KB = %d", iparm[16]);
	printf("\nPeak double precision memory in KB = %d", iparm[17]);
	printf("\nNumber of nonzeros in factors = %d", iparm[18]);
	printf("\nNumber of factorization MFLOPS = %d", iparm[19]);

/* ---------------------------------------------------------------- */
/* Termination and release of memory. */
/* ---------------------------------------------------------------- */

//	phase = -1; /* Release internal memory. */
//	PARDISO (pt, &maxfct, &mnum, &mtype, &phase,
//		&np, &jac, ijac, jjac, &idum, &nrhs,
//		iparm, &msglvl, &b, &u, &error);

	//-------This must be put at the end of the code--------//

	free(jac);
	free(ijac);
	free(jjac);
	free(b);
```

```
        free(u);
        free(temp);

        //-------------------------------------------------------//

        system ("pause");

        return 0;

}
```

Κεφάλαιο 4ο

4.1 Πλατφόρμες δοκιμών του PARDISO

Για την διεξαγωγή των δοκιμών στον PARDISO χρησιμοποιήθηκαν δύο τετραπύρηνες πλατφόρμες με τα εξής χαρακτηριστικά :

Πίνακας 4.1
Τεχνικά χαρακτηριστικά συστημάτων δοκιμών

Επεξεργαστής		
Κατασκευαστής	Intel	AMD
Ονομασία	Core 2 Quad Q6600	Phenom X4 9850 BE
Κωδική ονομασία chip	Kentsfield	Agena
Αριθμός πυρήνων	2x2	4
Αριθμός νημάτων (threads)	4	4
Συχνότητα λειτουργίας	2,40GHz	2,50GHz (*)
L1 Cache	2x128KB / πυρήνα	2x64KB / πυρήνα
L2 Cache	4MB / 2 πυρήνες	512KB / πυρήνα
L3 Cache	-	2MB κοινόχρηστη
Σετ εντολών	64bit	64bit
Λιθογραφία	65nm	65nm
Πλατφόρμα	LGA 775	AM2+
Chipset		
Κατασκευαστής	Intel	AMD
Ονομασία	G31	790FX
Ταχύτητα διαύλου	1066MHz	2000MHz
Διαθέσιμα κανάλια μνήμης	2	2
Μνήμη		
Πρότυπο	DDR2	DDR2
Συχνότητα λειτουργίας	800MHz	800MHz
Ποσότητα μνήμης	2x1GB+2x2GB	2x1GB+2x2GB

* Η παρουσία ξεκλείδωτου πολλαπλασιαστή (πρόθεμα BE στην ονομασία του) στον συγκεκριμένο επεξεργαστή, έδωσε την δυνατότητα εύκολου υπερχρονισμού του και εκτέλεση δοκιμών σε συχνότητα λειτουργίας 3,0GHz.

4.1.1 Ομοιότητες των συστημάτων δοκιμών

Για τους παραπάνω υπολογιστές, οι οποίοι δουλεύουν στο ίδιο λειτουργικό σύστημα (Windows 7 64bit), είναι καλό να σημειωθούν οι παρακάτω ομοιότητες :

- Βασίζονται και οι δύο σε τετραπύρηνες υλοποιήσεις οπότε δίνεται η δυνατότητα πιο άμεσης σύγκρισης της κλιμάκωσης των επιδόσεων των δύο

επεξεργαστών, καθώς ο PARDISO θα λειτουργεί με έναν, δύο ή τέσσερις επεξεργαστές.

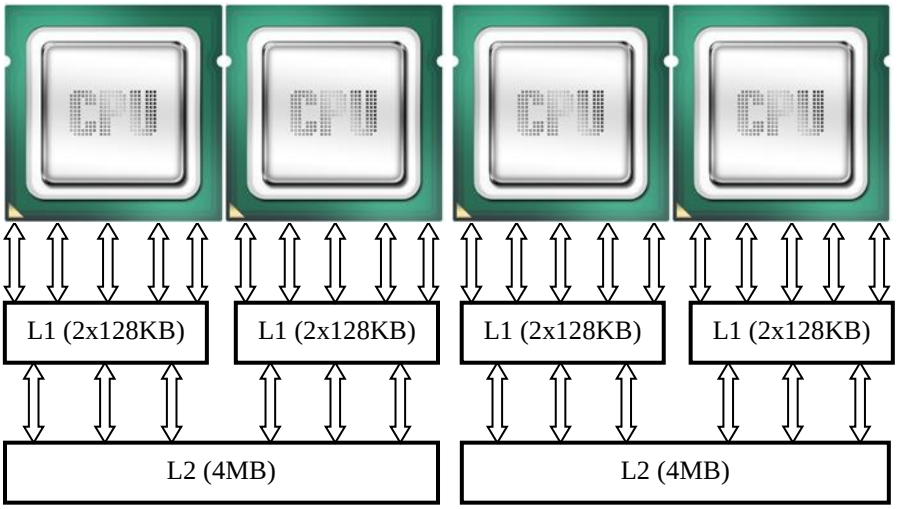
- Διαθέτουν ακριβώς τα ίδια χαρακτηριστικά μνήμης (ποσό, συχνότητα λειτουργίας και αριθμός καναλιών), οπότε οι διαφορές στον χρόνο επίλυσης ενός προβλήματος θα οφείλονται στην αρχιτεκτονική του εκάστοτε επεξεργαστή.
- Η ταχύτητα διαύλου και των δύο μηχανημάτων είναι μεγαλύτερη από την συχνότητα λειτουργίας της μνήμης γεγονός το οποίο, σε συνδυασμό με την ύπαρξη διπλού καναλιού επικοινωνίας της μνήμης με τον ελεγκτή, εξασφαλίζει την ελαχιστοποίηση πιθανής συμφόρησης του διαύλου όταν οι επεξεργαστές ζητούν να προσπελάσουν την διαμοιραζόμενη μνήμη.

4.1.2 Διαφορές των συστημάτων δοκιμών

Η βασική διαφορά, η οποία θα προσφέρει ένα πολύ καλό μέτρο σύγκρισης της συμπεριφοράς του PARDISO στα δύο chip, έγκειται στην αρχιτεκτονική που ακολούθησαν οι δύο κατασκευάστριες (AMD και Intel) για την παραγωγή τους.

Στην περίπτωση της σειράς Core 2 Quad, η Intel επέλεξε να μην σχεδιάσει ένα νέο chip με τέσσερις πυρήνες αλλά να τοποθετήσει στην ίδια πλατφόρμα δύο διπύρηνα chip της σειράς Core 2 (Σχήμα 4.1). Έτσι, οι επεξεργαστές της σειράς Core 2 Quad διαθέτουν μόνο δύο επίπεδα κρυφής μνήμης. Το πρώτο επίπεδο αποτελεί την προσωπική κρυφή μνήμη του κάθε πυρήνα, ενώ το δεύτερο μοιράζεται μεταξύ των δύο επεξεργαστών που βρίσκονται στο ίδιο chip (Σχήμα 4.2).

Σχήμα 4.1

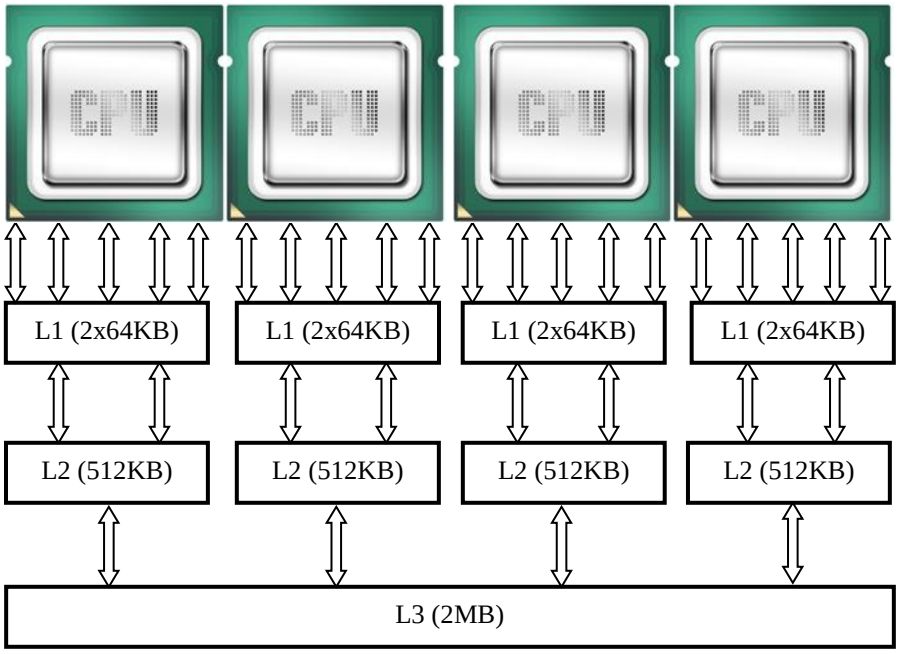


Σχήμα 4.2
Αρχιτεκτονική επεξεργαστή της
σειράς Core 2 Quad

Κατά αυτόν τον τρόπο η Intel κατάφερε να βγάλει γρήγορα στην αγορά έναν τετραπύρηνο επεξεργαστή, κρατώντας ταυτόχρονα και το κόστος ανάπτυξής του σε χαμηλά επίπεδα. Με την αρχιτεκτονική αυτή όμως, όπως αναφέρεται και στην ενότητα 1.2.2, απαιτείται συνεχής παρακολούθηση του διαύλου από τις δύο L2 Cache για να αποφευχθούν προβλήματα συνοχής της κρυφής μνήμης. Αυτό μπορεί να αποβεί χρονοβόρο σε περιπτώσεις που το υπολογιστικό φορτίο και των τεσσάρων επεξεργαστών ανέβει σε ψηλά επίπεδα.

Για την σειρά Phenom X4, η AMD επέλεξε να βασιστεί στην αρχιτεκτονική των επεξεργαστών της σειράς Opteron. Οι Opteron αποτελούσαν πολυπύρηνες λύσεις για διακομιστές με ένα chip (single chip servers). Έτσι, ο Phenom X4 αποτελεί «καθαρόαιμο» τετραπύρηνο επεξεργαστή, αφού όλοι του οι πυρήνες αποτελούν μέρος του ίδιου chip (Σχήμα 4.3). Η επιλογή της παραπάνω αρχιτεκτονικής από την AMD, έδωσε την δυνατότητα ο κάθε πυρήνας του Phenom να διαθέτει δύο επίπεδα προσωπικής κρυφής μνήμης, ενώ όλοι οι πυρήνες μοιράζονται το τρίτο επίπεδο (Σχήμα 4.4).

Σχήμα 4.3



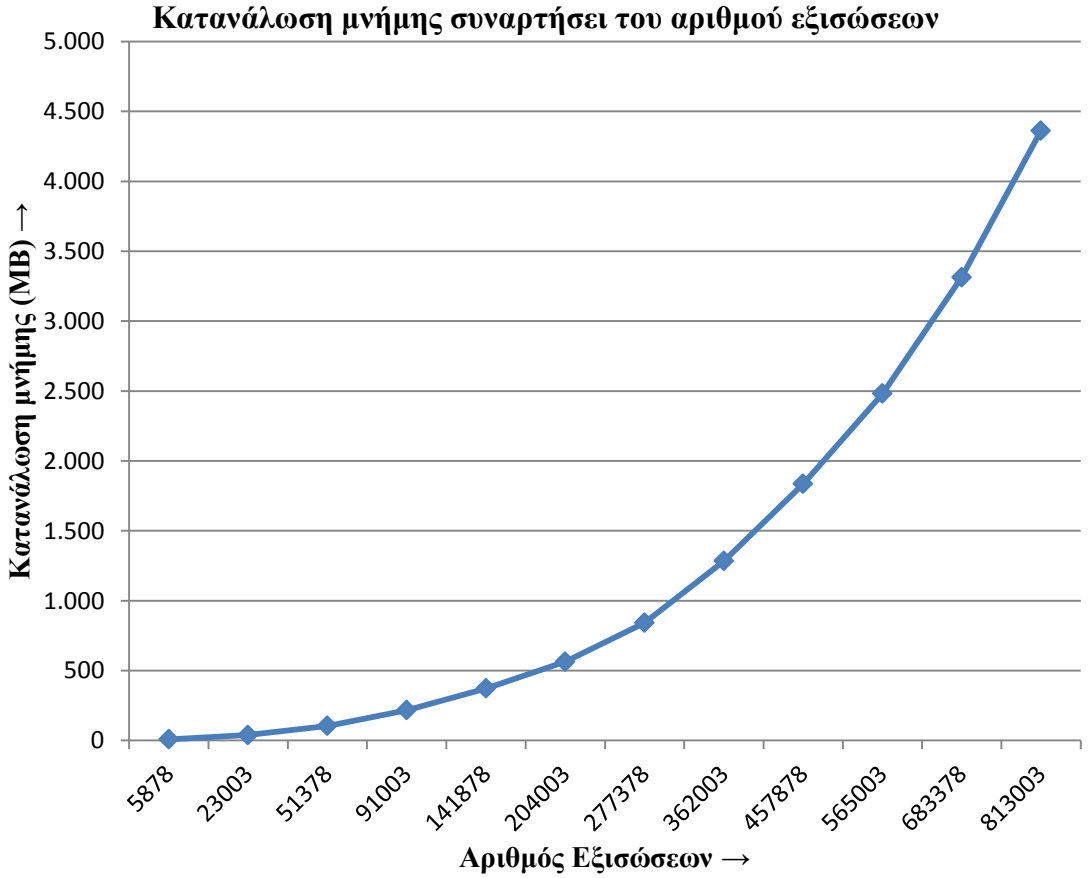
Σχήμα 4.4
Αρχιτεκτονική επεξεργαστή της
σειράς Phenom X4

Για να κρατήσει όμως η AMD χαμηλά το κόστος παραγωγής ενός chip αρκετά πιο πολύπλοκου από αυτό της Intel, αναγκάστηκε να εξοπλίσει τους Phenom με την μισή συνολική κρυφή μνήμη σε σχέση με τους Core 2 Quad, γεγονός που δίνει στο chip της Intel την δυνατότητα να αποθηκεύει πολύ περισσότερες μεταβλητές από την διαμοιραζόμενη κεντρική μνήμη του συστήματος.

4.2 Συνθήκες δοκιμών του PARDISO

Για τον έλεγχο της συμπεριφοράς του PARDISO στα παραπάνω υπολογιστικά συστήματα, πραγματοποιήθηκε η αριθμητική παραγοντοποίηση και επίλυση δώδεκα προβλημάτων που δημιουργούν συστήματα με 5878 έως 813003 εξισώσεις. Για τα προαναφερθέντα μεγέθη προβλημάτων, οι απαιτήσεις του PARDISO σε μνήμη μπορούσαν να καλυφθούν από την διαμοιραζόμενη μνήμη των υπολογιστών που χρησιμοποιήθηκαν (Διάγραμμα 4.1), ενώ θα πρέπει να αναφερθεί ότι στην περίπτωση του μεγαλύτερου προβλήματος που δοκιμάστηκε, το λειτουργικό σύστημα «αναγκάστηκε» να ελευθερώσει μνήμη για να καλυφθεί ο επιλύτης. Σε μεγαλύτερα προβλήματα, το λειτουργικό σύστημα δεν μπορούσε να ελευθερώσει επιπλέον μνήμη και ο επιλύτης έπρεπε να χρησιμοποιήσει τον σκληρό δίσκο ως εικονική μνήμη (swap/page file). Στην περίπτωση αυτή, οι επιδόσεις και των δύο συστημάτων έπεφταν κατακόρυφα, καθιστώντας ανούσια την εκτέλεση οποιασδήποτε δοκιμής.

Διάγραμμα 4.1



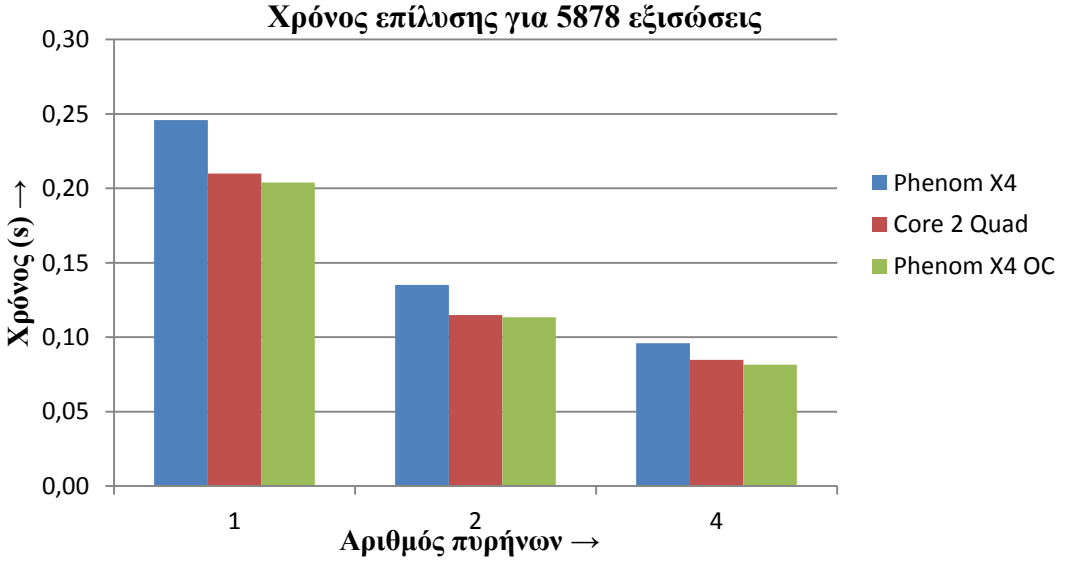
Τα χαρακτηριστικά των προβλημάτων και οι ακριβείς απαιτήσεις τους σε μνήμη, όπως αυτές αναφέρθηκαν στα στατιστικά του PARDISO, φαίνονται στο παράρτημα του κεφαλαίου.

4.3 Παρουσίαση και συζήτηση αποτελεσμάτων

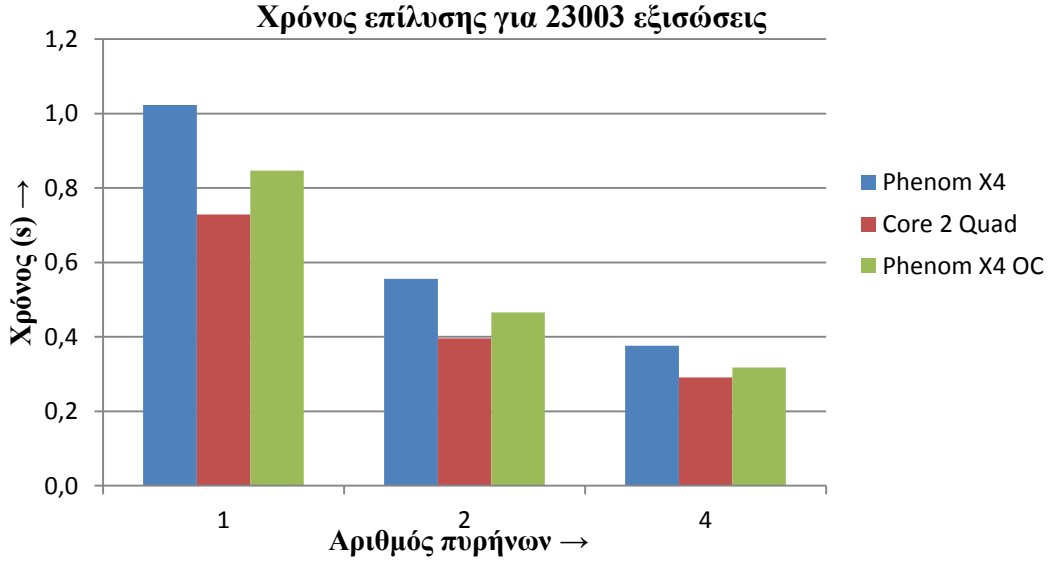
4.3.1 Παρουσίαση αποτελεσμάτων

Στα διαγράμματα που ακολουθούν, παρουσιάζονται οι χρόνοι αριθμητικής παραγοντοποίησης και επίλυσης των δώδεκα προβλημάτων χρησιμοποιώντας έναν, δύο και τέσσερις πυρήνες στο κάθε chip που δοκιμάστηκε. Επιπλέον, παρουσιάζονται τα αποτελέσματα δοκιμών του Phenom X4 σε υπερχρονισμένη διαμόρφωση στα 3,00GHz (Phenom X4 OC). Οι ακριβείς χρόνοι εκτέλεσης του PARDISO φαίνονται αναλυτικά στο παράρτημα του κεφαλαίου.

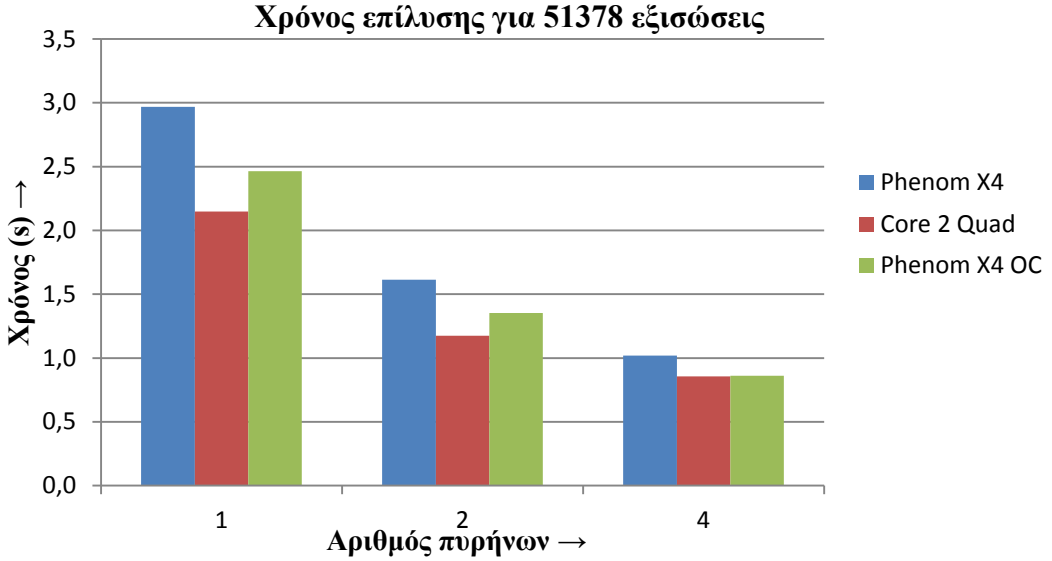
Διάγραμμα 4.2
Χρόνος επίλυσης για 5878 εξισώσεις



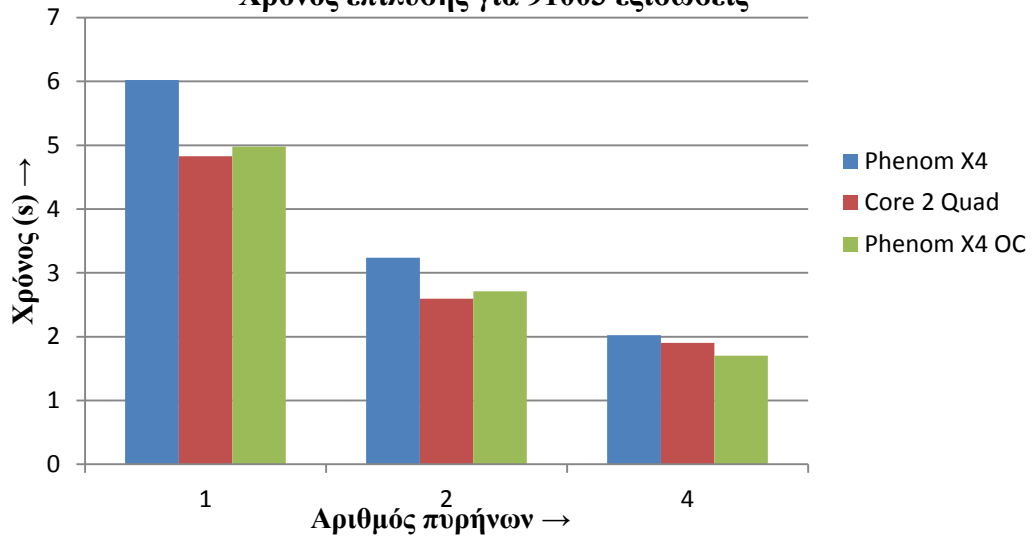
Διάγραμμα 4.3
Χρόνος επίλυσης για 23003 εξισώσεις



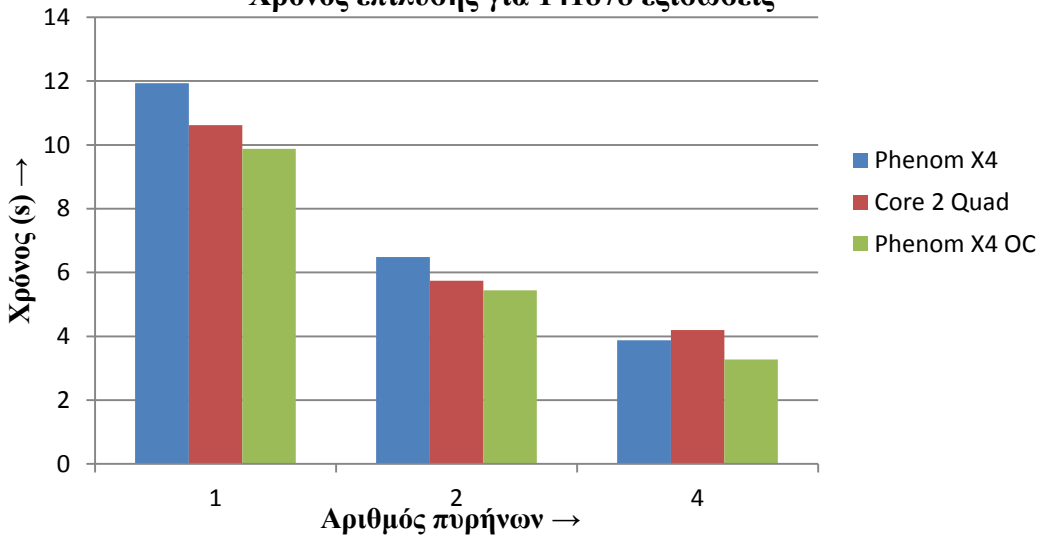
Διάγραμμα 4.4
Χρόνος επίλυσης για 51378 εξισώσεις



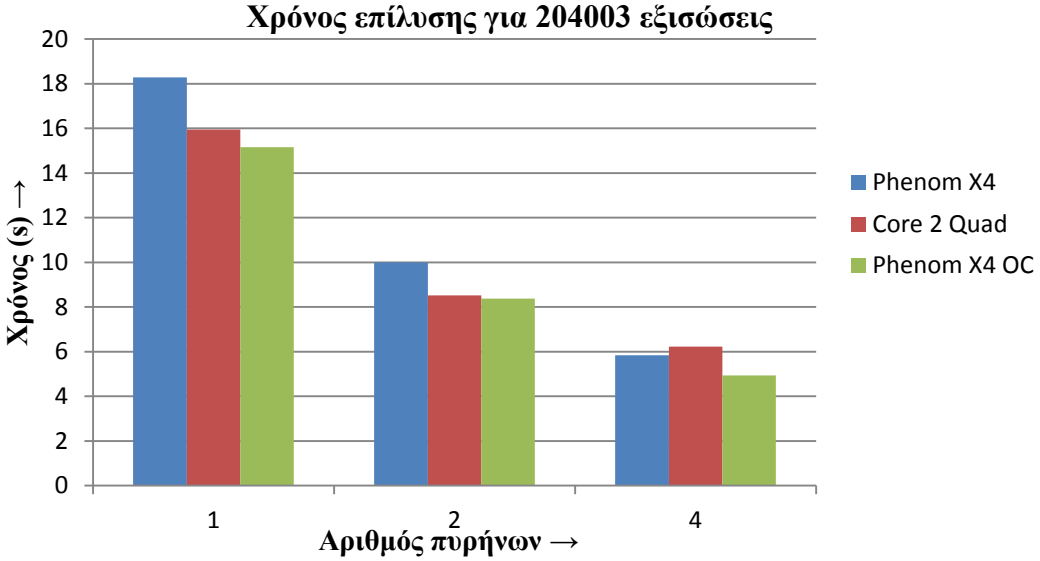
Διάγραμμα 4.5
Χρόνος επίλυσης για 91003 εξισώσεις



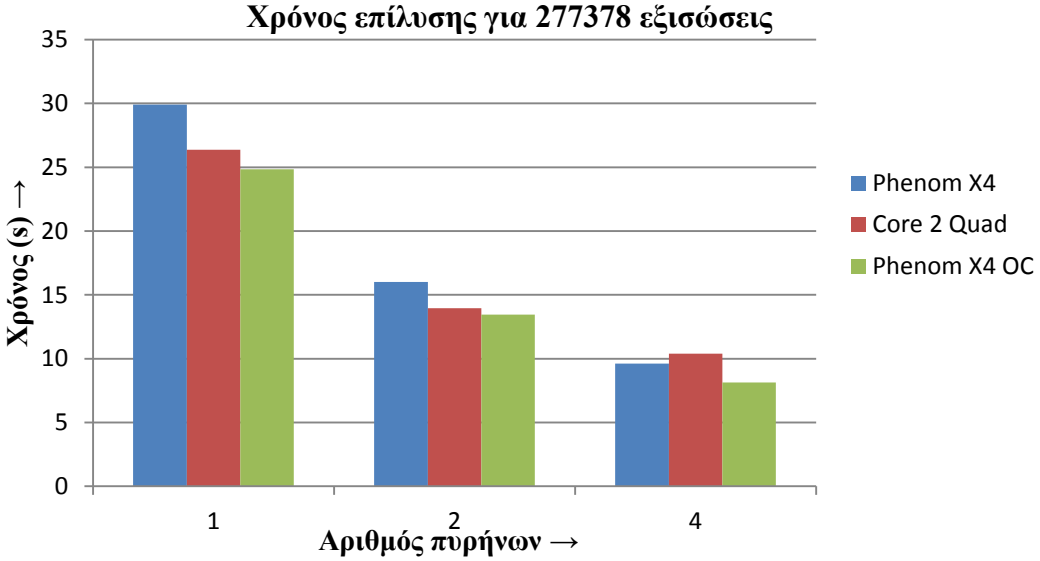
Διάγραμμα 4.6
Χρόνος επίλυσης για 141878 εξισώσεις



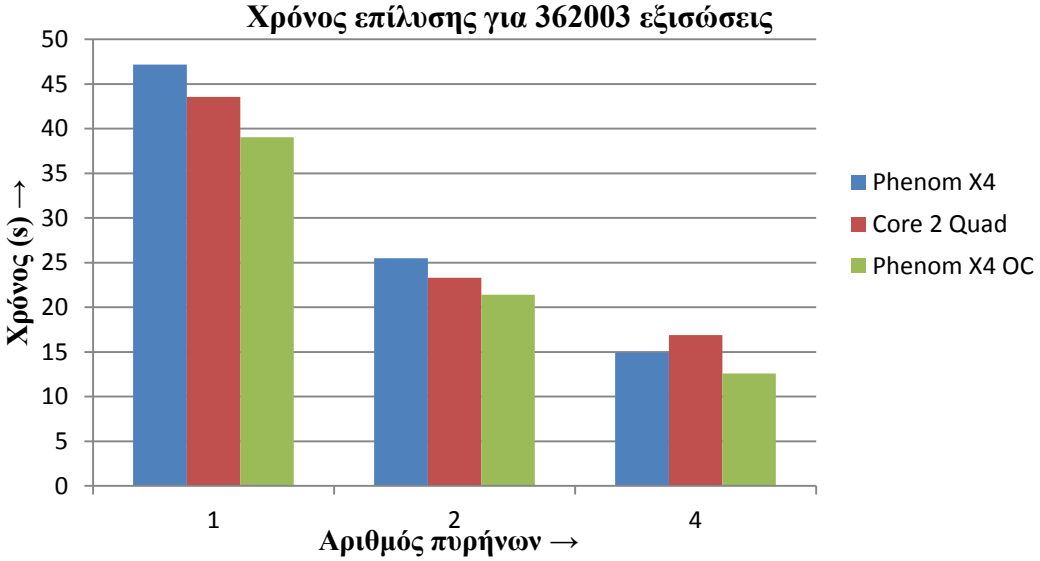
Διάγραμμα 4.7
Χρόνος επίλυσης για 204003 εξισώσεις



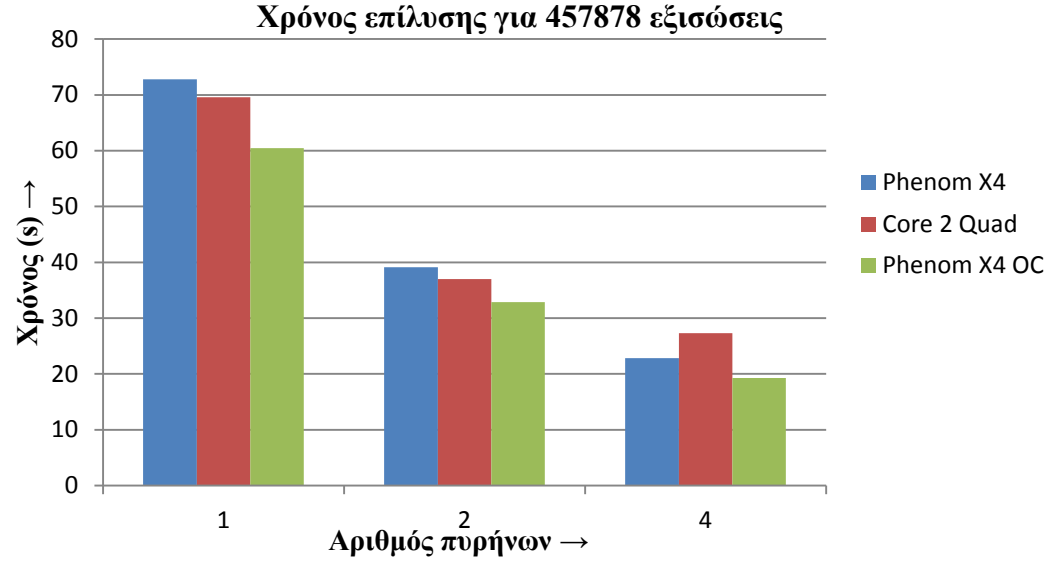
Διάγραμμα 4.8
Χρόνος επίλυσης για 277378 εξισώσεις



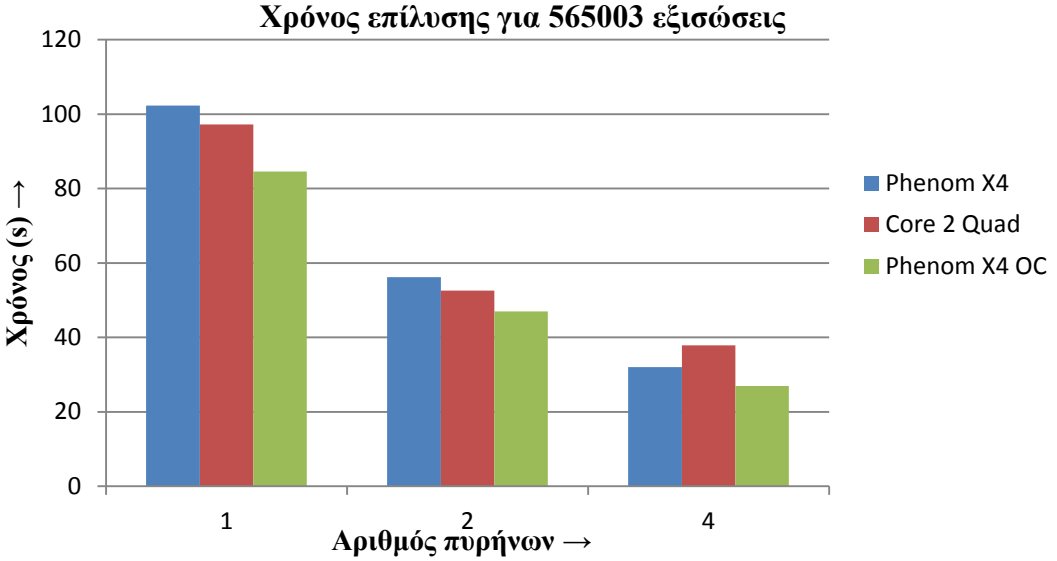
Διάγραμμα 4.9
Χρόνος επίλυσης για 362003 εξισώσεις



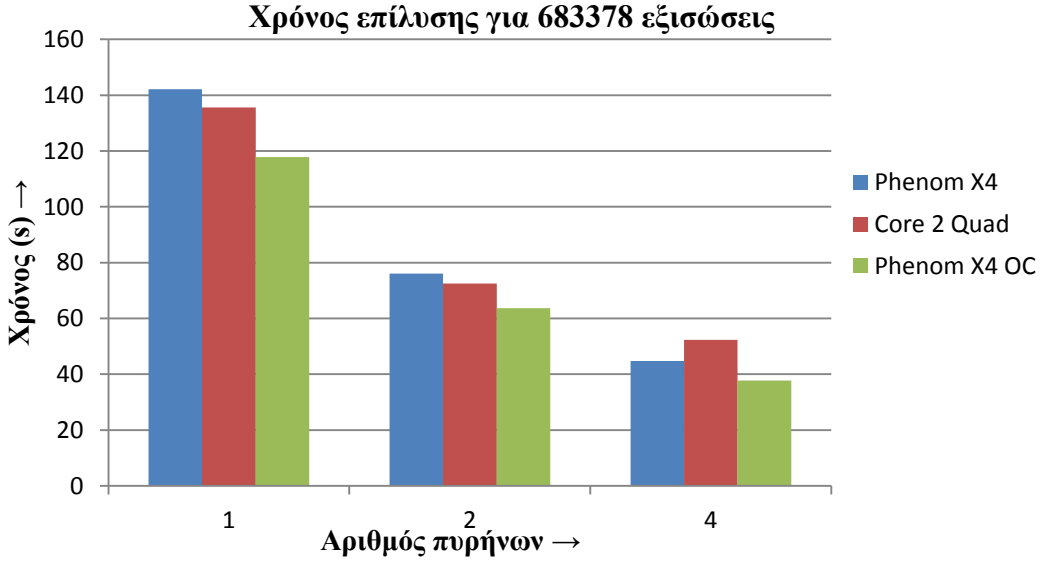
Διάγραμμα 4.10
Χρόνος επίλυσης για 457878 εξισώσεις



Διάγραμμα 4.11
Χρόνος επίλυσης για 565003 εξισώσεις

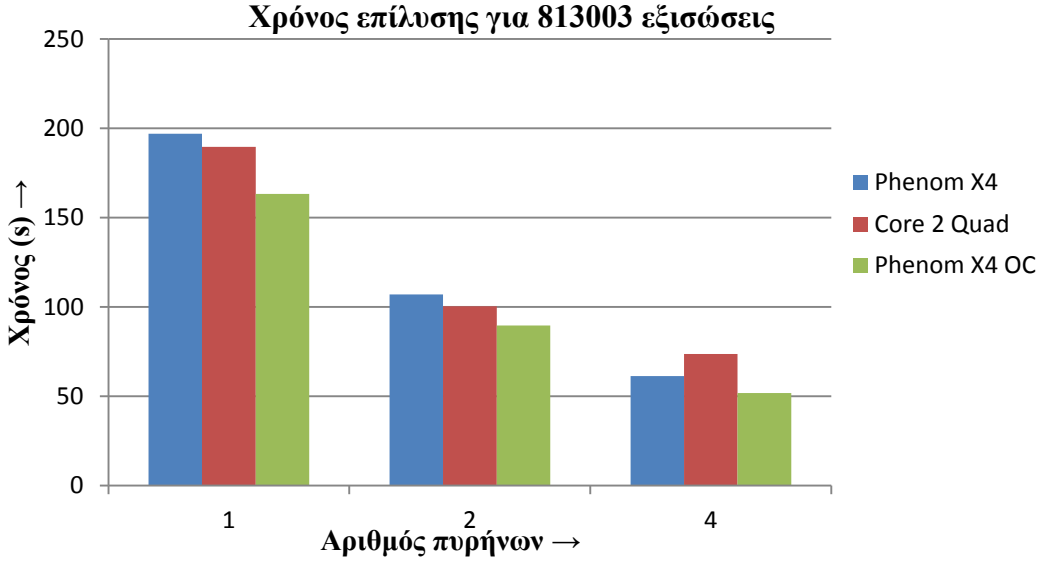


Διάγραμμα 4.12
Χρόνος επίλυσης για 683378 εξισώσεις



Διάγραμμα 4.13

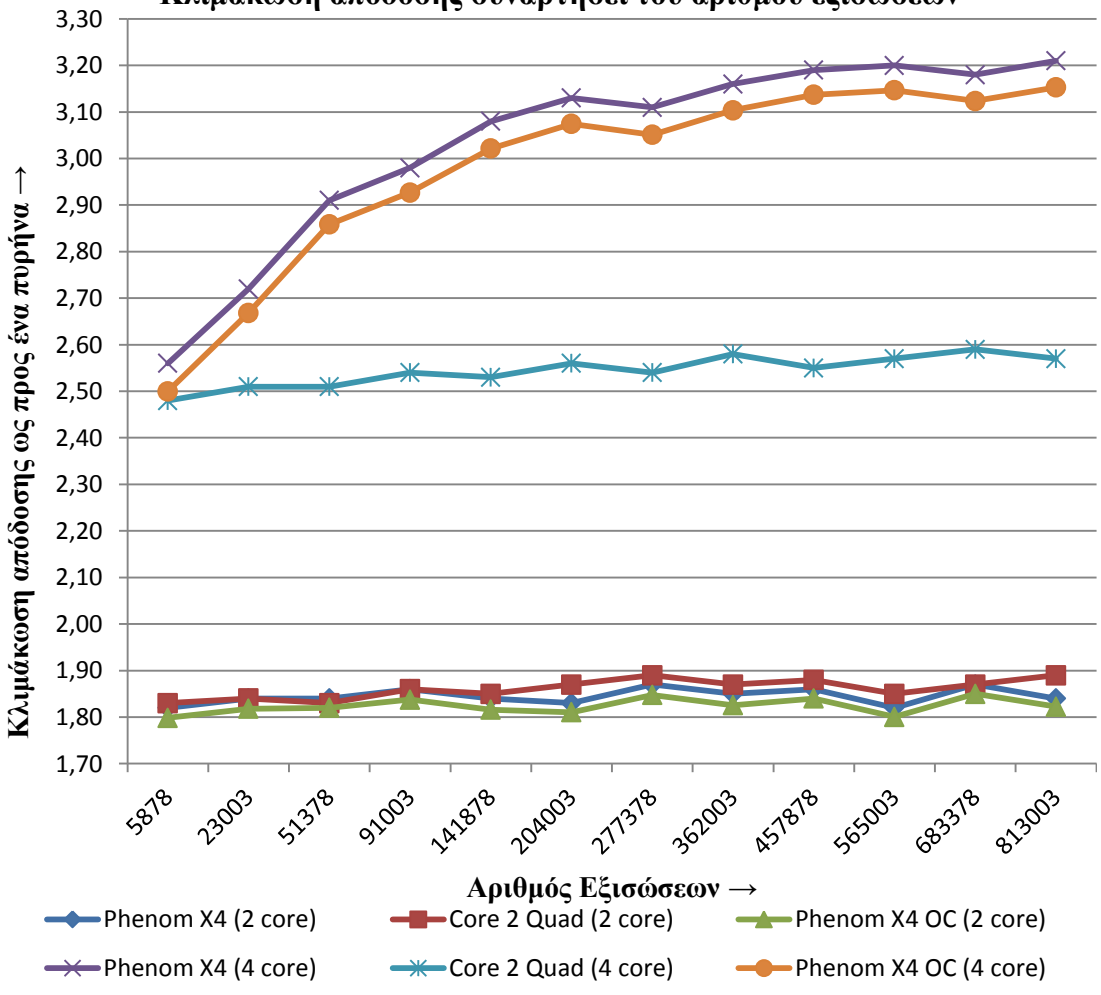
Χρόνος επίλυσης για 813003 εξισώσεις



Στο διάγραμμα που ακολουθεί, φαίνεται η κλιμάκωση του συντελεστή αύξησης των επιδόσεων του κάθε chip σε συνάρτηση με το μέγεθος του προβλήματος, καθώς ενεργοποιούνται διαδοχικά δύο και τέσσερις πυρήνες :

Διάγραμμα 4.14

Κλιμάκωση απόδοσης συναρτήσει του αριθμού εξισώσεων



4.3.2 Κλιμάκωση του χρόνου εκτέλεσης του PARDISO στην συνήθη συχνότητα λειτουργίας των επεξεργαστών

4.3.2.1 Υπολογισμοί σε ένα πυρήνα

Στην πολύ απλή περίπτωση όπου ο PARDISO εκτελείται σε ένα μόνο πυρήνα, ο επιλύτης δείχνει να συμπεριφέρεται καλύτερα στην αρχιτεκτονική του Core 2 Quad. Πιο συγκεκριμένα, τα 4,25MB (L1 + L2) κρυφής μνήμης στα οποία έχει πρόσβαση ο ένας πυρήνας του Core 2 Quad σε σύγκριση με τα 2,5MB του Phenom (L2 + L3), δίνουν την δυνατότητα στο chip της Intel να ολοκληρώσει τους υπολογισμούς του σε μικρότερο χρόνο. Μια τέτοια συμπεριφορά είναι αναμενόμενη, αφού η σαφώς μεγαλύτερη cache στον Core 2 δίνει την δυνατότητα στον επεξεργαστή να αποθηκεύσει μεγαλύτερο μέρος από τις πιο συχνά χρησιμοποιούμενες μεταβλητές του προς επίλυση προβλήματος, γεγονός το οποίο προσφέρει αυξημένο ποσοστό ευστοχιών της κρυφής μνήμης και ελαχιστοποιεί τα φαινόμενα ανταγωνισμού στην διαμοιραζόμενη κεντρική μνήμη.

4.3.2.2 Υπολογισμοί σε δύο πυρήνες

Καθώς ο αριθμός των εν χρήσει πυρήνων διπλασιάζεται, παρατηρούμε ότι ο PARDISO εξακολουθεί να επωφελείται της αρχιτεκτονικής της Intel συγκριτικά με αυτήν της AMD στην συνήθη συχνότητα λειτουργίας της. Όπως ακριβώς και στην περίπτωση των υπολογισμών σε ένα πυρήνα, η μεγαλύτερη κρυφή μνήμη που μοιράζονται ανά δύο οι πυρήνες του Core 2 Quad δίνει την δυνατότητα στους επεξεργαστές να αποθηκεύουν περισσότερες από τις πιο συχνά χρησιμοποιημένες μεταβλητές σε σχέση με τους δύο πυρήνες του Phenom, προσφέροντας μικρότερους χρόνους εκτέλεσης του επιλύτη σε όλα τα προβλήματα.

Η κλιμάκωση του χρόνου εκτέλεσης του PARDISO καθώς διπλασιάζεται ο αριθμός των πυρήνων είναι και στις δύο περιπτώσεις εξαιρετική, αφού παρατηρείται αύξηση της απόδοσης με συντελεστή από 1,82 μέχρι 1,89 ανάλογα με το μέγεθος του προβλήματος. Η παρατηρούμενη αύξηση της απόδοσης είναι πολύ κοντά στο θεωρητικό μέγιστο (x2) που μπορούμε να πετύχουμε με τον διπλασιασμό των πυρήνων σε περίπτωση που δεν παρατηρείται ανταγωνισμός στην μνήμη, γεγονός που δείχνει ότι ο PARDISO «σπάει» σε δύο παράλληλες διεργασίες ιδιαίτερα αποδοτικά ακόμα και σε αρχιτεκτονικές επεξεργαστών που διαφέρουν αρκετά μεταξύ τους.

4.3.2.3 Υπολογισμοί σε τέσσερις πυρήνες

Με την ενεργοποίηση όλων των διαθέσιμων πυρήνων στους επεξεργαστές η συμπεριφορά του PARDISO αλλάζει αρκετά και οι διαφορές στην αρχιτεκτονική των δύο συστημάτων φαίνονται πιο ξεκάθαρα ιδιαίτερα όσο το μέγεθος του προβλήματος αυξάνεται.

Σε μικρά προβλήματα και οι δύο αρχιτεκτονικές ξεκινούν με συντελεστή αύξησης απόδοσης κοντά στο 2,5. Καθώς το μέγεθος των προβλημάτων μεγαλώνει, ο συντελεστής αύξησης της απόδοσης του Core 2 Quad σταθεροποιείται σε τιμές μεταξύ 2,5 και 2,6 ενώ του Phenom X4 ακολουθεί ανοδική πορεία για να σταθεροποιηθεί μεταξύ 3,1 και 3,21 στα μεγαλύτερα προβλήματα που δοκιμάστηκαν. Είναι προφανές ότι η απουσία τρίτου επιπέδου κρυφής μνήμης που να μοιράζεται μεταξύ των τεσσάρων πυρήνων στο chip της Intel, δημιουργεί αυξημένες ανάγκες για εκπομπές στον δίαυλο επικοινωνίας με την διαμοιραζόμενη κεντρική μνήμη, ώστε να ξεπεραστούν τα προβλήματα συνοχής της κρυφής μνήμης που εμφανίζονται κατά την λειτουργία των επεξεργαστών. Στην περίπτωση του Phenom, η ύπαρξη της διαμοιραζόμενης L3 cache μειώνει τις απαιτήσεις για εκπομπές στον δίαυλο επικοινωνίας όταν εμφανίζονται προβλήματα συνοχής των προσωπικών cache του κάθε επεξεργαστή.

Στην περίπτωση λοιπόν όπου η αρχιτεκτονική του επεξεργαστή απαιτεί συχνότερη επικοινωνία με την διαμοιραζόμενη κεντρική μνήμη, ο PARDISO δεν επωφελείται σημαντικά κατά την μετάβαση από δύο σε τέσσερις ενεργούς πυρήνες, αφού οι επιδόσεις του συστήματος αυξάνονται περίπου κατά 30%. Σε αρχιτεκτονική επεξεργαστή η οποία βοηθά στην εξάλειψη των φαινομένων ανταγωνισμού στην μνήμη, ο PARDISO παρουσιάζει ικανοποιητικότατη κλιμάκωση κατά την μετάβαση από δύο σε τέσσερις πυρήνες, αφού οι επιδόσεις του συστήματος αυξάνονται περίπου κατά 60%.

4.3.3 Κλιμάκωση χρόνου εκτέλεσης του PARDISO και υπερχρονισμός επεξεργαστή

Όπως αναφέρθηκε και στην ενότητα 4.1, η παρουσία ξεκλείδωτου πολλαπλασιαστή στον Phenom X4, έδωσε την δυνατότητα να πραγματοποιηθούν δοκιμές του PARDISO και σε υπερχρονισμένη διαμόρφωση του chip, με συχνότητα λειτουργίας 3,0GHz (20% πάνω από την συνήθη συχνότητα λειτουργίας).

Με την διαμόρφωση αυτή η απόδοση του επεξεργαστή αυξήθηκε από 15% μέχρι 17,3% ανάλογα με το μέγεθος του προβλήματος και τον αριθμό των πυρήνων που χρησιμοποιήθηκαν. Αξίζει όμως να σημειωθεί ότι ενώ συνολικά με τον υπερχρονισμό επετεύχθησαν σαφώς μικρότεροι χρόνοι επίλυσης, σε σχεδόν όλα τα προβλήματα και με κάθε αριθμό πυρήνων, η κλιμάκωση της απόδοσης του υπερχρονισμένου επεξεργαστή ήταν λίγο χειρότερη από αυτήν του επεξεργαστή στην συνήθη συχνότητα λειτουργίας του. Αυτό πιθανότατα οφείλεται στο γεγονός ότι ενώ αυξήθηκε η συχνότητα λειτουργίας των πυρήνων, το μέγεθος της κρυφής τους μνήμης παραμένει σταθερό αφού αποτελεί μέρος της λιθογραφίας του chip. Έτσι, καθώς οι επεξεργαστές πραγματοποιούν υπολογισμούς πιο γρήγορα, αυξάνονται οι απαιτήσεις τους ως προς την προσπέλαση της cache και της RAM δημιουργώντας συχνότερα προβλήματα συνοχής της cache και ανταγωνισμού στην μνήμη, ρίχνοντας την κλιμάκωση της απόδοσης στον PARDISO.

4.4 Σύγκριση αποτελεσμάτων με δοκιμές βιβλιογραφίας

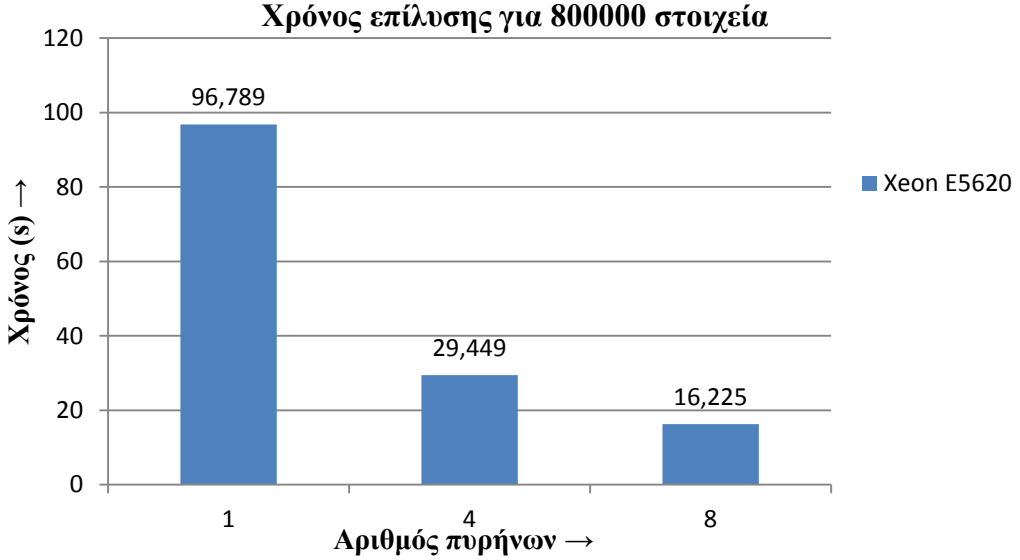
Με την έλευση της τελευταίας έκδοσης του PARDISO στις βιβλιοθήκες της Intel (MKL 10.2), το προσωπικό του προγράμματος PARDISO (pardiso-project.org) εκτέλεσε δοκιμές του επιλύτη σε ένα πρόβλημα 800000 στοιχείων για προσομοίωση μορφοποίησης μεταλλικών φύλλων στην αυτοκινητοβιομηχανία. Για τις δοκιμές τους χρησιμοποίησαν συστοιχία υπολογιστών με επεξεργαστές Intel Xeon E5620, με 12GB διαμοιραζόμενης μνήμης.

Πίνακας 4.2
Τεχνικά χαρακτηριστικά Xeon E5620

Κατασκευαστής	Intel
Ονομασία	Xeon E5620
Κωδική ονομασία chip	Gulftown
Αριθμός πυρήνων	4
Αριθμός νημάτων (threads)	8
Συχνότητα λειτουργίας	2,40-2,66GHz
Cache	12MB διαμοιραζόμενη
Σετ εντολών	64bit
Λιθογραφία	32nm
Πλατφόρμα	LGA 1366

Για το εν λόγω πρόβλημα προέκυψαν οι παρακάτω χρόνοι επίλυσης :

Διάγραμμα 4.15
Χρόνος επίλυσης για 800000 στοιχεία



Παρατηρώντας τους παραπάνω χρόνους επίλυσης, μπορούμε πολύ εύκολα να συμπεράνουμε ότι με χρήση της τελευταίας έκδοσης των βιβλιοθηκών της Intel σε συνδυασμό με πανίσχυρους επεξεργαστές τελευταίας γενιάς, αλλά και ιδιαίτερα υψηλού κόστους, μπορούμε να λύσουμε σε ελάχιστο χρόνο προβλήματα κατά πολύ μεγαλύτερα από αυτά που δοκιμάστηκαν στην παρούσα διπλωματική. Αξίζει να σημειωθεί ότι η αρχιτεκτονική Smart Cache του Xeon, στην οποία το σύνολο των 12MB κρυφής μνήμης είναι προσβάσιμο από όλους τους πυρήνες, δίνει την δυνατότητα στον PARDISO να επιτύχει εξαιρετική κλιμάκωση του χρόνου επίλυσης κατά την μετάβαση από έναν σε τέσσερις πυρήνες, αφού παρατηρούμε αύξηση των επιδόσεων του συστήματος κατά συντελεστή 3,29. Επίσης, ικανοποιητικότατη είναι και η κλιμάκωση του χρόνου επίλυσης που παρατηρήθηκε στο παραπάνω σύστημα όταν ενεργοποιήθηκαν οχτώ πυρήνες, αφού η απόδοση αυξήθηκε κατά συντελεστή 5,97.

4.5 Επίλογος

Γενικά, για να έχουμε αποδοτική οικονομικά και χρονικά λειτουργία του PARDISO πρέπει να δώσουμε ιδιαίτερη προσοχή στο μέγεθος των προβλημάτων που έχουμε να λύσουμε ώστε να χτίσουμε γύρω τους την κατάλληλη αρχιτεκτονική υπολογιστή.

Στην περίπτωση μικρών προβλημάτων, η παρουσία επεξεργαστή με πάνω από δύο πυρήνες είναι ουσιαστικά ασύμφορη, αφού ένας διπύρηνος επεξεργαστής με αρκετή

κρυφή μνήμη και μεγάλη συχνότητα λειτουργίας θα μπορούσε με συντελεστή απόδοσης κοντά στο 1,9 να εκτελέσει τον PARDISO το ίδιο γρήγορα αν όχι ταχύτερα από έναν τετραπύρηνο επεξεργαστή χαμηλότερης συχνότητας λειτουργίας που θα δουλέψει τέσσερις πιο «αργοκίνητους» πυρήνες με συντελεστή απόδοσης κοντά στο 2,5.

Στην περίπτωση μεγάλων προβλημάτων, οι παραπάνω από δύο πυρήνες αρχίζουν πλέον να έχουν λόγο ύπαρξης. Τα πολλά επίπεδα κρυφής μνήμης ενός σύγχρονου chip με τέσσερις πυρήνες και πάνω δίνουν την δυνατότητα να αυξηθεί ο συντελεστής κλιμάκωσης της απόδοσης του επεξεργαστή σε ικανοποιητικά επίπεδα, δίνοντάς μας την δυνατότητα επίλυσης ιδιαίτερα μεγάλων προβλημάτων σε κλάσματα του χρόνου που θα χρειαζόταν για την σειριακή τους επίλυση.

Φυσικά, για να έχει νόημα η χρήση του PARDISO, ή οποιουδήποτε άλλου παράλληλου επιλύτη, με επεξεργαστές δύο, τεσσάρων ή και παραπάνω πυρήνων θα πρέπει να ληφθεί υπόψη ο μεγαλύτερος «εχθρός» των παράλληλων διεργασιών, που δεν είναι άλλος από τις σειριακές διεργασίες που εκτελούνται πριν και μετά τον παράλληλο κώδικά μας. Με απλά λόγια, δεν θα είχε κανένα νόημα να ξοδέψουμε χρήμα για να εκτελέσουμε μια παράλληλη διεργασία ώστε να γλιτώσουμε μισό ή ένα λεπτό υπολογισμών, αν πριν και μετά την διεργασία αυτή εκτελούνται σειριακά προγράμματα με διάρκεια υπολογισμών είκοσι λεπτών. Πιο συγκεκριμένα, στον κώδικα που χρησιμοποιήθηκε στην παρούσα διπλωματική, πριν από κάθε εκτέλεση του PARDISO πραγματοποιούνταν η κατασκευή του προβλήματος και η ιδιαίτερα χρονοβόρα μεταφορά του αραιού πίνακα του προβλήματος από τον σκληρό δίσκο στην κεντρική μνήμη του υπολογιστή. Οι διαδικασίες αυτές διαρκούσαν από δευτερόλεπτα στα μικρά προβλήματα μέχρι αρκετά λεπτά στα μεγαλύτερα. Στην περίπτωση λοιπόν που δεν εκτελούσαμε απλές δοκιμές της απόδοσης του PARDISO, αλλά θέλαμε να μειώσουμε τον συνολικό χρόνο εκτέλεσης του προγράμματός μας, θα έπρεπε να ελαχιστοποιηθούν τα δεδομένα που εισάγονται από τον σκληρό δίσκο, ενώ η συνολική διαδικασία κατασκευής του προβλήματος θα έπρεπε να «σπάσει» σε παράλληλα νήματα και να μην εκτελείται σειριακά, αφήνοντας έναν ή περισσότερους πυρήνες ανενεργούς.

4.6 Παράρτημα

Πίνακας 4.3 Χρόνοι εκτέλεσης του PARDISO				
Εξισώσεις	Πυρήνες	Χρόνοι επίλυσης (s)		
		Phenom X4	Core 2 Quad	Phenom X4 OC
5878	1	0,246	0,210	0,204
	2	0,135	0,115	0,113
	4	0,096	0,085	0,082
23003	1	1,023	0,729	0,847
	2	0,556	0,396	0,466
	4	0,376	0,290	0,317
51378	1	2,968	2,147	2,464
	2	1,613	1,173	1,353
	4	1,020	0,855	0,862
91003	1	6,023	4,826	4,981
	2	3,238	2,595	2,710
	4	2,021	1,900	1,702
141878	1	11,932	10,618	9,880
	2	6,485	5,739	5,441
	4	3,874	4,197	3,270
204003	1	18,282	15,941	15,156
	2	9,990	8,525	8,372
	4	5,841	6,227	4,930
277378	1	29,921	26,384	24,835
	2	16,001	13,960	13,441
	4	9,621	10,387	8,139
362003	1	47,166	43,564	39,054
	2	25,495	23,296	21,390
	4	14,926	16,885	12,583
457878	1	72,809	69,597	60,431
	2	39,144	37,020	32,842
	4	22,824	27,293	19,263
565003	1	102,310	97,182	84,611
	2	56,215	52,531	46,995
	4	31,972	37,814	26,888
683378	1	142,117	135,576	117,815
	2	75,999	72,501	63,687
	4	44,691	52,346	37,719
813003	1	197,043	189,643	163,348
	2	107,088	100,340	89,633
	4	61,384	73,791	51,808

Πίνακας 4.4 Κλιμάκωση απόδοσης επεξεργαστών				
Εξισώσεις	Πυρήνες	Συντελεστής κλιμάκωσης απόδοσης		
		Phenom X4	Core 2 Quad	Phenom X4 OC
5878	2	1,82	1,83	1,80
	4	2,56	2,48	2,50
23003	2	1,84	1,84	1,82
	4	2,72	2,51	2,67
51378	2	1,84	1,83	1,82
	4	2,91	2,51	2,86
91003	2	1,86	1,86	1,84
	4	2,98	2,54	2,93
141878	2	1,84	1,85	1,82
	4	3,08	2,53	3,02
204003	2	1,83	1,87	1,81
	4	3,13	2,56	3,07
277378	2	1,87	1,89	1,85
	4	3,11	2,54	3,05
362003	2	1,85	1,87	1,83
	4	3,16	2,58	3,10
457878	2	1,86	1,88	1,84
	4	3,19	2,55	3,14
565003	2	1,82	1,85	1,80
	4	3,20	2,57	3,15
683378	2	1,87	1,87	1,85
	4	3,18	2,59	3,12
813003	2	1,84	1,89	1,82
	4	3,21	2,57	3,15

Πίνακας 4.5 Απαιτήσεις μνήμης του PARDISO	
Εξισώσεις	Μνήμη (MB)
5878	8
23003	40
51378	105
91003	217
141878	373
204003	564
277378	842
362003	1284
457878	1836
565003	2481
683378	3314
813003	4362

Βιβλιογραφία

1.7 Διεθνής Βιβλιογραφία

- Infineon Technologies North America Corporation and Kingston Technology Company, Inc. (September 2003). "Intel Dual-Channel DDR Memory Architecture White Paper", Rev. 1.0".
- Pinar Heggernes (April 1992), “Minimizing fill-in size and elimination tree height in parallel Cholesky factorization”, Department of Informatics, University of Bergen.
- Stephen Ingram, “Minimum Degree Reordering Algorithms : A Tutorial”, University of British Columbia.
- Mark T. Jones, Merrell L. Patrick, “Bunch-Kaufman factorization for real symmetric indefinite banded matrices”, NASA Langley Research Center, Hampton, Virginia.
- Charles Severance, “Shared-Memory Multiprocessors – Symmetric Multiprocessing Hardware”.
- Olaf Schenk, Klaus Gärtner, Wolfgang Fichtner, Andreas Stricker, “PARDISO : a high-performance serial and parallel sparse linear solver in semiconductor device simulation”, Integrated Systems Laboratory – Swiss Federal Institute of Technology Zürich, Weierstrass Institute for Applied Analysis and Stochastics of Berlin.
- Olaf Schenk, Klaus Gärtner, “Two-level dynamic scheduling in PARDISO : Improved scalability on shared memory multiprocessing systems”, Department of Computer Science, University of Basel – Switzerland, Weierstrass Institute for Applied Analysis and Stochastics of Berlin.
- Jennifer A. Scott, Yifan Hu, Nicolas I. M. Gould, “An evaluation of sparse direct symmetric solvers : an introduction and preliminary findings”, Computational Science and Engineering Department – Rutherford Appleton Laboratory – Chilton – Oxfordshire – England, Wolfram Research, Inc.
- Olaf Schenk, Klaus Gärtner, Wolfgang Fichtner, “Scalable parallel sparse LU factorization with a dynamical supernode pivoting approach in semiconductor

devise simulation", Integrated Systems Laboratory – Swiss Federal Institute of Technology Zürich.

1.8 Ιστοσελίδες

- Πρόγραμμα PARDISO : http://www.pardiso-project.org/
- Πληροφορίες για BLAS :
  - http://www.netlib.org/blas/faq.html
  - http://www.netlib.org/utk/people/JackDongarra/PAPERS/blast-toms.pdf
- Πληροφορίες για OpenMP : http://openmp.org/wp/about-openmp/
- Πληροφορίες για την παραγοντοποίηση Cholesky :
  - http://www.worldlingo.com/ma/enwiki/en/Cholesky_decomposition/1
  - http://www.netlib.org/utk/papers/factor/node9.html
  - http://www.worldlingo.com/ma/enwiki/en/Cholesky_decomposition/1
- Πληροφορίες για METIS :
  - http://glaros.dtc.umn.edu/gkhome/metis/metis/overview
  - http://people.sc.fsu.edu/~jburkardt/c_src/metis/metis.html
- Intel :
  - http://ark.intel.com/products/29765/Intel-Core2-Quad-Processor-Q6600-(8M-Cache-2_40-GHz-1066-MHz-FSB)
  - http://ark.intel.com/products/47925/Intel-Xeon-Processor-E5620-(12M-Cache-2_40-GHz-5_86-GTs-Intel-QPI)
- Wikipedia :
  - http://en.wikipedia.org/wiki/Cholesky_decomposition
  - http://en.wikipedia.org/wiki/LU_decomposition
  - http://en.wikipedia.org/wiki/Sparse_matrix
  - http://en.wikipedia.org/wiki/Xeon#3600.2F5600-series_.22Gulftown.22
  - http://en.wikipedia.org/wiki/List_of_AMD_Phenom_microprocessors