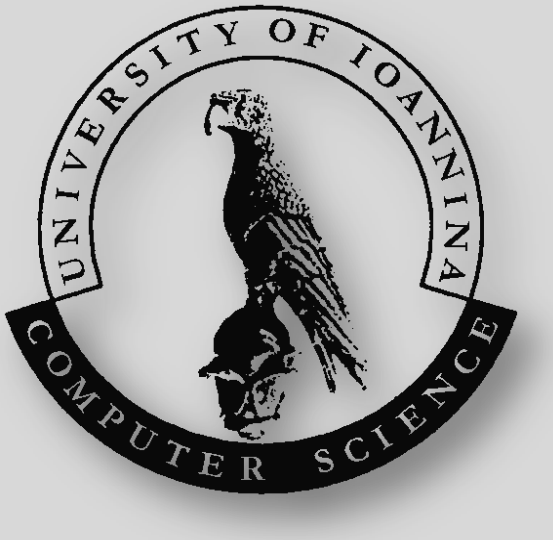
Καταγραφή χαρακτηριστικών προσπέλασης αρχείων για αναζήτηση σε συστήματα αρχείων

Καππές Γιώργος

Μιχέλη Ειρήνη

Επιβλέπων Καθηγητής: Στέργιος Β. Αναστασιάδης



Πανεπιστήμιο Ιωαννίνων

Τμήμα Πληροφορικής

Περιεχόμενα

Σχήματα

Πίνακες

Κεφάλαιο 1

Εισαγωγή

Τα τελευταία χρόνια η αλματώδης ανάπτυξη του παγκόσμιου ιστού έχει αυξήσει εκθετικά τον αριθμό των ιστοσελίδων και επομένως και τις πληροφορίες που διακινούνται. Αυτό συνετέλεσε στην επιτακτική ανάγκη δημιουργίας εργαλείων αναζήτησης, ώστε οι χρήστες να μπορούν να βρουν αυτό που θέλουν εύκολα και γρήγορα ανάμεσα στον τεράστιο όγκο δεδομένων. Τα συστήματα αναζήτησης που αναπτύχθηκαν χρησιμοποιούν κατά κόρον τεχνικές ανάλυσης περιεχομένου που προσδίδουν μεγάλη αποτελεσματικότητα στις αναζητήσεις. Οι τεχνικές αυτές βασίζονται στο ότι σχεδόν κάθε σελίδα στον παγκόσμιο ιστό περιέχει υπερσυνδέσεις (hyperlinks) προς άλλες σελίδες κοινού περιεχομένου. Το γεγονός αυτό διευκολύνει τη χρήση της ανάλυσης περιεχομένου σε συστήματα αναζήτησης στο διαδίκτυο. Η αποτελεσματικότητα αυτής της τεχνικής στο διαδίκτυο είχε ως συνέπεια τη χρήση της και σε έναν διαφορετικό τομέα, στα συστήματα αναζήτησης αρχείων με βάση το περιεχόμενο σε συστήματα αρχείων προσωπικών και εταιρικών υπολογιστών.

Η αύξηση της χωρητικότητας των αποθηκευτικών μέσων επιτρέπει στους χρήστες τους να αποθηκεύουν ολοένα και περισσότερα δεδομένα. Ο μεγάλος όγκος αποθηκευμένων δεδομένων όμως, δυσχεραίνει την οργάνωση και την εύρεση των αρχείων. Γενικά, είναι δύσκολο για τους χρήστες να θυμούνται την ακριβή θέση κάθε αρχείου, ώστε να το βρίσκουν αμέσως τη στιγμή που το χρειάζονται, ειδικά όταν εργάζονται καθημερινά στον υπολογιστή και διαχειρίζονται μεγάλο αριθμό δεδομένων. Προκειμένου να διευκολυνθούν οι χρήστες στην αυτοματοποιημένη αναζήτηση αρχείων με εύκολο και γρήγορο τρόπο έχουν αναπτυχθεί διάφορα εργαλεία αναζήτησης αρχείων όπως τα Indri [14], Terrier [15], Google Desktop [13]. Τα συστήματα αυτά χρησιμοποιούν την τεχνική ανάλυσης περιεχομένου προκειμένου να εξαγάγουν πληροφορίες περιεχομένων (content) από τα αρχεία του συστήματος και δημιουργούν καταλόγους, οι οποίοι στη συνέχεια μπορούν να χρησιμοποιηθούν για γρήγορη αναζήτηση.

Ωστόσο, στον τομέα αυτό η ανάλυση περιεχομένου δεν έχει την ίδια αποτελεσματικότητα που έχει στο διαδίκτυο και η κύρια αιτία είναι ότι στα συστήματα αρχείων δεν υπάρχουν

υπερσυνδέσεις από ένα αρχείο σε άλλα κοινού περιεχομένου, όπως συμβαίνει στο διαδίκτυο. Επιπλέον, άλλο ένα μειονέκτημα της χρήσης ανάλυσης περιεχομένου σε αρχεία είναι το γεγονός ότι δε μπορεί να χρησιμοποιηθεί σε αρχεία που τα περιεχόμενά τους είναι μη κατανοητά, όπως για παράδειγμα αρχεία εικόνας ή συμπιεσμένα αρχεία. Έτσι, οι ερευνητές αναζήτησαν άλλες μεθόδους που θα μπορούσαν να χρησιμοποιηθούν παράλληλα με την αναζήτηση περιεχομένου ώστε να βελτιωθεί η αναζήτηση αρχείων. Στο πλαίσιο αυτό πειραματίστηκαν με μια παράμετρο που μέχρι στιγμής δε χρησιμοποιούνταν από τα εργαλεία αναζήτησης: τα συμφραζόμενα (context).

Είναι γεγονός ότι οι περισσότεροι χρήστες οργανώνουν τα αρχεία τους λαμβάνοντας υπόψη το περιεχόμενό τους ώστε να μπορούν να τα εντοπίζουν εύκολα. Επιπλέον, έχει αποδειχθεί πως οι περισσότεροι χρήστες κάθε φορά που προσπελαύνουν ένα αρχείο συσχετίζουν με αυτό διάφορες ενέργειες, διεργασίες ή άλλα αρχεία που προσπέλασαν πρόσφατα. Η εύρεση των συμφραζόμενων ενός αρχείου μπορεί να πραγματοποιηθεί εξετάζοντας τα υπόλοιπα αρχεία που προσπέλασε ο χρήστης στα όρια ενός χρονικού διαστήματος (time window). Η τεχνική αυτή ονομάζεται προσωρινή τοπικότητα (temporal locality) και υλοποιήθηκε στο σύστημα αναζήτησης Connections [3].

Η χρήση της προσωρινής τοπικότητας αυξάνει την απόδοση της αναζήτησης, σε σύγκριση με την αποκλειστική χρήση τεχνικών ανάλυσης περιεχομένου, ωστόσο όπως θα δούμε, δημιουργεί την πιθανότητα να καταγραφούν πλαστές συσχετίσεις μεταξύ αρχείων. Αργότερα αναπτύχθηκε ένα νέο σύστημα, που λαμβάνει υπόψη την αιτιότητα (causality), με στόχο να ελαττωθεί η καταγραφή λανθασμένων συσχετίσεων. Η αιτιότητα χρησιμοποιεί τη ροή δεδομένων μεταξύ των εφαρμογών και των αρχείων και αναγνωρίζει με μεγαλύτερη ακρίβεια τις συσχετίσεις. Το κύριο πρόβλημα της αιτιότητας είναι η δυσκολία της καταγραφής καταστάσεων όπου η ροή των δεδομένων πραγματοποιείται σε κρυφά κανάλια, για παράδειγμα όταν ο χρήστης διαβάζει μια γραμμή από κάποιο γράμμα ηλεκτρονικού ταχυδρομείου που έλαβε και την πληκτρολογεί σε ένα έγγραφο.

1.1 – Σκοπός της εργασίας

Παρατηρώντας ότι οι παραπάνω τεχνικές παρουσιάζουν η καθεμιά τα δικά της μειονεκτήματα, συμπεραίνουμε ότι η αναζήτηση σε συστήματα αρχείων δεν μπορεί να βασίζεται αποκλειστικά είτε στην ανάλυση περιεχομένου είτε στα συμφραζόμενα. Επίσης, η χρήση και των δύο τεχνικών δεν είναι επαρκής. Οπότε θα ήταν αρκετά χρήσιμο εάν με κάποιο τρόπο συγκεντρώναμε περισσότερες πληροφορίες σχετικές με το πως τα αρχεία χρησιμοποιούνται από το χρήστη.

Στην παρούσα εργασία προτείνουμε ως λύση στο πρόβλημα της αναζήτησης δεδομένων κειμένου την καταγραφή στατιστικών της χρήσης των αρχείων. Έτσι, τα αρχεία που έχει χρησιμοποιήσει πρόσφατα ο χρήστης καταγράφονται σε διάφορες δομές ώστε να έχει τη δυνατότητα να τα βρίσκει πολύ γρήγορα. Για το σκοπό αυτό αναπτύσσουμε ένα σύστημα το οποίο βρίσκεται μεταξύ του επιπέδου των οδηγών συσκευών και των εφαρμογών και κρατάει στατιστικά της χρήσης των αρχείων σε έναν πίνακα κατακερματισμού πιστεύοντας ότι βελτιώνεται ο χρόνος της αναζήτησης των αρχείων που καταγράφονται στον πίνακα. Έπειτα, χρησιμοποιούμε ένα γράφο συσχετίσεων ώστε να καταγράψουμε τις συνδέσεις μεταξύ των αρχείων. Με τον τρόπο αυτό δημιουργούμε ένα περιβάλλον εικονικών συνδέσεων μεταξύ των αρχείων στο οποίο μπορούν να τρέξουν αλγόριθμοι κατάταξης όπως ο PageRank [12] και ο HITS [11]. Τέλος, εκτελούμε διάφορες μετρήσεις για να διαπιστώσουμε πόσο επιβαρύνεται το σύστημα από την κάθε μέθοδο υλοποίησης και αν η μέθοδος στην οποία χρησιμοποιούμε πίνακα κατακερματισμού είναι καλύτερη από τις ήδη υπάρχουσες.

1.2 – Διάταξη ενοτήτων

Η παρούσα εργασία δομείται ως εξής: Το κεφάλαιο 2 περιλαμβάνει μια ανασκόπηση των μεθόδων που έχουν προταθεί και υλοποιηθεί για αναζήτηση σε συστήματα αρχείων. Στο κεφάλαιο 3 παρουσιάζουμε σύντομα την αρχιτεκτονική του λειτουργικού συστήματος Windows XP και εισάγουμε κάποιες γενικές έννοιες για τους οδηγούς συσκευών. Επιπλέον, αναλύουμε ορισμένα προγραμματιστικά θέματα που αφορούν τους οδηγούς όπως ο συγχρονισμός και η χρήση της μνήμης. Στο κεφάλαιο 4, αφού προτείνουμε την μέθοδο καταγραφής στατιστικών της χρήσης των αρχείων ως μια εναλλακτική μέθοδο οργάνωσης των αρχείων για την αποτελεσματικότερη αναζήτησή τους, ορίζουμε την αρχιτεκτονική σχεδίαση του συστήματος που υλοποιούμε. Στο κεφάλαιο 5 κάνουμε μια αναφορά στις μεθόδους που ακολουθήσαμε για την υλοποίηση του συστήματος και παρουσιάζουμε τις δομές και τους αλγορίθμους που χρησιμοποιούμε. Στο κεφάλαιο 6 αρχικά κάνουμε μια περιγραφή του περιβάλλοντος στο οποίο εκτελέσαμε τα πειράματα και των εργαλείων που χρησιμοποιήσαμε. Ακολούθως, παρουσιάζουμε τα αποτελέσματα των πειραμάτων. Τέλος, στο κεφάλαιο 7 κάνουμε μια αναφορά στα συμπεράσματα της εργασίας.

Κεφάλαιο 2

Προηγούμενη έρευνα

Με την αύξηση της χωρητικότητας των αποθηκευτικών μέσων, η αναζήτηση στο διαδίκτυο έγινε πιο αποτελεσματική από την αναζήτηση σε έναν προσωπικό υπολογιστή. Οι τεχνικές αναζήτησης που χρησιμοποιούνται στο διαδίκτυο δε μπορούν να χρησιμοποιηθούν με την ίδια αποτελεσματικότητα στα συστήματα αρχείων προσωπικών υπολογιστών γιατί τα αρχεία οργανώνονται στο δίσκο με διαφορετικό τρόπο από ότι οι ιστοσελίδες στο διαδίκτυο. Για το λόγο αυτό, ο τομέας της αναζήτησης πληροφοριών σε συστήματα αρχείων προσωπικών και εταιρικών υπολογιστών έχει αρχίσει να κεντρίζει το ενδιαφέρον.

Στο κεφάλαιο αυτό, αρχικά παρουσιάζουμε αλγόριθμους και κριτήρια κατάταξης που χρησιμοποιούνται για την κατάταξη των αποτελεσμάτων της αναζήτησης στο διαδίκτυο και σε συστήματα αρχείων προσωπικών και εταιρικών υπολογιστών. Στη συνέχεια, εξετάζουμε τις μεθόδους αναζήτησης σε συστήματα αρχείων που έχουν προταθεί και υλοποιηθεί.

2.1 – Κριτήρια κατάταξης αποτελεσμάτων αναζήτησης

Η διαδικασία της αναζήτησης πληροφοριών στον παγκόσμιο ιστό έχει γίνει αρκετά δύσκολη εξαιτίας του μεγάλου όγκου δεδομένων. Ποιες ιστοσελίδες θέλει πραγματικά ο χρήστης να ανακτήσει όταν πληκτρολογεί λέξεις κλειδιά σε μια μηχανή αναζήτησης; Μπορεί να υπάρχουν χιλιάδες σελίδες που περιέχουν τις λέξεις κλειδιά που πληκτρολόγησε, αλλά ένα πολύ μικρότερο υποσύνολό τους τον ενδιαφέρει πραγματικά αφού μαζί με τις έγκυρες και τεκμηριωμένες ιστοσελίδες υπάρχουν και κάποιες που το περιεχόμενό τους είναι ανακριβές και λανθασμένο. Έτσι, ο χρήστης απαιτεί από τη μηχανή αναζήτησης να ξεχωρίζει τις σχετικές από τις ανακριβείς ιστοσελίδες και να τις κατατάσσει έτσι ώστε οι σχετικότερες να εμφανίζονται πρώτες.

2.1.1 – Χρήση των κριτηρίων κατάταξης στον παγκόσμιο ιστό

Ως απάντηση στο παραπάνω πρόβλημα δημιουργήθηκαν διάφοροι αλγόριθμοι κατάταξης και χρησιμοποιήθηκαν στις μηχανές αναζήτησης του διαδικτύου. Τέτοιοι αλγόριθμοι είναι ο HITS [11] και ο PageRank [12], οι οποίοι αναλύουν τη δομή των συνδέσεων (hyperlinks) από μια ιστοσελίδα προς άλλη του διαδικτύου, έτσι ώστε να προσφέρουν έναν απλό μηχανισμό αξιολόγησης της ποιότητας και της σχετικότητας του περιεχομένου μιας ιστοσελίδας. Οι αλγόριθμοι αυτοί ιστορικά πετυχαίνουν καλά αποτελέσματα, ωστόσο πρόσφατες μελέτες αποδεικνύουν ότι πιο απλές μέθοδοι μπορούν να πετύχουν τα ίδια ή καλύτερα αποτελέσματα από τους παραπάνω αλγορίθμους. Επιπλέον, υπάρχουν πολλοί χρήστες που προσπαθούν να εκμεταλλευτούν τους αλγορίθμους αυτούς, έτσι ώστε να πετύχουν καλύτερη κατάταξη των ιστοσελίδων τους. Για να το πετύχουν αυτό, δημιουργούν πολλές ιστοσελίδες με μικρό και ανακριβές περιεχόμενο (spam) με συνδέσεις προς μια ιστοσελίδα στόχο. Η τεχνική αυτή (link stuffing) μπορεί να εξαπατήσει τις μηχανές αναζήτησης οι οποίες χρησιμοποιούν αλγόριθμο κατάταξης βασισμένο στην ανάλυση των συνδέσεων.

Τα προβλήματα που παρουσιάζουν οι αλγόριθμοι κατάταξης που χρησιμοποιούν ανάλυση συνδέσεων, όπως ο PageRank, οδήγησαν τους ερευνητές να ανακαλύψουν νέες μεθόδους για την βελτίωση της κατάταξης των ιστοσελίδων. Οι Richardson et al. [5], πρότειναν τη χρήση ενός νευρωνικού δικτύου (RankNet) για το πρόβλημα της κατάταξης και απέδειξαν ότι αναγνωρίζει τις σχετικές από τις ανακριβείς ιστοσελίδες με μεγαλύτερη επιτυχία απ’ ότι ο αλγόριθμος PageRank. Διαπιστώθηκε ότι υπάρχουν πολλά χαρακτηριστικά που διακρίνουν μια ιστοσελίδα τα οποία μπορούν να χρησιμοποιηθούν σε ένα νευρωνικό δίκτυο ως κριτήρια για τη δημιουργία ενός αξιόπιστου μηχανισμού κατάταξης. Επειδή το νευρωνικό δίκτυο μπορεί να χρησιμοποιήσει διάφορους συνδυασμούς των κριτηρίων καθίσταται πιο εύκολη και αποτελεσματική η άμυνα απέναντι στους spammers. Αν κάποιος spammer ανακαλύψει και εκμεταλλευτεί κάποιο από τα χαρακτηριστικά μιας ιστοσελίδας, μπορεί πολύ εύκολα το συγκεκριμένο χαρακτηριστικό να αφαιρεθεί από το νευρωνικό δίκτυο. Έτσι, γίνεται φανερή η δυνατότητα των μηχανισμών κατάταξης που βασίζονται σε νευρωνικά δίκτυα να προσαρμόζονται σε νέες τεχνικές που χρησιμοποιούν οι spammers.

Όπως αναφέραμε και παραπάνω, οι μηχανισμοί κατάταξης που βασίζονται σε νευρωνικά δίκτυα εκμεταλλεύονται διάφορα χαρακτηριστικά των ιστοσελίδων τα οποία χρησιμοποιούν ως κριτήρια για να ξεχωρίσουν τις έγκυρες και σχετικές από τις ανακριβείς. Στη συνέχεια, θα

αναφέρουμε και θα εξηγήσουμε κάποια από τα πιο σημαντικά χαρακτηριστικά που χρησιμοποιεί το RankNet.

- Δημοτικότητα (Popularity). Ένα από τα πιο σημαντικά χαρακτηριστικά μιας ιστοσελίδας είναι η δημοτικότητά της. Η δημοτικότητα εκφράζεται ως ο αριθμός των επισκέψεων που δέχτηκε η ιστοσελίδα μέσα σε ένα καθορισμένο χρονικό διάστημα. Ιστοσελίδες με μεγάλη επισκεψιμότητα αποκτούν μεγαλύτερη εγκυρότητα από τις ιστοσελίδες με μικρή επισκεψιμότητα.

- Κείμενο υπερσυνδέσεων προς την ιστοσελίδα (Anchor text). Ένα επιπλέον χαρακτηριστικό που μπορεί να χρησιμοποιηθεί ως κριτήριο κατάταξης είναι το κείμενο μιας υπερσύνδεσης προς την ιστοσελίδα που αναλύεται και το πλήθος των ξεχωριστών λέξεων που περιλαμβάνει.

- Περιεχόμενα σελίδας. Εκτός από τα παραπάνω, μπορούν να χρησιμοποιηθούν ως κριτήρια ορισμένα χαρακτηριστικά που διέπουν το περιεχόμενο της ιστοσελίδας, όπως το πλήθος των λέξεων που περιλαμβάνονται στο σώμα και η συχνότητα του πιο συχνού όρου.

- PageRank. Το «βάρος» μιας ιστοσελίδας, όπως υπολογίζεται με βάση τον αλγόριθμο PageRank.

- Πεδίο (domain). Ως κριτήρια κατάταξης μπορούν να χρησιμοποιηθούν επίσης μέσοι όροι χαρακτηριστικών που διέπουν όλες τις ιστοσελίδες που ανήκουν στο ίδιο πεδίο.

2.1.2 – Κριτήρια κατάταξης σε αναζήτηση σε συστήματα αρχείων

Αν και στα συστήματα αρχείων δεν υπάρχει το spam όπως ορίζεται στο διαδίκτυο, ο συνεχής αυξανόμενος αποθηκευτικός χώρος που διαθέτουν πλέον οι προσωπικοί υπολογιστές ωθεί τους χρήστες να μη σβήνουν από το δίσκο έγγραφα ή αρχεία που δε χρειάζονται πλέον. Κατά τη διαδικασία της αναζήτησης, τα αρχεία αυτά είναι δυνατό να παίξουν το ρόλο του spam και να καταταχτούν σε υψηλές θέσεις της λίστας των αποτελεσμάτων, αφήνοντας πίσω τους άλλα σημαντικά αρχεία που πιθανόν ενδιαφέρουν τον χρήστη. Το γεγονός αυτό, σε συνδυασμό με την έλλειψη συνδέσεων από ένα αρχείο σε άλλα, καθιστά αδύνατη τη χρήση αλγορίθμων κατάταξης που βασίζονται στην ανάλυση συνδέσεων όπως ο PageRank. Διαφαίνεται έτσι η σημασία της χρήσης κριτηρίων κατάταξης για την αξιολόγηση και κατάταξη των αποτελεσμάτων, έτσι ώστε σημαντικά

αρχεία για τα οποία ενδιαφέρεται ο χρήστης να εμφανίζονται στις πρώτες θέσεις των αποτελεσμάτων.

Η χρήση νευρωνικών δικτύων για αναζήτηση σε προσωπικούς υπολογιστές, σε συνδυασμό με την κατάλληλη επιλογή των κριτηρίων, έχει αποδειχτεί ότι βελτιώνει την κατάταξη των αποτελεσμάτων της αναζήτησης. Λαμβάνοντας υπόψη τον τρόπο που οργανώνουν οι χρήστες τα αρχεία τους και τις χρονικές στιγμές που τα προσπελαύνουν, μπορούμε να χρησιμοποιήσουμε κάποια από τα χαρακτηριστικά των αρχείων ως κριτήρια κατάταξης των αποτελεσμάτων της αναζήτησης:

- Ημερομηνίες δημιουργίας και τελευταίας προσπέλασης του αρχείου. Οι ημερομηνίες δημιουργίας και τελευταίας προσπέλασης ενός αρχείου μπορούν να παίξουν καθοριστικό ρόλο για την κατάταξη του αρχείου. Αρχεία που δημιουργήθηκαν ή και προσπελάστηκαν πρόσφατα έχουν μεγαλύτερη πιθανότητα να ενδιαφέρουν τον χρήστη από τα παλαιότερα αρχεία. Με το κριτήριο αυτό μπορούμε να εξασφαλίσουμε ότι αρχεία που ο χρήστης δε χρειάζεται αλλά ταυτόχρονα δεν έχει διαγράψει από το δίσκο θα εμφανιστούν σε χαμηλότερες θέσεις της αναζήτησης.

- Διαδρομή καταλόγων όπου βρίσκεται το αρχείο (path). Ένα άλλο κριτήριο που μπορεί να χρησιμοποιηθεί είναι η διαδρομή και ο κατάλογος που βρίσκεται ένα αρχείο στο σύστημα. Αρχεία που βρίσκονται σε καταλόγους που δεν ενδιαφέρουν το χρήστη, για παράδειγμα κατάλογοι συστήματος (c:\windows) σε περίπτωση που ο χρήστης αναζητάει ένα έγγραφο, θα πρέπει να εμφανίζονται σε χαμηλότερες θέσεις τις αναζήτησης.

- Τύπος αρχείου – κατάληξη (file extension). Επίσης, ο τύπος του αρχείου μπορεί να παίξει σημαντικό ρόλο για την κατάταξή του κατά την αναζήτηση. Για παράδειγμα, αν ο χρήστης ενδιαφέρεται για ένα αρχείο κειμένου, εκτελέσιμα αρχεία που τυχαίνει να έχουν το ίδιο όνομα με το αρχείο που ενδιαφέρει το χρήστη θα πρέπει να εμφανιστούν σε χαμηλότερες θέσεις της αναζήτησης.

- Περιεχόμενα αρχείου (file content). Εκτός από τα παραπάνω, το περιεχόμενο ενός αρχείου μπορεί επίσης να χρησιμοποιηθεί ως κριτήριο κατάταξής του. Ωστόσο, η ανάλυση περιεχομένου μπορεί να γίνει μόνο σε αρχεία κειμένου και όχι σε εικόνες, βίντεο ή εκτελέσιμα αρχεία.

2.2 – Αναζήτηση με βάση τα περιεχόμενα

Αρχικά, τα πρώτα συστήματα αναζήτησης αρχείων σε προσωπικούς υπολογιστές βασίζονταν στην ανάλυση περιεχομένου. Στο βαθμό που είναι γνωστός ο τρόπος λειτουργίας τους, το Spotlight [8] των Mac OS, το Windows Desktop Search [10], το Google Desktop [13] και το Yahoo! Desktop Search [9], είναι μερικά από τα συστήματα που χρησιμοποιούν την παραπάνω τεχνική. Η βασική ιδέα είναι ότι αναλύονται τα περιεχόμενα των αρχείων. Κάθε αρχείο προσδιορίζεται από κάποια χαρακτηριστικά που περιγράφουν το περιεχόμενό του και δημιουργείται ένα ευρετήριο. Ο χρήστης πληκτρολογεί λέξεις-κλειδιά που αντιπροσωπεύουν το αρχείο το οποίο θέλει να βρει και στη συνέχεια το σύστημα εξετάζει αν οι λέξεις-κλειδιά ταιριάζουν με κάποια από τα χαρακτηριστικά των αρχείων που είχε ήδη συλλέξει στο ευρετήριο. Στη συνέχεια, τα αρχεία που φαίνεται να ταιριάζουν, κατατάσσονται με βάση τα κριτήρια κατάταξης και εμφανίζονται ως αποτελέσματα στο χρήστη.

2.2.1 – Mac OS

Από την έκδοση X, το λειτουργικό σύστημα MAC OS περιλαμβάνει ένα εξελιγμένο εργαλείο αναζήτησης, το Spotlight [8]. Το Spotlight χρησιμοποιεί τα μεταδεδομένα των αρχείων και τα οργανώνει σε ευρετήρια, έτσι ώστε να βελτιώνεται η διαδικασία αναζήτησής τους. Τα μεταδεδομένα περιλαμβάνουν πληροφορίες που χαρακτηρίζουν ένα αρχείο, όπως για παράδειγμα το όνομα του δημιουργού του ή την ημερομηνία τελευταίας τροποποίησης. Επίσης, είναι δυνατό να περιλαμβάνουν διάφορες λέξεις κλειδιά ή άλλες πληροφορίες που προσδιορίζουν ένα αρχείο.

Σε αντίθεση με άλλα συστήματα αναζήτησης αρχείων, το Spotlight είναι μέρος του λειτουργικού συστήματος και συνεργάζεται απευθείας με το σύστημα αρχείων. Η Apple έχοντας ως στόχο τη βελτίωση της αναζήτησης αρχείων σχεδίασε ένα σύστημα αρχείων που είναι αποδοτικό στις αναζητήσεις αρχείων, το HFS+. Κάθε φορά που ένα αρχείο δημιουργείται, τροποποιείται ή διαγράφεται, το σύστημα αρχείων διασφαλίζει ότι τα ευρετήρια αναζήτησης του συστήματος είναι κατάλληλα ενημερωμένα.

Το Spotlight περιλαμβάνει το κυρίως πρόγραμμα το οποίο είναι υπεύθυνο για την ενημέρωση των ευρετηρίων. Επιπλέον, αποτελείται από διάφορα επιπρόσθετα προγράμματα εισαγωγείς μεταδεδομένων (Spotlight importers) τα οποία είναι υπεύθυνα για να εξάγουν τα μεταδεδομένα από συγκεκριμένους τύπους αρχείων. Την πρώτη φορά που ο χρήστης θα εισέλθει στο λειτουργικό σύστημα, το Spotlight δημιουργεί ευρετήρια μεταδεδομένων τα οποία χαρακτηρίζουν τα αρχεία που υπάρχουν στο σύστημα. Έπειτα, κάθε φορά που δημιουργείται ένα νέο αρχείο,

τροποποιείται ή διαγράφεται ένα υπάρχον, ο πυρήνας του λειτουργικού συστήματος ενημερώνει το Spotlight, το οποίο αφού αναγνωρίσει τον τύπο του αρχείου, καλεί τον κατάλληλο εισαγωγέα μεταδεδομένων για να εξάγει τα μεταδεδομένα από το αρχείο και τα εισάγει στο ευρετήριο. Όταν ο χρήστης αναζητήσει κάποιο αρχείο, το Spotlight αναζητά το ευρετήριο μεταδεδομένων και επιστρέφει τα αρχεία που ικανοποιούν τα κριτήρια της αναζήτησης.

2.2.2 – Windows XP

Τα Windows XP δίνουν και αυτά τη δυνατότητα στο χρήστη να αναζητά αρχεία στο δίσκο του συστήματος παρέχοντας του ένα εργαλείο αναζήτησης. Το Windows Desktop Search [10] δημιουργεί ένα πλήρες ευρετήριο των αρχείων, χρησιμοποιώντας τα ονόματα των αρχείων, τα περιεχόμενά τους, σε όσα από αυτά είναι κατανοητό, αλλά και τα μεταδεδομένα του κάθε αρχείου. Από την ολοκλήρωση της δημιουργίας του αρχικού ευρετηρίου και έπειτα, το ευρετήριο ανανεώνεται καθώς νέα αρχεία προστίθενται στο σύστημα. Ο χρήστης για να κάνει μια αναζήτηση πρέπει να πληκτρολογήσει κάποιες λέξεις κλειδιά τις οποίες το Windows Desktop Search χρησιμοποιεί για να δει αν ταιριάζουν με κάποια από τις εγγραφές του ευρετηρίου. Για την ενίσχυση της αναζήτησης ο χρήστης μπορεί να επιλέξει κριτήρια, όπως αναζήτηση μόνο για συγκεκριμένους τύπους αρχείων, για αρχεία που έχουν τροποποιηθεί σε συγκεκριμένα χρονικά διαστήματα και για αρχεία με συγκεκριμένο μέγεθος.

2.2.3 – Google Desktop

Εργαλείο αναζήτησης σε συστήματα αρχείων είναι και το Google Desktop [13] το οποίο, όπως και το Windows Desktop Search, στηρίζεται στην ανάλυση περιεχομένων και τη δημιουργία ευρετηρίου. Το Google Desktop δημιουργεί ένα αντίγραφο του ευρετηρίου στην κρυφή μνήμη έτσι ώστε να γίνεται γρηγορότερα η πρόσβαση σε αυτό. Η δημιουργία του ευρετηρίου γίνεται μόνο όταν ο υπολογιστής είναι αδρανής πάνω από τριάντα δευτερόλεπτα, ώστε να μη δυσχεραίνει την απόδοσή του. Σημαντικό είναι το ότι ο χρήστης μπορεί να βρει και αρχεία που έχει διαγράψει, εάν το Google Desktop έχει αποθηκεύσει αντίγραφα αυτών όταν είχαν αναζητηθεί παλιότερα πριν διαγραφούν. Η δημιουργία ευρετηρίου συμπεριλαμβάνει αρκετούς τύπους αρχείων, ενώ με εγκατάσταση επιπλέον λογισμικού μπορούν να συμπεριληφθούν στο ευρετήριο ακόμα περισσότεροι τύποι αρχείων. Στο παράθυρο όπου γίνεται η πληκτρολόγηση των λέξεων κλειδιών, αρχικά εμφανίζονται τα αρχεία που είναι συχνότερα στα αποτελέσματα των αναζητήσεων. Πληκτρολογώντας κάποια γράμματα στο

πλαίσιο, πριν καν ολοκληρωθούν οι λέξεις κλειδιά, αρχίζουν να εμφανίζονται τα πρώτα αποτελέσματα. Και όταν γραφεί πλήρως η λέξη κλειδί τότε θα εμφανιστούν και τα αποτελέσματα με τη βέλτιστη αντιστοιχία. Το Google Desktop δίνει τη δυνατότητα ανανέωσης του ευρετηρίου καθώς νέα αρχεία προστίθενται στο σύστημα ή τροποποιούνται τα ήδη υπάρχοντα.

2.2.4 – Yahoo! Desktop Search

Το εργαλείο αναζήτησης σε συστήματα αρχείων της Yahoo! είναι παρόμοιο με το Google Desktop και βασίζεται και αυτό στη δημιουργία και χρήση ενός ευρετηρίου αναζήτησης. Κατά την πρώτη εκτέλεση του προγράμματος δημιουργείται το αρχικό ευρετήριο το οποίο περιλαμβάνει τα αρχεία που βρίσκονται στους τοπικούς δίσκους καθώς και τα γράμματα ηλεκτρονικού ταχυδρομείου που έχει αποθηκευμένα ο χρήστης. Το πρόγραμμα δίνει στο χρήστη τη δυνατότητα να επιλέξει για εισαγωγή στο ευρετήριο τις πληροφορίες που χαρακτηρίζουν ένα αρχείο. Για παράδειγμα, ο χρήστης μπορεί να επιλέξει να γίνει εισαγωγή στο ευρετήριο του ονόματος του αρχείου, του μεγέθους του, του περιεχομένου του ή συνδυασμός των παραπάνω.

Αφού δημιουργηθεί το αρχικό ευρετήριο, ο χρήστης μπορεί να επιλέξει τη χρονική στιγμή που επιθυμεί να ενημερώνεται το ευρετήριο για αλλαγές. Επίσης, μπορεί να επιλέξει να μην ενημερώνεται το ευρετήριο όταν εργάζεται σε εφαρμογές και ο επεξεργαστής του συστήματος είναι απασχολημένος διότι η ενημέρωση του ευρετηρίου καταναλώνει σχεδόν το 90% των πόρων του επεξεργαστή.

2.3 – Αναζήτηση με βάση τα συμφραζόμενα

Για την αύξηση της απόδοσης στην αναζήτηση αρχείων χρησιμοποιήθηκε μια νέα τεχνική που βασίζεται στα συμφραζόμενα (context). Τα συμφραζόμενα ενός αρχείου μπορεί να είναι άλλα αρχεία που προσπέλασε ο χρήστης πρόσφατα, η εργασία που εκτελούσε τη στιγμή που προσπέλασε το συγκεκριμένο αρχείο και γενικά οποιαδήποτε ενέργεια ή δεδομένα συσχετίζει ο χρήστης με κάποιο αρχείο. Όταν ο χρήστης θελήσει να αναζητήσει κάποιο αρχείο είναι πολύ πιθανό να μη θυμάται την ακριβή του θέση, ωστόσο να θυμάται άλλα αρχεία που χρησιμοποιούσε παράλληλα με το αρχείο που αναζητεί και να οδηγηθεί έτσι στη θέση του αρχείου. Μπορούμε να διαπιστώσουμε ότι τα συμφραζόμενα ενός αρχείου παίζουν σπουδαίο ρόλο στον προσδιορισμό της θέσης του αρχείου και ο αυτόματος προσδιορισμός των συμφραζομένων θα μπορούσε να βοηθήσει το χρήστη να βρει πολύ εύκολα το αρχείο που ψάχνει στον προσωπικό του υπολογιστή.

2.3.1 – Προσωρινή τοπικότητα

Ένα ζήτημα που προκύπτει είναι πως μπορούμε να ανακαλύψουμε τα συμφραζόμενα των αρχείων που χρησιμοποιεί ο χρήστης. Μια λύση που προτάθηκε και υλοποιήθηκε είναι η προσωρινή τοπικότητα (temporal locality): Αρχεία που προσπελαύνονται μέσα σε ένα συγκεκριμένο χρονικό διάστημα θεωρείται ότι σχετίζονται μεταξύ τους. Ένα σύστημα που χρησιμοποιεί την προσωρινή τοπικότητα είναι το Connections [3] το οποίο θα περιγράψουμε σύντομα στην παρούσα ενότητα.

Το Connections είναι ένα εργαλείο αναζήτησης αρχείων που συνδυάζει την αναζήτηση βασισμένη στο περιεχόμενο των αρχείων και τις πληροφορίες από τα συμφραζόμενα που συγκεντρώνονται μέσω των ενεργειών του χρήστη πάνω σε αρχεία που υπάρχουν στο σύστημα. Κάνοντας χρήση των πληροφοριών αυτών αναγνωρίζει προσωρινές συσχετίσεις μεταξύ των αρχείων.

Οι περισσότεροι χρήστες συνήθως οργανώνουν τα αρχεία τους κάτω από καταλόγους σύμφωνα με το περιεχόμενό τους ή σύμφωνα με τον τρόπο που σχετίζονται μεταξύ τους. Για παράδειγμα, όλα τα αρχεία που χρησιμοποιούνται για μια συγκεκριμένη εργασία θα σχετίζονται μεταξύ τους και θα είναι στον ίδιο φάκελο. Αυτού του είδους οι συσχετίσεις δεν μπορούν να βρεθούν χρησιμοποιώντας την ανάλυση περιεχομένου των αρχείων. Οπότε, εντοπίζοντας τις κλήσεις συστήματος που αφορούν τα αρχεία, το Connections μπορεί να δημιουργήσει προσωρινές σχέσεις μεταξύ των αρχείων, οι οποίες προκύπτουν από τις προσπελάσεις του χρήστη στα αρχεία αυτά. Αρχεία που προσπελαύνονται μέσα σε ένα συγκεκριμένο χρονικό διάστημα θεωρείται ότι σχετίζονται μεταξύ τους. Η συσχέτιση αυτή αναπαρίσταται με ένα γράφο, όπου κάθε κόμβος αντιπροσωπεύει ένα αρχείο και οι ακμές δηλώνουν ότι υπάρχει συσχέτιση μεταξύ των δύο αρχείων που ενώνονται. Κάθε ακμή προσδιορίζεται από ένα βάρος που δείχνει πόσο ισχυρή είναι η συγκεκριμένη συσχέτιση, δηλαδή πόσες φορές έχει καταγραφεί. Ουσιαστικά το Connections λειτουργεί ακολουθώντας τα παρακάτω βήματα:

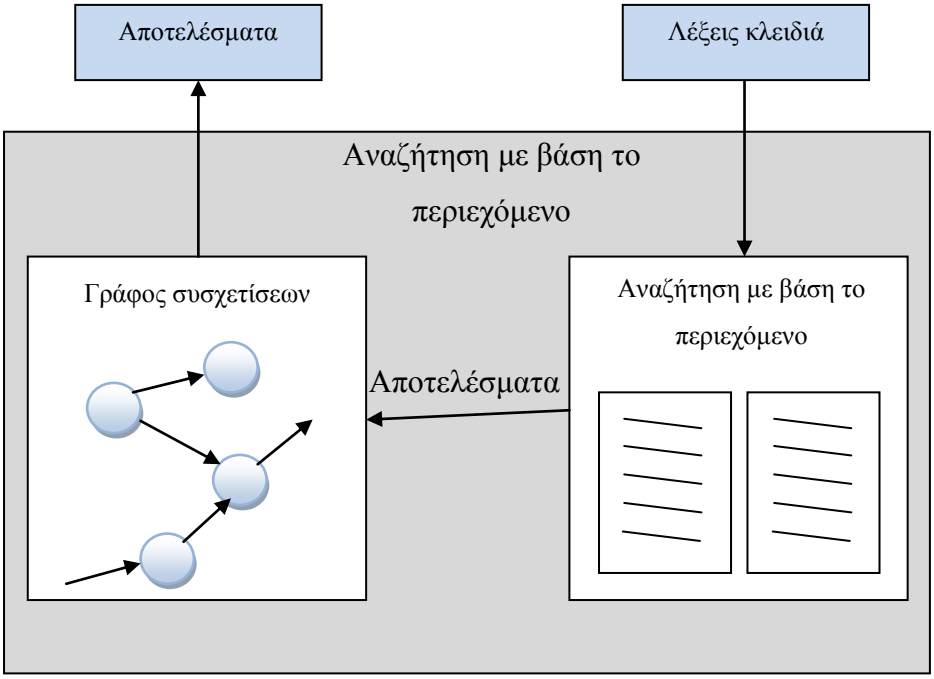
- Όταν ο χρήστης θέλει να ψάξει για ένα αρχείο, εισάγει διάφορες λέξεις κλειδιά. Τότε το Connections χρησιμοποιεί την ανάλυση περιεχομένου, προκειμένου να βρει το αρχείο που ζητείται και ανακτά κάποια αποτελέσματα.

- Ανιχνεύει τις συσχετίσεις μεταξύ των αρχείων που χρησιμοποιεί ο χρήστης και τις αποτυπώνει χρησιμοποιώντας ένα γράφο. Για να καταγραφούν οι συσχετίσεις μεταξύ των αρχείων το Connections χρησιμοποιεί έναν «ανιχνευτή» κλήσεων συστήματος για τα αρχεία

και το γράφο συσχετίσεων που προαναφέρθηκε. Ο «ανιχνευτής» βρίσκεται μεταξύ των εφαρμογών και του συστήματος αρχείων, καταγράφοντας τη δραστηριότητα του τελευταίου.

- Στη συνέχεια, χρησιμοποιώντας τα αποτελέσματα από την ανάλυση περιεχομένου εντοπίζει μέσω του γράφου τα αρχεία με τα οποία συσχετίζονται και δημιουργεί ένα μικρότερο γράφο με αυτά.

- Τέλος, ταξινομεί τα συνδυασμένα αποτελέσματα που προκύπτουν από την ανάλυση περιεχομένου και τις συσχετίσεις και επιστρέφει στο χρήστη το τελικό αποτέλεσμα.



Σχήμα 2.1 – Απεικόνιση της λειτουργίας του Connections. Ο χρήστης τροφοδοτεί με λέξεις κλειδιά το σύστημα και με χρήση αυτών γίνεται μια αναζήτηση με βάση τα περιεχόμενα. Τα αποτελέσματα της αναζήτησης αυτής εισάγονται στον γράφο συσχετίσεων.

Το Connections παρακολουθεί τις αιτήσεις που γίνονται προς τα αρχεία και δημιουργεί το γράφο. Δύο ή περισσότερα αρχεία συνδέονται στο γράφο εάν προσπελαστούν μέσα σε ένα συγκεκριμένο χρονικό διάστημα. Το διάστημα αυτό προσδιορίζει το χρόνο κατά τον οποίο ο χρήστης ασχολείται με κάποια συγκεκριμένη εργασία, οπότε και τα αρχεία που θα χρησιμοποιήσει θα συσχετίζονται. Όταν γίνει αίτηση προς ένα αρχείο τότε προστίθεται στο γράφο με βάρος ένα και κάθε φορά που γίνεται αίτηση προς το συγκεκριμένο αρχείο μέσα στο ίδιο χρονικό διάστημα το βάρος της ακμής αυξάνεται κατά ένα. Οι ακμές του γράφου μπορεί να είναι κατευθυνόμενες ή μη και αυτό έχει να κάνει με την κατεύθυνση προς την οποία γίνεται η διάσχισή του.

Η διάσχιση του γράφου ξεκινάει από τον κόμβο στον οποίο υπάρχει το αρχείο που ψάχνει ο χρήστης και η λύση περιλαμβάνει τα αρχεία για τα οποία υπάρχει μονοπάτι προς τον αρχικό κόμβο. Όμως υπάρχει η πιθανότητα να έχουν σχηματιστεί λανθασμένες ακμές και να συνδέονται αρχεία τα οποία δεν συσχετίζονται, για παράδειγμα ο χρήστης να διαβάζει την αλληλογραφία του και να ακούει κάποιο τραγούδι. Για να μετριάσει τέτοιου είδους σφάλματα, το Connections χρησιμοποιεί δύο παραμέτρους: το μήκος του μονοπατιού και το βάρος.

- Μήκος μονοπατιού: Το μήκος μονοπατιού καθορίζει το μέγιστο αριθμό κόμβων που θα ληφθούν υπόψη στη λύση. Ξεκινώντας από το ζητούμενο αρχείο ακολουθώντας τις ακμές, όσο απομακρυνόμαστε από τον κόμβο εκκίνησης τόσο μικρότερη είναι η συσχέτιση μεταξύ των αρχείων. Οπότε, ελαχιστοποιώντας το μήκος του μονοπατιού μειώνουμε την πιθανότητα να συμπεριλάβουμε στη λύση μας αρχεία που δεν σχετίζονται μεταξύ τους.

- Βάρος: Το βάρος των ακμών που απαρτίζουν το γράφο θα πρέπει να έχει τουλάχιστον μια προκαθορισμένη τιμή προκειμένου να συμπεριληφθεί στη λύση το αρχείο στο οποίο οδηγεί η ακμή. Οπότε δεν λαμβάνονται υπόψη αρχεία που προσπελαύνονται συχνά αλλά δεν συσχετίζονται μεταξύ τους.

Αφού βρεθούν τα αρχεία που συσχετίζονται με αυτό που αναζητά ο χρήστης, κατατάσσονται ώστε να έχουμε πρώτα τα αρχεία που συσχετίζονται περισσότερο με το ζητούμενο αρχείο. Το Connections γι’ αυτό το σκοπό χρησιμοποιεί τρεις αλγορίθμους: τον Basic-BFS και δύο ακόμα αλγορίθμους, επέκταση του Basic-BFS, οι οποίοι βασίζονται στους αλγορίθμους αναζήτησης στο διαδίκτυο, HITS και PageRank:

- Basic-BFS: χρησιμοποιεί τα αποτελέσματα από την αναζήτηση με βάση το περιεχόμενο, ώστε να γίνει και η κατάταξη στα αποτελέσματα που προκύπτουν από τις συσχετίσεις των αρχείων. Συλλέγει πληροφορίες από το γράφο και βρίσκει το βάρος που προκύπτει για κάθε αρχείο. Το βάρος του αρχείου είναι το άθροισμα των βαρών των ακμών που το συνδέουν με τα άλλα αρχεία συν το βάρος που έχουν από την ανάλυση περιεχομένου. Έτσι, εάν ένα αρχείο χρησιμοποιείται σπάνια σε συνδυασμό με τα αρχεία που βρέθηκαν από την ανάλυση περιεχομένου, τότε θα λάβει χαμηλό βάρος, αντίθετα θα λάβει υψηλό βάρος.

- HITS: Ο αλγόριθμος HITS υπολογίζει δύο τιμές ενός κόμβου του γραφήματος: την αξία του περιεχομένου του αρχείου που αντιπροσωπεύει ένας κόμβος, εξετάζοντας τον αριθμό των

εισερχόμενων προς αυτόν ακμών (Authority) και την αξία των συνδέσμων του με άλλους κόμβους, εξετάζοντας τον εξερχόμενο αριθμό ακμών. (Hub).

- PageRank: μπορεί να χρησιμοποιηθεί σε συνδυασμό με τον Basic-BFS με δύο διαφορετικούς τρόπους, είτε εφαρμόζοντας πρώτα τον PageRank στα αποτελέσματα από την ανάλυση περιεχομένου και μετά τον Basic-BFS στα τελικά αποτελέσματα, είτε τον Basic-BFS στα αποτελέσματα της ανάλυσης περιεχομένου και τον PageRank στα τελικά αποτελέσματα. Επίσης, είναι δυνατόν να γίνει η χρήση του PageRank για την κατάταξη των τελικών αποτελεσμάτων αγνοώντας την κατάταξη των αποτελεσμάτων της ανάλυσης περιεχομένου.

2.3.2 – Αιτιότητα (Causality)

Η χρήση της προσωρινής τοπικότητας στο εργαλείο Connections βελτιώνει την αναζήτηση σε σύγκριση με εργαλεία που χρησιμοποιούν μόνο ανάλυση περιεχομένου. Ωστόσο, πέρα από τα πολλά πλεονεκτήματα που εισάγει η προσωρινή τοπικότητα έχει ως βασικό μειονέκτημα τη μεγάλη πιθανότητα να καταγραφούν πλαστές συσχετίσεις μεταξύ των αρχείων.

Με σκοπό να αντιμετωπιστεί το μειονέκτημα της προσωρινής τοπικότητας και παράλληλα να χρησιμοποιηθούν όλα τα πλεονεκτήματα που έχει, προτάθηκε μια εναλλακτική μέθοδος για την ανάλυση και την καταγραφή των συσχετίσεων μεταξύ των αρχείων, η αιτιότητα. Πρόκειται για μια μέθοδο η οποία για να καταγράψει τις συσχετίσεις μεταξύ των αρχείων χρησιμοποιεί τη ροή δεδομένων που πραγματοποιείται στο σύστημα μεταξύ των εφαρμογών και των αρχείων. Η μέθοδος αναγνωρίζει με μεγαλύτερη ακρίβεια τις συσχετίσεις και μειώνεις τα πλαστά αποτελέσματα που παράγει η προσωρινή τοπικότητα.

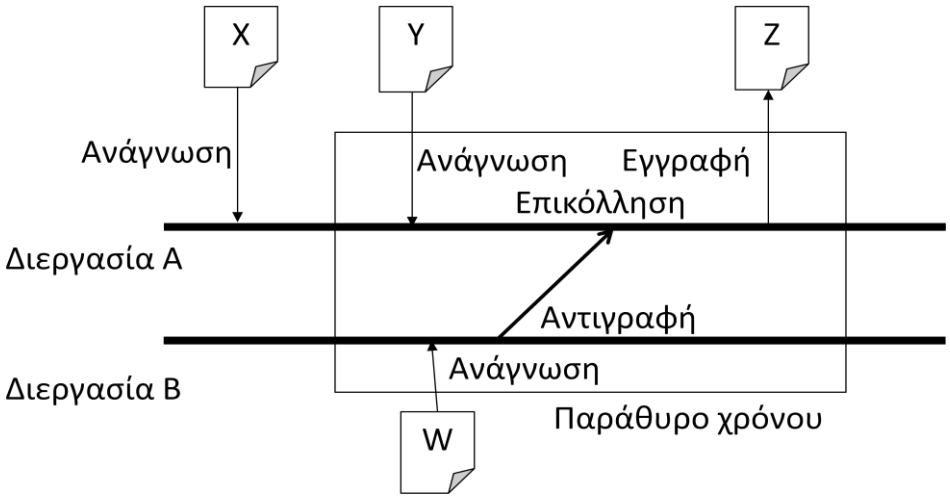
Αρχικά το εργαλείο τρέχει μια αναζήτηση περιεχομένου χρησιμοποιώντας τις λέξεις κλειδιά που έδωσε ο χρήστης και στη συνέχεια χρησιμοποιεί το διάγραμμα συσχετίσεων ώστε να ανακατατάξει τα αποτελέσματα της αναζήτησης περιεχομένου και να τα εμπλουτίσει με περισσότερα. Αυτά τα αποτελέσματα θα επιστραφούν στον χρήστη. Επίσης, τρέχουν στο σύστημα διάφορες εργασίες παρασκηνίου οι οποίες τοποθετούν σε ευρετήριο τα περιεχόμενα των αρχείων που χρησιμοποιούνται και δημιουργούν το διάγραμμα των συσχετίσεων.

Η δημιουργία του διαγράμματος των συσχετίσεων γίνεται καταγράφοντας τις συγγενείς συσχετίσεις που υπάρχουν μεταξύ των αρχείων που χρησιμοποιήθηκαν από τον χρήστη. Μια συγγενής συσχέτιση, αρχείο1 → αρχείο2, υποδεικνύει ότι το αρχείο αρχείο1 είναι πρόγονος του αρχείου αρχείο2. Η καταγραφή των συγγενών συσχετίσεων γίνεται με δύο μεθόδους: με χρήση της

προσωρινής τοπικότητας και με χρήση της αιτιότητας. Έχοντας αναλύσει παραπάνω τη μέθοδο της προσωρινής τοπικότητας, θα εστιάσουμε στη συνέχεια στην αιτιότητα.

Όπως είδαμε, η αιτιότητα χρησιμοποιεί τη ροή των δεδομένων μεταξύ των εφαρμογών για να καταγράψει συγγενής συσχετίσεις μεταξύ αρχείων. Ουσιαστικά ο αλγόριθμος καταγράφει τη ροή των δεδομένων κατά την είσοδο σε μια διεργασία, ώστε να ανακαλύψει ποιες έξοδοι σχετίζονται με τις αντίστοιχες εισόδους στη διεργασία. Κάθε φορά που πραγματοποιείται μια κλήση εγγραφής δημιουργούνται οι εξής συσχετίσεις:

- Όλα τα προηγούμενα αρχεία που διαβάστηκαν από μια συγκεκριμένη διεργασία συσχετίζονται με το αρχείο που γράφεται.
- Ο αλγόριθμος καταγράφει τη ροή δεδομένων μεταξύ των διεργασιών δημιουργώντας επιπρόσθετες συσχετίσεις.



Σχήμα 2.2 – Τρόπος συσχέτισης των αρχείων. Τα αρχεία Y και W συνδέονται με το αρχείο Z, ενώ το X όχι αφού δεν βρίσκεται στο χρονικό όριο που έχει τεθεί.

Ο αλγόριθμος που βασίζεται στην αιτιότητα καταγράφει λιγότερες συσχετίσεις από τον αντίστοιχο που χρησιμοποιεί την προσωρινή τοπικότητα, μειώνοντας την ύπαρξη πλαστών συσχετίσεων. Έτσι, αποφεύγεται η καταγραφή αρχείων που χρησιμοποιούνται από διαφορετικές μη συσχετιζόμενες διεργασίες που εκτελούνται ταυτόχρονα. Για παράδειγμα, όταν ο χρήστης συντάσσει κάποιο κείμενο και ταυτόχρονα ακούει μουσική από ένα πρόγραμμα αναπαραγωγής, το έγγραφο και το τραγούδι που ακούει δε συσχετίζονται μεταξύ τους.

Η χρήση της αιτιότητας εκτός από τα πλεονεκτήματα που αναφέρθηκαν, δε μπορεί να καταγράψει καταστάσεις όταν η ροή των δεδομένων πραγματοποιείται με κρυφά κανάλια. Για παράδειγμα όταν δεν υπάρχουν συμφραζόμενα, επειδή ο χρήστης διαβάζει μια λέξη από κάποιο έγγραφο και την πληκτρολογεί σε ένα άλλο.

2.4 – Περίληψη

Η εκθετική αύξηση των πληροφοριών που διακινούνται στο διαδίκτυο σε συνδυασμό με την αύξηση της χωρητικότητας των αποθηκευτικών μέσων έχει ως συνέπεια την αποθήκευση μεγάλου όγκου δεδομένων στους προσωπικούς υπολογιστές και στους διακομιστές αποθήκευσης. Ωστόσο, ο μεγάλος όγκος αποθηκευμένων δεδομένων δυσχεραίνει την οργάνωση και την εύρεση των αρχείων. Στο διαδίκτυο το πρόβλημα της αναζήτησης λύνεται αποτελεσματικά με τη χρήση αλγορίθμων όπως ο PageRank και ο HITS, οι οποίοι εκμεταλλεύονται την ύπαρξη υπερσυνδέσεων μεταξύ των ιστοσελίδων. Από την άλλη μεριά, στο χώρο των προσωπικών υπολογιστών και των διακομιστών αποθήκευσης οι παραπάνω αλγόριθμοι δε μπορούν να εφαρμοστούν αποτελεσματικά γιατί δεν υπάρχουν υπερσυνδέσεις μεταξύ των αρχείων.

Για την αναζήτηση αρχείων σε έναν προσωπικό υπολογιστή ή διακομιστή αρχείων αρχικά χρησιμοποιήθηκε η μέθοδος της ανάλυσης περιεχομένου. Η μέθοδος αυτή βασίζεται στη δημιουργία ευρετηρίων που περιλαμβάνουν λέξεις κλειδιά οι οποίες χαρακτηρίζουν το κάθε αρχείο. Υλοποιήθηκαν διάφορα συστήματα που χρησιμοποιούν αυτή τη μέθοδο, όπως το Google Desktop, το Spotlight, και το Yahoo! Desktop Search.

Αποδείχθηκε, όμως, ότι η μέθοδος της ανάλυσης περιεχομένου δε λύνει αποτελεσματικά το πρόβλημα της αναζήτησης. Για το λόγο αυτό προτάθηκαν νέες μέθοδοι οι οποίες προσπαθούν να χρησιμοποιήσουν τα συμφραζόμενα ενός αρχείου για να δημιουργήσουν συνδέσεις μεταξύ των αρχείων που χρησιμοποιεί ο χρήστης. Αφού δημιουργηθούν αυτές οι συνδέσεις και αποθηκευτούν σε έναν γράφο είναι δυνατόν να εκτελεστούν στη συνέχεια αλγόριθμοι όπως ο PageRank και ο HITS.

Στην παρούσα εργασία, επανεξετάζουμε τις μεθόδους αναζήτησης αρχείων στους προσωπικούς υπολογιστές και στους διακομιστές αποθήκευσης με σκοπό να αυξήσουμε την απόδοσή τους και ταυτόχρονα να μειώσουμε την επιβάρυνση που επιφέρουν στο σύστημα.

Κεφάλαιο 3

Οδηγοί συσκευών

Στο κεφάλαιο αυτό αρχικά παρουσιάζουμε την αρχιτεκτονική διαστρωμάτωσης των Windows XP. Στη συνέχεια, εισάγουμε κάποιες γενικές έννοιες για τους οδηγούς συσκευών και αναλύουμε ορισμένα προγραμματιστικά θέματα, όπως ο συγχρονισμός και η χρήση της μνήμης.

3.1 – Εισαγωγή στην αρχιτεκτονική των Windows XP

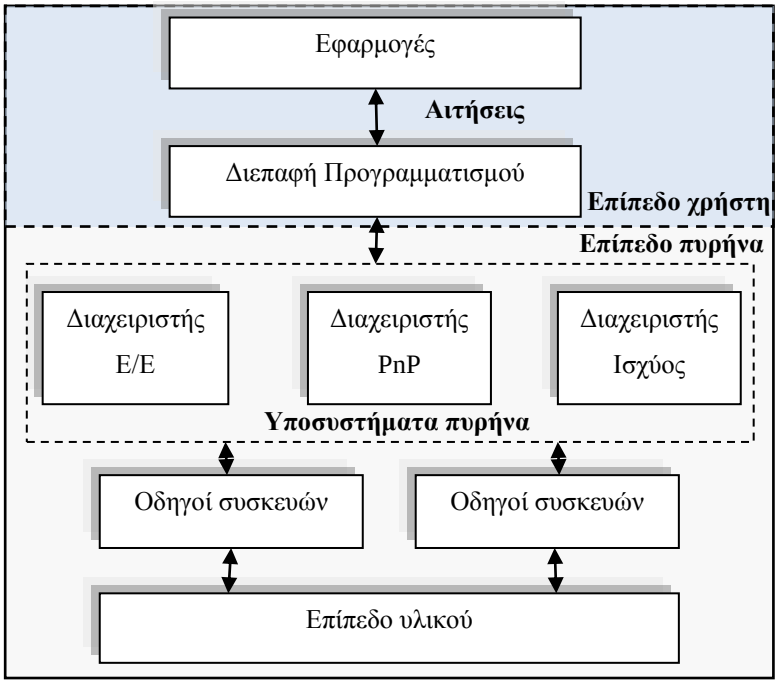
Τα Windows XP περιλαμβάνουν μια αρχιτεκτονική διαστρωμάτωσης η οποία αποτελείται από δύο μέρη, το επίπεδο χρήστη και το επίπεδο πυρήνα. Στο επίπεδο χρήστη εκτελούνται οι εφαρμογές και οι υπηρεσίες χρήστη οι οποίες δεν έχουν απευθείας πρόσβαση στα συστατικά του πυρήνα. Το επίπεδο χρήστη μπορεί να επικοινωνήσει με τον πυρήνα διαμέσου της διεπαφής προγραμματισμού εφαρμογών (Windows API). Η διεπαφή χρησιμοποιεί συστατικά των Windows, όπως το Ntdll.dll για να προωθήσει την αίτηση της εφαρμογής χρήστη στο κατάλληλο υποσύστημα του επιπέδου πυρήνα. Το επίπεδο πυρήνα έχει απευθείας πρόσβαση στους πόρους του υλικού και του συστήματος και εκτελεί κώδικα σε μια προστατευόμενη περιοχή της μνήμης. Επίσης, ελέγχει την πρόσβαση στον χρονοπρογραμματισμό των διεργασιών, στη διαχείριση της μνήμης και στη λειτουργία του υλικού. Το επίπεδο πυρήνα αποτρέπει τις εφαρμογές και τις υπηρεσίες του επιπέδου χρήστη να έχουν πρόσβαση σε κρίσιμα σημεία του λειτουργικού συστήματος.

Το επίπεδο πυρήνα αποτελείται από υποσυστήματα τα οποία επιτρέπουν την επικοινωνία των οδηγών συσκευών με το υπόλοιπο σύστημα. Τα τρία κυριότερα υποσυστήματα που χρησιμοποιούνται από τους οδηγούς συσκευών είναι ο διαχειριστής Εισόδου/ Εξόδου (I/O Manager), ο διαχειριστής σύνδεσης και χρήσης (PnP Manager) και ο διαχειριστής ισχύος (Power Manager). Στη συνέχεια θα περιγράψουμε συνοπτικά τα παραπάνω υποσυστήματα.

- Ο διαχειριστής Ε/Ε είναι ο πυρήνας του συστήματος Ε/Ε. Συνδέει εφαρμογές και υπηρεσίες χρήστη με εικονικές, λογικές και φυσικές συσκευές του συστήματος και καθορίζει έναν τρόπο για την παράδοση των αιτήσεων Ε/Ε στις συσκευές. Για την επικοινωνία αυτή ο διαχειριστής Ε/Ε καλεί μια από τις ρουτίνες για γρήγορη Ε/Ε του οδηγού συσκευής. Αν τα δεδομένα δε βρίσκονται στην κρυφή μνήμη, χρησιμοποιεί το πακέτο IRP (I/O Request Packet), το οποίο προωθεί στον κατάλληλο οδηγό. Το πακέτο IRP είναι ουσιαστικά μια δομή δεδομένων η οποία περιέχει τις απαραίτητες πληροφορίες που χαρακτηρίζουν μια αίτηση Ε/Ε. Μόλις ο οδηγός λάβει το πακέτο IRP, αφού εκτελέσει την κατάλληλη εργασία, το επιστρέφει στον διαχειριστή Ε/Ε για ολοκλήρωση ή για τη μεταβίβασή του σε κάποιον άλλο οδηγό για περαιτέρω επεξεργασία.

- Ο διαχειριστής σύνδεσης και χρήσης παρέχει υποστήριξη για την αυτόματη προσαρμογή του συστήματος σε αλλαγές υλικού. Επίσης, είναι υπεύθυνος για τη φόρτωση, τη ρύθμιση και την απεγκατάσταση του οδηγού μιας συσκευής υλικού.

- Ο διαχειριστής ισχύος διαχειρίζεται την κατανάλωση ενέργειας και τη μετάβαση του συστήματος σε καταστάσεις εξοικονόμησης ενέργειας (sleep mode).



Σχήμα 3.1 - Η αρχιτεκτονική των Windows XP. Οι εφαρμογές χρησιμοποιούν τη διεπαφή προγραμματισμού εφαρμογών για να επικοινωνήσουν με τα υποσυστήματα του πυρήνα. Οι οδηγοί συσκευών οργανώνονται σε στοίβες και μία στοίβα μπορεί να περιέχει περισσότερους από έναν οδηγό.

3.2 – Η αρχιτεκτονική των οδηγών συσκευών

Ένας οδηγός (driver) είναι ένα σύνολο από υπορουτίνες τις οποίες καλεί το λειτουργικό σύστημα για να επικοινωνήσει με το υλικό του υπολογιστή. Ο διαχειριστής Ε/Ε (I/O Manager) του λειτουργικού συστήματος χρησιμοποιεί μια δομή δεδομένων που ονομάζεται αντικείμενο οδηγού (driver object) για να αναπαραστήσει κάθε οδηγό που υπάρχει στο σύστημα. Τα Windows XP οργανώνουν τους οδηγούς συσκευών σε στοίβες επιτρέποντας περισσότερους από έναν οδηγούς να ανήκουν σε μια στοίβα. Κάθε οδηγός σε μια στοίβα επεξεργάζεται το τμήμα της αίτησης για το οποίο είναι υπεύθυνος. Αν η αίτηση δεν ολοκληρωθεί τότε μεταβιβάζεται στον αμέσως επόμενο οδηγό στη στοίβα για περαιτέρω επεξεργασία.

Το πρότυπο με το οποίο θα ασχοληθούμε στην παρούσα εργασία ονομάζεται Win32 Driver Model (WDM). Όλα όσα θα αναφέρουμε στη συνέχεια θα ισχύουν για οδηγούς συσκευών που ακολουθούν αυτό το πρότυπο. Οι οδηγοί που ανήκουν στο πρότυπο αυτό διαιρούνται σε τρεις κατηγορίες:

- Οδηγοί διαύλου (Bus Drivers). Είναι υπεύθυνοι για τη διαχείριση ενός λογικού ή φυσικού δίαυλου.

- Οδηγοί υπηρεσίας/ συσκευών (Function Drivers). Διαχειρίζονται μια συγκεκριμένη συσκευή υλικού και παρέχουν στο λειτουργικό σύστημα μια διεπαφή για τη χρήση της.

- Οδηγοί φίλτρα (Filter Drivers). Βρίσκονται ένα επίπεδο πάνω ή κάτω από τους οδηγούς υπηρεσίας και είναι υπεύθυνοι για την βελτίωση ή την τροποποίηση της συμπεριφοράς της συσκευής. Σε επόμενη παράγραφο θα περιγράψουμε αναλυτικότερα τους οδηγούς αυτούς.

3.2.1 – Η δομή ενός οδηγού συσκευής

Οι οδηγοί συσκευών αποτελούνται από ρουτίνες που καλούνται για να επεξεργαστούν διάφορα στάδια μιας αίτησης Ε/Ε. Οι σημαντικότερες ρουτίνες είναι οι εξής:

- Ρουτίνα αρχικοποίησης (*DriverEntry*): Εκτελείται από τον διαχειριστή Ε/Ε όταν φορτώνεται ο οδηγός στο σύστημα. Περιλαμβάνει την αρχικοποίηση των δομών δεδομένων και των καθολικών μεταβλητών του οδηγού.

- Ρουτίνα εισαγωγής συσκευής (*AddDevice*): Η ρουτίνα αυτή περιέχεται σε οδηγούς που υποστηρίζουν τη λειτουργία σύνδεσης και χρήσης (Plug and Play). Χρησιμοποιείται από τον διαχειριστή PnP ώστε να στείλει μια ειδοποίηση στο λειτουργικό σύστημα όταν τοποθετηθεί η συσκευή για την οποία είναι υπεύθυνος ο συγκεκριμένος οδηγός.

- Ρουτίνες αποστολής αίτησης (Dispatch Routines): Πρόκειται για τις κύριες λειτουργίες που παρέχει ένας οδηγός, όπως εγγραφή και ανάγνωση.

- Ρουτίνες ελέγχου E/E (I/O Control Routines): Οι ρουτίνες αυτές είναι υπεύθυνες για τον έλεγχο της μεταφοράς δεδομένων από και προς τον οδηγό.

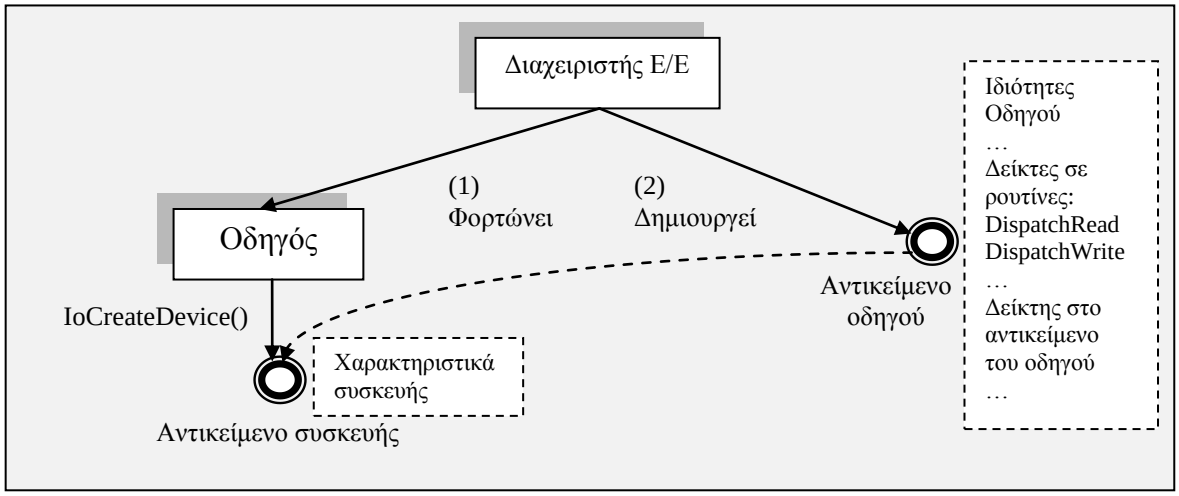
Βασικές Ρουτίνες	**Ρουτίνες ελέγχου E/E**	**Ρουτίνες αποστολής αίτησης**
DriverEntry	StartIo	DisaptchPnp
AddDevice	FastIoControl	DispatchPower
DriverUnload	…	DispatchWmi
…		DispatchRead
		DispatchWrite
		…

Πίνακας 3.1 - Οι βασικές ρουτίνες ενός οδηγού συσκευής

3.2.2 – Το αντικείμενο του οδηγού

Όπως αναφέραμε, κάθε οδηγός αναπαρίσταται στο σύστημα ως μια δομή που ονομάζεται αντικείμενο οδηγού. Όταν ο οδηγός φορτώνεται στο σύστημα, το αντικείμενο οδηγού δημιουργείται και αρχικοποιείται από τον διαχειριστή E/E. Το αντικείμενο οδηγού περιλαμβάνει τις διευθύνσεις των ρουτινών αποστολής αιτήσεων. Από το σημείο αυτό και μετά, όταν ο διαχειριστής E/E θελήσει να επικοινωνήσει με μια συσκευή, αναζητά το αντικείμενο του οδηγού που είναι υπεύθυνος γι' αυτή και καλεί κάποια από τις ρουτίνες που περιλαμβάνει.

Τα Windows XP ορίζουν και το αντικείμενο συσκευής (device object) ως μια δομή που αναπαριστά μοναδικά μια συσκευή του συστήματος. Η δομή αυτή περιέχει χαρακτηριστικά τις συσκευής και δημιουργείται από τον αντίστοιχο οδηγό της συσκευής όταν αυτός φορτώνεται στο σύστημα.



Σχήμα 3.2 - Δημιουργία αντικειμένου οδηγού και συσκευής. Ο διαχειριστής Ε/Ε κατά τη φόρτωση ενός οδηγού δημιουργεί μια δομή που το περιγράφει. Αντίστοιχα, ο οδηγός, κατά τη φόρτωσή του, δημιουργεί το αντικείμενο συσκευής που περιγράφει τα χαρακτηριστικά της συσκευής την οποία ελέγχει.

3.3 – Διαχείριση αιτήσεων Ε/Ε

Το μοντέλο Ε/Ε των Windows ακολουθεί την αρχιτεκτονική διαστρωμάτωσης που περιγράψαμε στην ενότητα 3.1. Μια αίτηση Ε/Ε ξεκινάει από το ανώτερο επίπεδο της διαστρωμάτωσης, δηλαδή το επίπεδο εφαρμογών και μεταβαίνει προς τα κάτω. Με τη βοήθεια του διαχειριστή Ε/Ε, τελικά φτάνει στον κατάλληλο οδηγό της συσκευής στην οποία αναφέρεται.

Οι εφαρμογές συνήθως παραθέτουν στο σύστημα σύγχρονες αιτήσεις Ε/Ε. Για παράδειγμα, όταν μια εφαρμογή ζητάει από κάποιον οδηγό συσκευής να διαβάσει δεδομένα από τη συσκευή, περιμένει μέχρι να της επιστραφούν τα δεδομένα που ζήτησε και στη συνέχεια συνεχίζει την επεξεργασία ή ζητάει επιπλέον δεδομένα. Στην περίπτωση αυτή, ο διαχειριστής Ε/Ε περιμένει μέχρι ο οδηγός να ολοκληρώσει την ανάγνωση πριν επιστρέψει τα δεδομένα στην εφαρμογή.

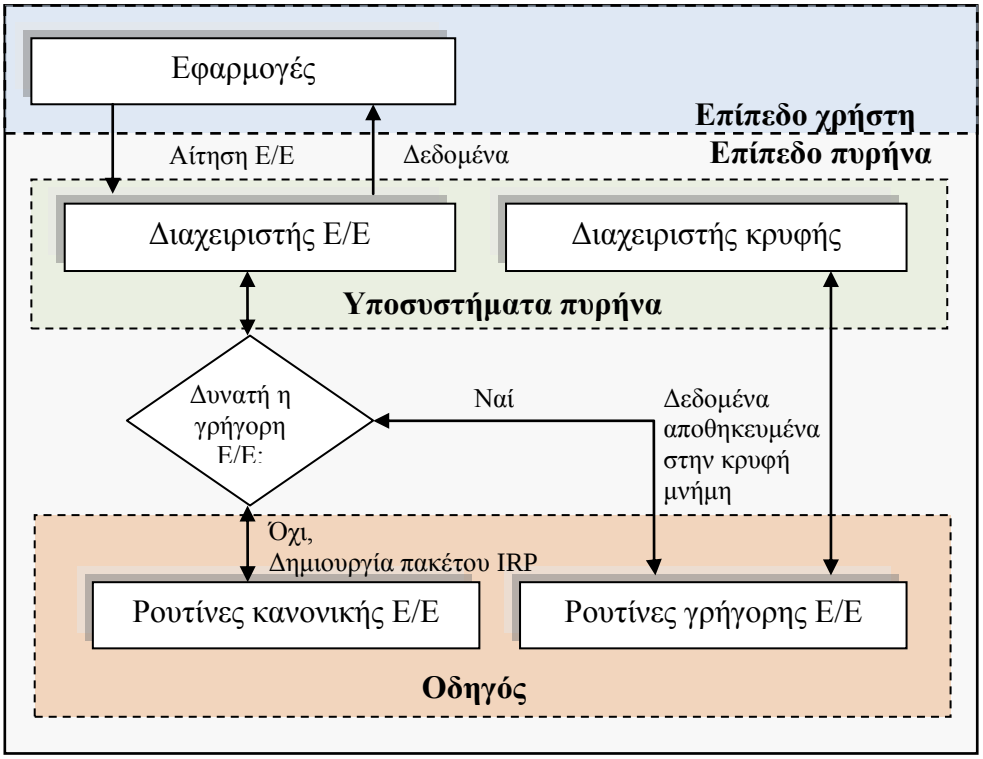
Στους οδηγούς συσκευών η λειτουργία Ε/Ε γίνεται ασύγχρονα. Για παράδειγμα, όταν ένας οδηγός δέχεται μια αίτηση ανάγνωσης από μια εφαρμογή, ο οδηγός επιστρέφει αμέσως τα δεδομένα που ζητήθηκαν εάν αυτά είναι διαθέσιμα. Σε αντίθετη περίπτωση, εκτελεί τις απαραίτητες ενέργειες ώστε να αποκτήσει τα δεδομένα και ενημερώνει τον διαχειριστή Ε/Ε ότι η αίτηση είναι εκκρεμής.

Στο μεσοδιάστημα, μέχρι να αποκτήσει τα δεδομένα, ο οδηγός μπορεί να επεξεργαστεί άλλες αιτήσεις.

Πολλές φορές, οι εφαρμογές μπορούν να χρησιμοποιήσουν και αυτές ασύγχρονη Ε/Ε προκειμένου να ωφεληθούν από τον τρόπο που χειρίζονται τις αιτήσεις Ε/Ε οι οδηγοί. Έτσι, μόλις ο οδηγός ενημερώσει τον διαχειριστή Ε/Ε ότι η αίτηση είναι εκκρεμής, ο διαχειριστής Ε/Ε ενημερώνει με τη σειρά του την εφαρμογή. Μόλις τα δεδομένα γίνουν διαθέσιμα, ο οδηγός ενημερώνει τον διαχειριστή Ε/Ε, ο οποίος πληροφορεί το νήμα της εφαρμογής που είχε πραγματοποιήσει την αίτηση ότι τα δεδομένα είναι διαθέσιμα.

3.3.1 – Γρήγορη Ε/Ε

Όταν το σύστημα χρησιμοποιεί κρυφή μνήμη, η κύρια μέθοδος για τη μεταβίβαση μιας αίτησης Ε/ Ε από τον διαχειριστή Ε/Ε σε έναν οδηγό συσκευής, είναι η γρήγορη Ε/ Ε (Fast I/O). Η γρήγορη Ε/ Ε είναι μια σύγχρονη λειτουργία (synchronous operation) κατά την οποία ο οδηγός συσκευής διαβάζει τα δεδομένα απευθείας από την κρυφή μνήμη και τα επιστρέφει στον διαχειριστή Ε/ Ε.

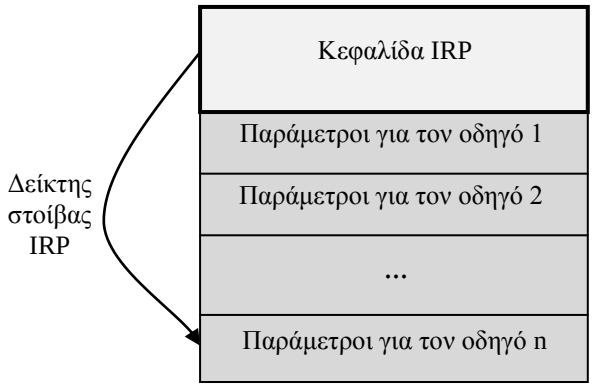


Σχήμα 3.3 - Αίτηση Ε/Ε από μια εφαρμογή. Ο διαχειριστής Ε/Ε αρχικά καλεί τις ρουτίνες γρήγορης Ε/Ε του οδηγού. Αν τα δεδομένα υπάρχουν στην κρυφή μνήμη, η αίτηση είναι επιτυχής και του επιστρέφονται τα δεδομένα. Διαφορετικά, στέλνει στον οδηγό ένα πακέτο αίτησης IRP.

Όταν ο διαχειριστής Ε/Ε δέχεται μια αίτηση Ε/Ε από κάποια εφαρμογή, εκτελεί την αντίστοιχη ρουτίνα για γρήγορη Ε/Ε του κατάλληλου οδηγού. Εάν τα δεδομένα υπάρχουν στην κρυφή μνήμη, τότε ο οδηγός μπορεί να ολοκληρώσει την αίτηση και επιστρέφει την τιμή «Αληθής» στον διαχειριστή Ε/Ε. Σε αντίθετη περίπτωση, ο οδηγός επιστρέφει την τιμή «Ψευδής». Τότε ο διαχειριστής Ε/Ε δημιουργεί ένα πακέτο IRP που περιγράφει την αίτηση και καλεί την κατάλληλη ρουτίνα κανονικής Ε/Ε του οδηγού μεταβιβάζοντας του το πακέτο IRP.

3.3.2 – Πακέτα IRP

Όταν το σύστημα δε χρησιμοποιεί κρυφή μνήμη, ή όταν η λειτουργία Ε/Ε είναι ασύγχρονη, η κύρια μέθοδος για την επικοινωνία του διαχειριστή Ε/Ε και των οδηγών συσκευών είναι τα πακέτα IRP (I/O Request Packet). Τα πακέτα IRP περιέχουν πληροφορίες για μια αίτηση Ε/Ε. Κάθε πακέτο IRP αποτελείται από δύο μέρη: την κεφαλίδα και ένα πίνακα με παραμέτρους για την αίτηση. Η κεφαλίδα περιλαμβάνει πληροφορίες που χαρακτηρίζουν την αίτηση, όπως ο τύπος, το μέγεθός της και ο προσωρινός χώρος μνήμης που θα χρησιμοποιηθεί για τη μεταφορά των δεδομένων. Ο πίνακας παραμέτρων του πακέτου IRP περιέχει τον κωδικό των ρουτινών των οδηγών που θα επεξεργαστούν την αίτηση, καθώς και διάφορες παραμέτρους που θα περαστούν στις ρουτίνες. Το μέγεθος του πίνακα εξαρτάται από τον αριθμό των οδηγών που απαιτούνται για την ολοκλήρωση της αίτησης. Να σημειώσουμε ότι ο οδηγός κατά τη φόρτωσή του ορίζει τους κωδικούς των συναρτήσεων που προσφέρει για τη διαχείριση Ε/Ε στο αντικείμενό του, έτσι ο διαχειριστής Ε/Ε μπορεί εύκολα να καλεί τη ρουτίνα που χρειάζεται.



Σχήμα 3.4 - Δομή του πακέτου IRP

Η ανταλλαγή των δεδομένων μεταξύ των εφαρμογών χρήστη και των οδηγών συσκευών μπορεί να γίνει με τρεις τρόπους:

- Με ενδιάμεσο χώρο αποθήκευσης στη μνήμη (Buffered I/O). Ο διαχειριστής Ε/Ε δημιουργεί έναν προσωρινό χώρο στη μνήμη στον οποίο αντιγράφει τα δεδομένα που περιέχει ο προσωρινός χώρος μνήμης της εφαρμογής χρήστη. Στη συνέχεια στέλνει στον οδηγό έναν δείκτη στον δικό του προσωρινό χώρο και όχι στον προσωρινό χώρο της εφαρμογής.

- Απευθείας (Direct I/O): Ο διαχειριστής Ε/Ε διασφαλίζει ότι ο προσωρινός χώρος μνήμης της εφαρμογής βρίσκεται στην κύρια μνήμη και στέλνει στον οδηγό μια λίστα περιγραφέων μνήμης (MDL) που τον περιγράφουν.

- Ούτε με ενδιάμεσο χώρο αποθήκευσης στη μνήμη ούτε απευθείας (Neither buffered nor direct I/O). Το λειτουργικό σύστημα στέλνει στον οδηγό συσκευής το μέγεθος και τη διεύθυνση του προσωρινού χώρου αποθήκευσης στη μνήμη της εφαρμογής και ο οδηγός τον διαχειρίζεται απευθείας. Η διαχείριση αυτή πρέπει να γίνει με προσοχή διότι το λειτουργικό σύστημα δε διασφαλίζει ότι η διεύθυνση του προσωρινού χώρου μνήμης της εφαρμογής είναι έγκυρη.

Πολλές φορές για να εξυπηρετηθεί μια αίτηση Ε/Ε απαιτείται η επεξεργασία της από διαφορετικούς οδηγούς. Όταν απαιτείται η μεταβίβαση ενός πακέτου IRP από έναν οδηγό σε άλλο που βρίσκεται χαμηλότερα στη στοίβα, τότε ο οδηγός που κατέχει το πακέτο IRP εκτελεί τρία βήματα ώστε να το μεταβιβάσει στον χαμηλότερο οδηγό:

1. Καθορίζει τις παραμέτρους που θα χρησιμοποιήσει ο επόμενος οδηγός.

2. Θέτει μια ρουτίνα ολοκλήρωσης (IoCompletion) αν η αίτηση χρειάζεται περαιτέρω επεξεργασία από τον συγκεκριμένο οδηγό πριν ολοκληρωθεί.

3. Μεταβιβάζει το πακέτο στον επόμενο οδηγό καλώντας την συνάρτηση IoCallDriver.

Από τη στιγμή που ο οδηγός μεταβιβάζει το πακέτο IRP σε έναν άλλο οδηγό δε το κατέχει πλέον και δε μπορεί να το επεξεργαστεί πια.

Μόλις ένας οδηγός ολοκληρώσει την επεξεργασία μιας αίτησης, καλεί τη ρουτίνα *IoCompleteRequest*. Η ρουτίνα αυτή μεταβιβάζει το δείκτη στοίβας ένα επίπεδο πάνω. Στο σημείο αυτό, αν ο προηγούμενος οδηγός που κατείχε το πακέτο IRP είχε θέσει μια ρουτίνα ολοκλήρωσης, αυτή καλείται από το διαχειριστή Ε/Ε και το πακέτο μεταβιβάζεται σε αυτόν.

Σχήμα 3.5 - Ολοκλήρωση αίτησης – δείκτης στοίβας. Το σχήμα δείχνει την θέση στοίβας μόλις ο οδηγός C ολοκληρώσει την επεξεργασία της αίτησης. Ο δείκτης στοίβας δείχνει πλέον στις παραμέτρους του προηγούμενου οδηγού.

3.4 – Συγχρονισμός

Τα Windows XP είναι ένα πολυπρογραμματιστικό σύστημα το οποίο είναι δυνατό να τρέχει σε ένα πολυεπεξεργαστικό περιβάλλον. Για το λόγο αυτό ένας οδηγός είναι απαραίτητο να χρησιμοποιεί τεχνικές συγχρονισμού για τα ευαίσθητα δεδομένα του, όπως οι καθολικές μεταβλητές.

Ένας μηχανισμός που παρέχεται από το λειτουργικό σύστημα και λύνει μερικώς το πρόβλημα συγχρονισμού είναι οι διακοπές (interrupts). Η μέθοδος αυτή είναι πολύ χρήσιμη για τον καθορισμό της διεργασίας που θα πρέπει να τρέξει μια συγκεκριμένη χρονική στιγμή σε έναν επεξεργαστή. Τα Windows XP καθορίζουν επίπεδα προτεραιότητας, τα οποία ονομάζονται επίπεδα αίτησης διακοπών (interrupt request level) και συμβολίζονται ως IRQLs. Το σύστημα χρησιμοποιεί τις διαφορετικές τιμές των IRQL προκειμένου να διασφαλίσει ότι οι πιο κρίσιμες διεργασίες θα τρέξουν πρώτες. Να σημειωθεί ότι διεργασίες με υψηλότερο IRQL έχουν προτεραιότητα. Κάθε χρονική στιγμή, ο επεξεργαστής τρέχει σε ένα συγκεκριμένο επίπεδο IRQL. Αν συμβεί μια διακοπή με επίπεδο μεγαλύτερο από το επίπεδο στο οποίο τρέχει ο επεξεργαστής, τότε ο επεξεργαστής αυξάνει το επίπεδο IRQL στο οποίο τρέχει, τοποθετεί τη διεργασία που έτρεχε σε αναμονή και εκτελεί τη νέα με το υψηλότερο IRQL. Επειδή οι τιμές των IRQL εξαρτώνται από την αρχιτεκτονική του κάθε επεξεργαστή είναι χρήσιμο να αναφερόμαστε σε αυτές με τη χρήση ονομάτων – σταθερών. Οι τιμές IRQL που μπορεί να χρησιμοποιήσει ένας οδηγός είναι (από τη μικρότερη στη μεγαλύτερη):

- PASSIVE_LEVEL: Πρόκειται για την προκαθορισμένη τιμή στην οποία τρέχουν οι εφαρμογές χρήστη και οι μικρής προτεραιότητας ρουτίνες των οδηγών συσκευών.

- DISPATCH_LEVEL: Σε αυτό το επίπεδο τρέχουν οι υψηλής προτεραιότητας ρουτίνες των οδηγών συσκευών. Οι ρουτίνες που τρέχουν σε αυτό το επίπεδο έχουν περιορισμένη πρόσβαση σε στοιχεία του συστήματος.

- DIRQL: Είναι το μεγαλύτερο επίπεδο προτεραιότητας που μπορεί να έχει μια ρουτίνα ενός οδηγού.

Ρουτίνα Συγχρονισμού	**Περιγραφή**
ExAcquireResourceSharedLite()	Αποκτά το αντικείμενο συγχρονισμού για κοινόχρηστη χρήση.
ExAcquireResourceExclusiveLite()	Αποκτά το αντικείμενο συγχρονισμού για αποκλειστική χρήση.
ExReleaseResourceLite()	Απελευθερώνει ένα αντικείμενο αμοιβαίου αποκλεισμού.
KeEnterCriticalRegion()	Πριν κληθούν οι ρουτίνες που αναφέρθηκαν πρέπει να απενεργοποιηθεί η ασύγχρονη κλήση διαδικασιών (APC) με τη χρήση της ρουτίνας αυτής.
KeLeaveCriticalRegion()	Ενεργοποιείται η ασύγχρονη κλήση διαδικασιών.
KeAcquireSpinLock()	Αποκτά το αντικείμενο αμοιβαίου αποκλεισμού και αυξάνει το επίπεδο IRQL.
KeReleaseSpinLock()	Απελευθερώνει το αντικείμενο αμοιβαίου αποκλεισμού και μειώνει το επίπεδο IRQL.

Πίνακας 3.2 - Βασικές ρουτίνες συγχρονισμού που παρέχονται από το λειτουργικό σύστημα.

Αν και τα IRQL λύνουν το πρόβλημα του συγχρονισμού σε ένα περιβάλλον με έναν επεξεργαστή, είναι δυνατόν σε ένα πολυεπεξεργαστικό περιβάλλον να τρέχουν ταυτόχρονα ρουτίνες με διαφορετική τιμή IRQL. Το γεγονός αυτό μπορεί να οδηγήσει σε αδιέξοδα και κάνει αναγκαία

την ύπαρξη διαφορετικών μεθόδων συγχρονισμού. Για το λόγο αυτό, το λειτουργικό σύστημα παρέχει αντικείμενα όπως οι σημαφόροι (semaphore) και τα αντικείμενα αμοιβαίου αποκλεισμού (mutexes - mutual exclusion).

3.5 – Μνήμη

Οι οδηγοί συσκευών που τρέχουν σε επίπεδο πυρήνα χρησιμοποιούν τη μνήμη με διαφορετικό τρόπο από τις εφαρμογές επιπέδου χρήστη. Το κάθε επίπεδο περιλαμβάνει τη δικιά του εικονική μνήμη, η οποία καθορίζεται από το λειτουργικό σύστημα και αντιστοιχίζεται στη φυσική μνήμη του συστήματος. Οι εφαρμογές χρήστη δεν έχουν δικαίωμα πρόσβασης στην εικονική μνήμη του πυρήνα. Αντίθετα, οι οδηγοί συσκευών που τρέχουν στο επίπεδο πυρήνα, μπορούν να χρησιμοποιήσουν την εικονική μνήμη του επιπέδου χρήστη. Με τον τρόπο αυτό γίνεται συνήθως η ανταλλαγή δεδομένων μεταξύ οδηγών και εφαρμογών.

Επειδή η μνήμη επιπέδου πυρήνα είναι περιορισμένη, οι οδηγοί πρέπει να τη διαχειρίζονται αποδοτικά: είναι απαραίτητο ένας οδηγός να απελευθερώνει τη μνήμη που δέσμευσε και να χρησιμοποιεί προσεκτικά τη μνήμη στοίβας (stack memory). Η μνήμη στοίβας του επιπέδου πυρήνα είναι πολύ μικρότερη από την αντίστοιχη του επιπέδου χρήστη και δε μεγαλώνει κατά τη χρήση της. Έτσι, ένας οδηγός πρέπει να τη χρησιμοποιεί προσεκτικά και όπου είναι δυνατό να χρησιμοποιεί τη μνήμη σωρού (heap memory) αντί για τη μνήμη στοίβας. Τέλος, είναι σημαντικό ένας οδηγός να μπορεί να χειριστεί καταστάσεις χαμηλής μνήμης, διαφορετικά το σύστημα μπορεί να καταρρεύσει.

Ένα σημαντικό πρόβλημα που παρουσιάζεται στους οδηγούς συσκευών είναι τα σφάλματα σελίδας (page faults). Ένα σφάλμα σελίδας συμβαίνει όταν η εφαρμογή προσπαθεί να χρησιμοποιήσει μια σελίδα μνήμης του εικονικού χώρου διευθύνσεων η οποία δεν έχει φορτωθεί στον φυσικό χώρο διευθύνσεων. Όταν συμβαίνει ένα σφάλμα σελίδας σε ένα τμήμα κώδικα που τρέχει σε επίπεδο IRQL ≥ DISPATCH_LEVEL τότε προκαλείται σφάλμα στον οδηγό.

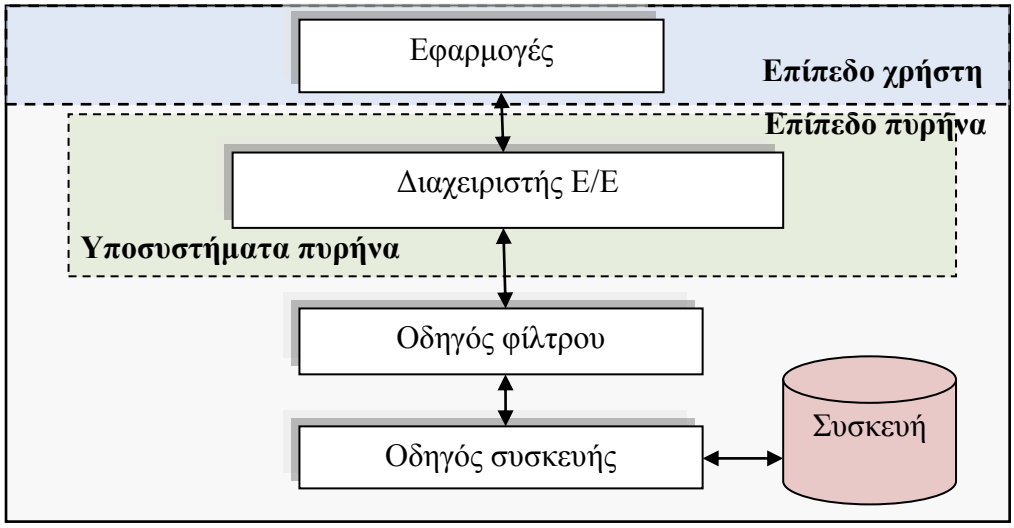
Η λύση στο παραπάνω πρόβλημα είναι η διαίρεση της μνήμης σωρού του πυρήνα σε δύο περιοχές:

- Σελιδοποιημένη περιοχή μνήμης (Paged Pool). Οι σελίδες της περιοχής αυτής μπορεί να βρίσκονται στον σκληρό δίσκο και όχι στη φυσική μνήμη, γεγονός που μπορεί να προκαλέσει σφάλματα σελίδας. Η περιοχή αυτή χρησιμοποιείται σε τμήματα κώδικα που τρέχουν σε επίπεδο IRQL < DISPATCH_LEVEL.

- Μη-Σελιδοποιημένη περιοχή μνήμης (Nonpaged Pool). Οι σελίδες της περιοχής αυτής βρίσκονται πάντα στη φυσική μνήμη και μπορεί να χρησιμοποιηθεί από τμήματα κώδικα που τρέχουν σε οποιοδήποτε επίπεδο, επειδή δε συμβαίνουν σφάλματα σελίδας.

3.6 – Οδηγοί φίλτρα

Μια ειδική κατηγορία οδηγών είναι οι οδηγοί φίλτρα (filter-drivers). Ένας οδηγός-φίλτρο υλοποιεί μια συσκευή-φίλτρο. Πρόκειται για μια ειδική συσκευή η οποία συνδέεται στην αντίστοιχη πραγματική συσκευή και συλλαμβάνει τις αιτήσεις οι οποίες κατευθύνονται προς την πραγματική συσκευή πριν αυτές φτάσουν στον προορισμό τους. Έτσι, οι οδηγοί αυτοί επιτρέπουν την επέκταση ή την αλλαγή των λειτουργιών μιας συσκευής.



Σχήμα 3.6 - Καταγραφή αιτήσεων από έναν οδηγό φίλτρο. Ο διαχειριστής Ε/Ε ανακατευθύνει τις αιτήσεις στον οδηγό φίλτρο, αντί για τον πραγματικό οδηγό. Ο οδηγός φίλτρο, αφού τις επεξεργαστεί, τις μεταβιβάζει στον πραγματικό οδηγό.

Οι οδηγοί φίλτρα χρησιμοποιούνται σε αντιϊκά προγράμματα, σε προγράμματα διατήρησης αντιγράφων ασφαλείας και σε προγράμματα κρυπτογράφησης. Ωστόσο, μπορούν να χρησιμοποιηθούν για την καταγραφή πληροφοριών από τις αιτήσεις των προγραμμάτων χρήστη προς τις συσκευές υλικού.

Ένας οδηγός φίλτρο μπορεί να συνδεθεί σε έναν πραγματικό οδηγό καλώντας τη ρουτίνα *IoAttacheDeviceByPointer* που παρέχει ο διαχειριστής Ε/Ε. Η ρουτίνα αυτή συνδέει το αντικείμενο του οδηγού φίλτρου στο αντικείμενο του πραγματικού οδηγού. Μόλις η σύνδεση ολοκληρωθεί, ο διαχειριστής Ε/Ε προωθεί τις αιτήσεις Ε/Ε στον οδηγό φίλτρο και όχι στον πραγματικό οδηγό.

3.7 – Περίληψη

Τα Windows XP αποτελούνται από δύο επίπεδα, το επίπεδο χρήστη και το επίπεδο πυρήνα. Το επίπεδο χρήστη επικοινωνεί με το επίπεδο πυρήνα, αλλά δεν έχει απευθείας πρόσβαση στους πόρους του υλικού και του συστήματος. Το επίπεδο πυρήνα είναι εκείνο που επικοινωνεί με τους οδηγούς συσκευών. Ένα από τα σημαντικότερα υποσυστήματα του επιπέδου πυρήνα είναι ο διαχειριστής Ε/Ε ο οποίος διαχειρίζεται τις αιτήσεις Ε/Ε.

Ο οδηγός συσκευής είναι ένα σύνολο από ρουτίνες τις οποίες καλεί το λειτουργικό σύστημα για να επικοινωνήσει με το υλικό του υπολογιστή και να διεκπεραιώσει τις αιτήσεις που δέχεται. Οι εφαρμογές συνήθως στέλνουν σύγχρονες αιτήσεις Ε/Ε ενώ οι λειτουργίες Ε/Ε στον οδηγό γίνονται ασύγχρονα. Όταν στο σύστημα υπάρχει κρυφή μνήμη χρησιμοποιείται η γρήγορη Ε/Ε. Στην περίπτωση αυτή, ο διαχειριστής Ε/Ε καλεί τις ρουτίνες γρήγορης Ε/Ε του οδηγού. Αν τα δεδομένα βρίσκονται στην κρυφή μνήμη ο οδηγός ολοκληρώνει την αίτηση, αλλιώς ο διαχειριστής Ε/Ε δημιουργεί ένα πακέτο αίτησης IRP και καλεί τις ρουτίνες κανονικής Ε/Ε του οδηγού. Το πακέτο IRP περιέχει τις απαραίτητες πληροφορίες για την αίτηση Ε/Ε. Για να αποφεύγονται προβλήματα συγχρονισμού οι οδηγοί χρησιμοποιούν τα επίπεδα προτεραιότητας που καθορίζονται από τα Windows XP. Επιπλέον, ένα θέμα που απαιτεί ιδιαίτερη προσοχή είναι η χρήση της μνήμης. Επειδή η μνήμη στοίβας του επιπέδου πυρήνα είναι περιορισμένη πρέπει να χρησιμοποιείται με σύνεση.

Μια ειδική κατηγορία οδηγών είναι οι οδηγοί φίλτρα που δημιουργούν μια συσκευή-φίλτρο. Η συσκευή-φίλτρο συνδέεται στην αντίστοιχη πραγματική συσκευή και συλλαμβάνει τις αιτήσεις οι οποίες κατευθύνονται προς την πραγματική συσκευή πριν αυτές φτάσουν στον προορισμό τους.

Κεφάλαιο 4

Αρχιτεκτονική

Σε αυτό το κεφάλαιο αρχικά προτείνουμε τη συγκέντρωση στατιστικών της χρήσης των αρχείων ως λύση για τη βελτίωση της αναζήτησης σε προσωπικούς υπολογιστές ή διακομιστές αποθήκευσης. Στη συνέχεια παρουσιάζουμε την αρχιτεκτονική του συστήματος που υλοποιούμε. Αφού το διαχωρίσουμε σε υποσυστήματα, εξετάζουμε το κάθε ένα χωριστά.

4.1 – Προτεινόμενη λύση

Τα συστήματα αναζήτησης που έχουν υλοποιηθεί χρησιμοποιούν μεθόδους οι οποίες δεν είναι βέλτιστες για αναζήτηση σε συστήματα αρχείων. Τα αρχεία οργανώνονται στο δίσκο με διαφορετικό τρόπο από τον οποίο οργανώνονται οι ιστοσελίδες στο Διαδίκτυο. Επειδή στα συστήματα αρχείων δεν υπάρχουν οι υπερσυνδέσεις μεταξύ των αρχείων όπως υπάρχουν στο Διαδίκτυο, ο αλγόριθμος PageRank ο οποίος βασίζεται στην ύπαρξη υπερσυνδέσεων δε μπορεί να εφαρμοστεί αποτελεσματικά, με αποτέλεσμα να μειώνεται η επίδοση των συστημάτων αναζήτησης που τον χρησιμοποιούν. Για να αντιμετωπίσουν αυτό το πρόβλημα, διάφορα άλλα συστήματα χρησιμοποιούν την προσωρινή τοπικότητα έτσι ώστε να δημιουργήσουν συσχετίσεις μεταξύ των αρχείων. Οι συσχετίσεις αυτές κάνουν δυνατή την εφαρμογή του αλγορίθμου PageRank, ωστόσο είναι δυνατό να δημιουργηθούν πολλές λανθασμένες συσχετίσεις. Ακόμη και πιο εξελιγμένα συστήματα, που χρησιμοποιώντας την αιτιότητα κατάφεραν να αποφύγουν την δημιουργία μη σωστών συσχετίσεων, φαίνεται ότι δεν μπορούν να εντοπίσουν όλες τις συσχετίσεις.

Στόχος μας είναι η ανάπτυξη ενός λογισμικού που συντελεί στη βελτίωση της απόδοσης και της ταχύτητας της αναζήτησης σε συστήματα αρχείων. Στην παρούσα εργασία προτείνουμε ως λύση τη συγκέντρωση στατιστικών της χρήσης των αρχείων στο σύστημα. Πιστεύουμε ότι η διατήρηση αυτών των στατιστικών σε έναν πίνακα κατακερματισμού θα φέρει καλύτερα αποτελέσματα στην αναζήτηση και θα μειώσει την επιβάρυνση του συστήματος από ενδεχόμενη συντήρηση γράφου

συσχετίσεων. Η συγκέντρωση των στατιστικών γίνεται μέσω ενός οδηγού φίλτρου. Ο οδηγός φίλτρο βρίσκεται ένα επίπεδο πιο πάνω από τον πραγματικό οδηγό του συστήματος αρχείων και καταγράφει όλες τις αιτήσεις που κατευθύνονται προς αυτόν. Αφού καταγράψει τις αιτήσεις τις δρομολογεί στη συνέχεια στον πραγματικό οδηγό έτσι ώστε να εξυπηρετηθούν.

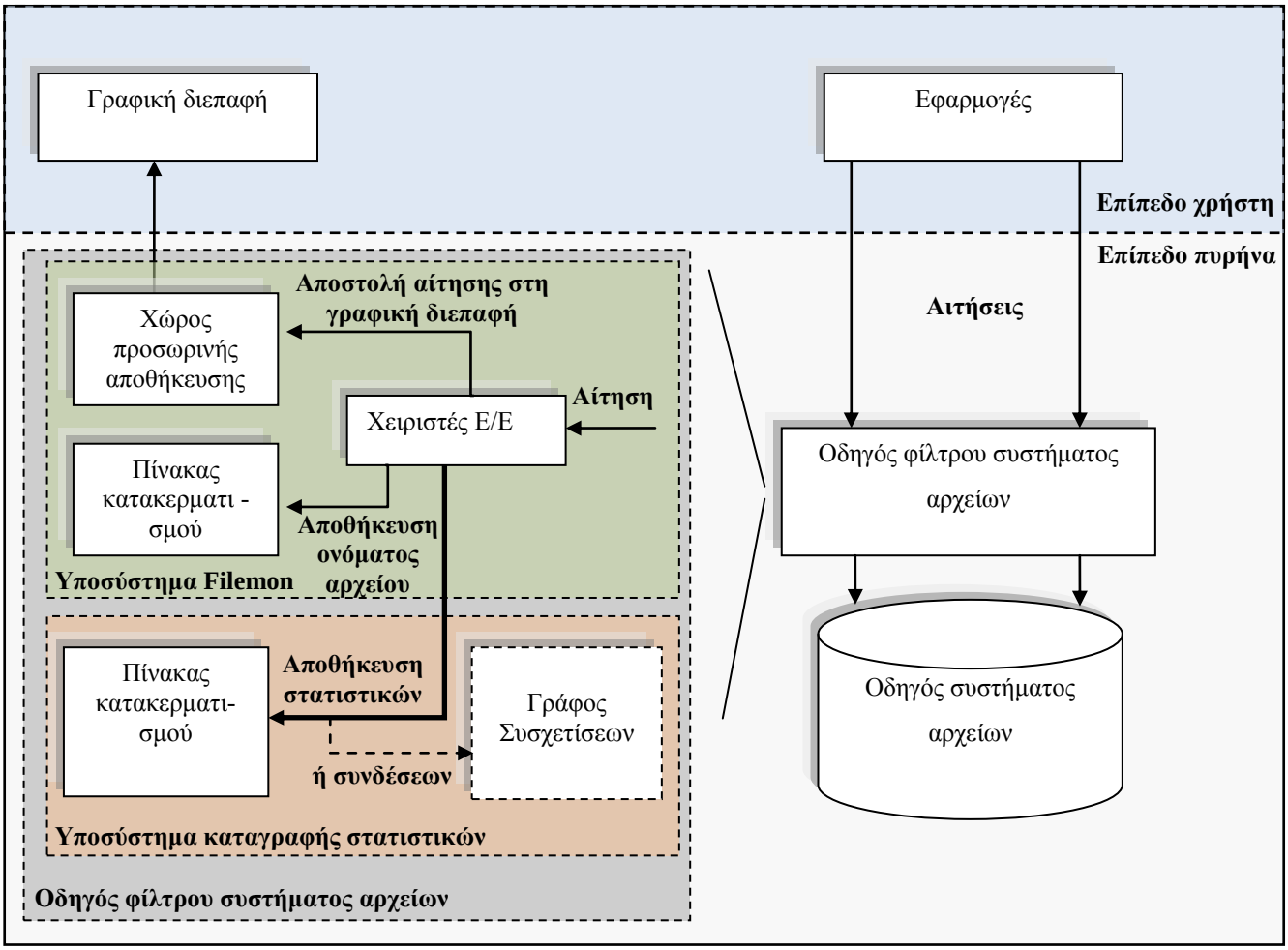
Για να διαπιστώσουμε αν η καταγραφή στατιστικών της χρήσης των αρχείων σε έναν πίνακα κατακερματισμού επιβαρύνει λιγότερο το σύστημα από τη διατήρηση ενός γράφου συσχετίσεων, υλοποιούμε ένα δεύτερο σύστημα το οποίο καταγράφει συσχετίσεις μεταξύ των αρχείων σε έναν γράφο. Έτσι, δημιουργείται ένα περιβάλλον εικονικών συνδέσεων στο οποίο μπορούν να εκτελεστούν αλγόριθμοι κατάταξης όπως ο PageRank και ο HITS. Έπειτα, εκτελούμε πειράματα συγκρίνοντας τα δύο συστήματα.

Το γεγονός ότι τα Windows είναι λογισμικό κλειστού κώδικα, καθιστά πιο δύσκολη την ανάπτυξη ενός οδηγού. Για το λόγο αυτό ως βασικό μέσο επίτευξης του στόχου της παρούσας εργασίας χρησιμοποιήθηκε ο κώδικας του Filemon τον οποίο τροποποιήσαμε σύμφωνα με τις ανάγκες μας. Το Filemon είναι ένα πρόγραμμα το οποίο χρησιμοποιεί μια γραφική διεπαφή σε συνδυασμό με έναν οδηγό φίλτρο συστήματος αρχείων (file system filter driver). Ο οδηγός-φίλτρο βρίσκεται ένα επίπεδο πάνω από τον οδηγό του συστήματος αρχείων και μπορεί να παρέμβει στις αιτήσεις που πηγαίνουν σε αυτόν. Σκοπός του είναι να παρακολουθεί και να παρουσιάζει τη δραστηριότητα των αρχείων ενός συστήματος στιγμιαία. Όταν εκτελείται το εκτελέσιμο πρόγραμμα του Filemon (Filemon.exe) εξάγεται ο οδηγός φίλτρο και φορτώνεται στη μνήμη του υπολογιστή. Μόλις ο χρήστης τερματίσει το πρόγραμμα της διεπαφής, ο οδηγός φίλτρο απομακρύνεται από το σύστημα.

4.2 – Υποσυστήματα

Το σύστημα που υλοποιούμε αποτελείται από έναν οδηγό φίλτρου και από ένα απλό πρόγραμμα διεπαφής χρήστη. Ο διαχειριστής Ε/Ε του λειτουργικού συστήματος προωθεί όλες τις αιτήσεις Ε/Ε προς τον οδηγό φίλτρο αντί για τον πραγματικό οδηγό συστήματος αρχείων. Ο οδηγός φίλτρο καταγράφει τα χαρακτηριστικά των αιτήσεων και κρατά τα στατιστικά για κάθε αρχείο. Έπειτα, η αίτηση προωθείται στον πραγματικό οδηγό για την εξυπηρέτησή της. Επιπλέον, οι αιτήσεις που καταγράφηκαν εμφανίζονται στη λίστα που περιλαμβάνει το πρόγραμμα της διεπαφής. Ο χρήστης έχει τη δυνατότητα να τροποποιήσει διάφορες παραμέτρους του οδηγού μέσω του προγράμματος διεπαφής και να αποθηκεύσει σε αρχείο της επιλογής του τα στατιστικά χρήσης των

αρχείων που έχουν καταγραφεί. Η λειτουργία και η δομή του συστήματος που προτείνουμε φαίνεται στο Σχήμα 4.1.



Σχήμα 4.1 - Λειτουργία και δομή του συστήματος που προτείνουμε. Στην παρούσα εργασία θα υλοποιήσουμε δύο εκδοχές του συστήματος: μία που θα κρατάει στατιστικά της χρήσης των αρχείων σε έναν πίνακα κατακερματισμού και μία που θα κρατάει τις συσχετίσεις μεταξύ των αρχείων σε έναν γράφο συσχετίσεων.

4.2.1 – Γραφική Διεπαφή

Η γραφική διεπαφή καθιστά δυνατή την αλληλεπίδραση του χρήστη με το πρόγραμμα και την επικοινωνία με τον οδηγό φίλτρο του συστήματος αρχείων. Μέσω της γραφικής διεπαφής προσδιορίζεται ο τόμος στον οποίο θα γίνει η παρακολούθηση των αρχείων καθώς και οι παράμετροι με βάση τις οποίες θα γίνει η παρακολούθηση των αιτήσεων. Για παράδειγμα, μπορεί η παρακολούθηση να αφορά μόνο συγκεκριμένα αρχεία και συγκεκριμένους φακέλους. Αφού ξεκινήσει η διαδικασία παρακολούθησης των αιτήσεων, η διεπαφή στέλνει ανά τακτά χρονικά διαστήματα αίτηση στον οδηγό του προγράμματος, ώστε να λάβει από εκείνον τις πληροφορίες που

έχει καταγράψει σχετικά με τις αιτήσεις στο σύστημα αρχείων. Έτσι είναι δυνατή η εμφάνιση των πληροφοριών αυτών στη διεπαφή και η μετέπειτα επεξεργασία τους από το χρήστη. Οι πληροφορίες που συλλέγονται αναφέρονται στον ακριβή χρόνο που έγινε η αίτηση, στο όνομα και τη διαδρομή της θέσης του αρχείου, στο είδος της αίτησης και στο αποτέλεσμα της αίτησης δηλαδή αν ήταν επιτυχής ή όχι.

4.2.2 – Οδηγός Φίλτρο

Εκτός από το γραφικό περιβάλλον, το σύστημα μας περιλαμβάνει και έναν οδηγό-φίλτρο συστήματος αρχείων. Όταν ο οδηγός φίλτρο λάβει την εντολή να ξεκινήσει την παρακολούθηση ενός συγκεκριμένου τόμου, δημιουργεί ένα φίλτρο αντικείμενο συσκευής (filter device object) και το συνδέει στο αντικείμενο συσκευής (device object) που αντιπροσωπεύει τον συγκεκριμένο τόμο. Έτσι, ο διαχειριστής Ε/Ε ανακατευθύνει την πορεία μιας αίτησης που πηγαίνει στο αντικείμενο συσκευής, στον οδηγό που είναι συνδεδεμένος με τη συσκευή αυτή, δηλαδή στον οδηγό του συστήματός μας.

Ο οδηγός για να μπορέσει να λάβει και να καταγράψει τις πληροφορίες των αιτήσεων, περιλαμβάνει ένα σύνολο συναρτήσεων που χρησιμοποιούνται για τη διαχείριση των αιτήσεων που λαμβάνει ο οδηγός φίλτρο, έναν χώρο προσωρινής αποθήκευσης (buffer) και έναν πίνακα κατακερματισμού (hash table). Επιπλέον, περιέχονται και άλλες συναρτήσεις οι οποίες είναι υπεύθυνες για τη διαχείριση και τη λειτουργία των δομών, των φίλτρων και των προσωρινών χώρων αποθήκευσης και για τη δημιουργία και σύνδεση των συσκευών αντικειμένων με τις πραγματικές συσκευές συστήματος αρχείων για τις μονάδες που επιθυμεί ο χρήστης να παρακολουθήσει. Στη συνέχεια της παρούσας ενότητας θα περιγράψουμε τα βασικά τμήματα του οδηγού φίλτρου και του υποσυστήματος που θα διατηρεί τα στατιστικά.

Ρουτίνες διαχείρισης Ε/Ε

Ο οδηγός φίλτρου περιλαμβάνει ρουτίνες για τη διαχείριση των αιτήσεων Ε/Ε που φτάνουν σε αυτόν. Οι ρουτίνες αυτές καλούνται από τον διαχειριστή Ε/Ε και αφού επεξεργαστούν την αίτηση τη μεταβιβάζουν στον πραγματικό οδηγό συσκευής. Οι ρουτίνες μπορούν να χωριστούν σε δύο κατηγορίες: η πρώτη περιλαμβάνει τις ρουτίνες που είναι υπεύθυνες για την γρήγορη Ε/Ε και η δεύτερη τις ρουτίνες που διαχειρίζονται τα πακέτα IRP.

Για την καταγραφή στατιστικών που προτείναμε στην παρούσα εργασία, τροποποιήσαμε τις ρουτίνες που σχετίζονται με τη διαχείριση των αιτήσεων ανάγνωσης και εγγραφής, ώστε να μπορούμε να καταγράφουμε τα αρχεία στα οποία αναφέρεται η κάθε αίτηση. Έτσι, οι ρουτίνες διαχείρισης Ε/Ε συνδέουν τον υπάρχον οδηγό του Filemon με το υποσύστημα καταγραφής στατιστικών που υλοποιήσαμε.

Χώρος αποθήκευσης

Για να είναι δυνατή η ανταλλαγή δεδομένων μεταξύ του οδηγού φίλτρου και του προγράμματος διεπαφής ο οδηγός παρέχει ένα χώρο προσωρινής αποθήκευσης στον οποίο αποθηκεύονται όλα τα δεδομένα των αιτήσεων και οι ρουτίνες για τη διαχείρισή του.

Πίνακας Κατακερματισμού

Όταν ο οδηγός φίλτρο λάβει μια αίτηση από το σύστημα αρχείων, ενημερώνει έναν πίνακα κατακερματισμού καταχωρώντας τον χειριστή αρχείου (file handler) που φέρει η αίτηση και το αντίστοιχο όνομα του αρχείου. Κάθε φορά που έρχεται μια αίτηση γίνεται έλεγχος αν ο συγκεκριμένος χειριστής αρχείου έχει ήδη εισαχθεί στον πίνακα. Αν υπάρχει τότε γίνεται αναζήτηση του χειριστή και εξάγεται το αντίστοιχο όνομα. Διαφορετικά, ο οδηγός βρίσκει το πλήρες όνομα του αρχείου και το εισάγει στον πίνακα κατακερματισμού μαζί με τον χειριστή του αρχείου. Έτσι, την επόμενη φορά που θα έρθει αίτηση προς το συγκεκριμένο αρχείο, το όνομα του να βρεθεί αμέσως μέσω του πίνακα.

Υποσύστημα καταγραφής στατιστικών και συνδέσεων

Στην εργασία μας προτείναμε ως λύση για τη διατήρηση των στατιστικών της χρήσης των αρχείων μια δομή πίνακα κατακερματισμού. Το προκαθορισμένο μέγεθος του πίνακα είναι 1.000.000 θέσεις και για την αποφυγή συγκρούσεων σε κάθε θέση του πίνακα υπάρχει μια απλά διασυνδεδεμένη λίστα. Σε κάθε θέση του πίνακα αποθηκεύεται το όνομα και η διαδρομή ενός αρχείου (path) και δύο τιμές που αντιστοιχούν στον αριθμό των εγγραφών και των αναγνώσεων που πραγματοποιήθηκαν στο συγκεκριμένο αρχείο.

Επίσης, παρέχονται ρουτίνες διαχείρισης των στατιστικών και του πίνακα κατακερματισμού οι οποίες είναι υπεύθυνες για την εισαγωγή νέων ή την ενημέρωση των υπαρχόντων στατιστικών του πίνακα, τη διαγραφή εγγραφών από τον πίνακα ή την εκτύπωσή τους σε αρχείο.

Για να διαπιστωθεί αν όντως η καταγραφή στατιστικών της χρήσης των αρχείων σε πίνακα κατακερματισμού λειτουργεί θετικά και βελτιώνει την αναζήτηση στο σύστημα αρχείων, συγκρίνουμε την απόδοση της τεχνικής αυτής με την καταγραφή συσχετίσεων μεταξύ των αρχείων. Για την καταγραφή των συσχετίσεων χρησιμοποιούμε ένα γράφο συσχετίσεων προκειμένου να δημιουργήσουμε συνδέσεις μεταξύ των αρχείων ώστε να είναι δυνατή η εκτέλεση αλγορίθμων κατάταξης όπως ο PageRank. Κάθε κόμβος του γράφου αντιστοιχεί σε ένα αρχείο. Όταν τα αρχεία συσχετίζονται μεταξύ τους συνδέονται με μια ακμή. Για να προσδιορίσουμε πότε δύο αρχεία συσχετίζονται καθορίζουμε ένα χρονικό διάστημα μέσα στο οποίο τα δύο αρχεία χρησιμοποιήθηκαν. Το χρονικό αυτό διάστημα ονομάζεται παράθυρο χρόνου συσχετίσεων (connections time window). Κάθε ακμή του γράφου έχει ένα βάρος το οποίο αρχικοποιείται με την τιμή 1. Κάθε φορά που γίνεται αίτηση προς το συγκεκριμένο αρχείο μέσα στο ίδιο χρονικό διάστημα το βάρος της ακμής προσαυξάνεται κατά ένα. Τέλος, είναι σημαντικό να αναφέρουμε την ύπαρξη μιας ρουτίνας που μειώνει τα βάρη του γράφου και όταν είναι απαραίτητο διαγράφει ακμές ή και κόμβους. Το χρονικό διάστημα που μεσολαβεί μεταξύ δύο διαδοχικών ανανεώσεων του γράφου το ονομάζουμε παράθυρο ενημέρωσης ακμών (aging time window).

4.3 – Περίληψη

Οι μέθοδοι αναζήτησης που έχουν υλοποιηθεί για αναζήτηση σε συστήματα αρχείων δεν είναι βέλτιστες. Προτείνουμε μια εναλλακτική οργάνωση των αρχείων σε έναν πίνακα κατακερματισμού καταγράφοντας στατιστικά της χρήσης τους, όπως ο αριθμός των εγγραφών και αναγνώσεων. Το σύστημα που υλοποιούμε περιλαμβάνει έναν οδηγό φίλτρου συστήματος αρχείων, ο οποίος καταγράφει τις αιτήσεις των εφαρμογών στο σύστημα αρχείων. Ο οδηγός φίλτρου περιλαμβάνει αρχικά μια δομή πίνακα κατακερματισμού στον οποίο καταγράφει τα στατιστικά χρήσης των αρχείων. Στη συνέχεια, αντικαθιστούμε τον πίνακα κατακερματισμού με έναν γράφο στον οποίο καταγράφονται συσχετίσεις μεταξύ των αρχείων. Σκοπός μας είναι να συγκρίνουμε την απόδοση των δύο εναλλακτικών αυτών μεθόδων οργάνωσης των αρχείων. Επιπλέον, περιλαμβάνεται και ένα πρόγραμμα διεπαφής χρήστη από το οποίο γίνεται ο καθορισμός των παραμέτρων του οδηγού και η εμφάνιση των στατιστικών.

Κεφάλαιο 5

Υλοποίηση συστήματος

Στην παρούσα ενότητα θα περιγράψουμε τα βήματα που ακολουθήσαμε για να υλοποιήσουμε το σύστημα καταγραφής στατιστικών που προτείναμε και τις δομές δεδομένων που χρησιμοποιήσαμε. Αρχικά, κάνουμε μια αναφορά στις ρουτίνες του Filemon που τροποποιήσαμε και στη σύνδεση του υπάρχοντος λογισμικού με το υποσύστημα καταγραφής στατιστικών. Στη συνέχεια, περιγράφουμε τα δύο υποσυστήματα καταγραφής στατιστικών και συνδέσεων που υλοποιήσαμε παρουσιάζοντας τις δομές τους, πίνακα κατακερματισμού και γράφο και τις ρουτίνες διαχείρισής τους.

5.1 – Τροποποίηση του κώδικα του Filemon

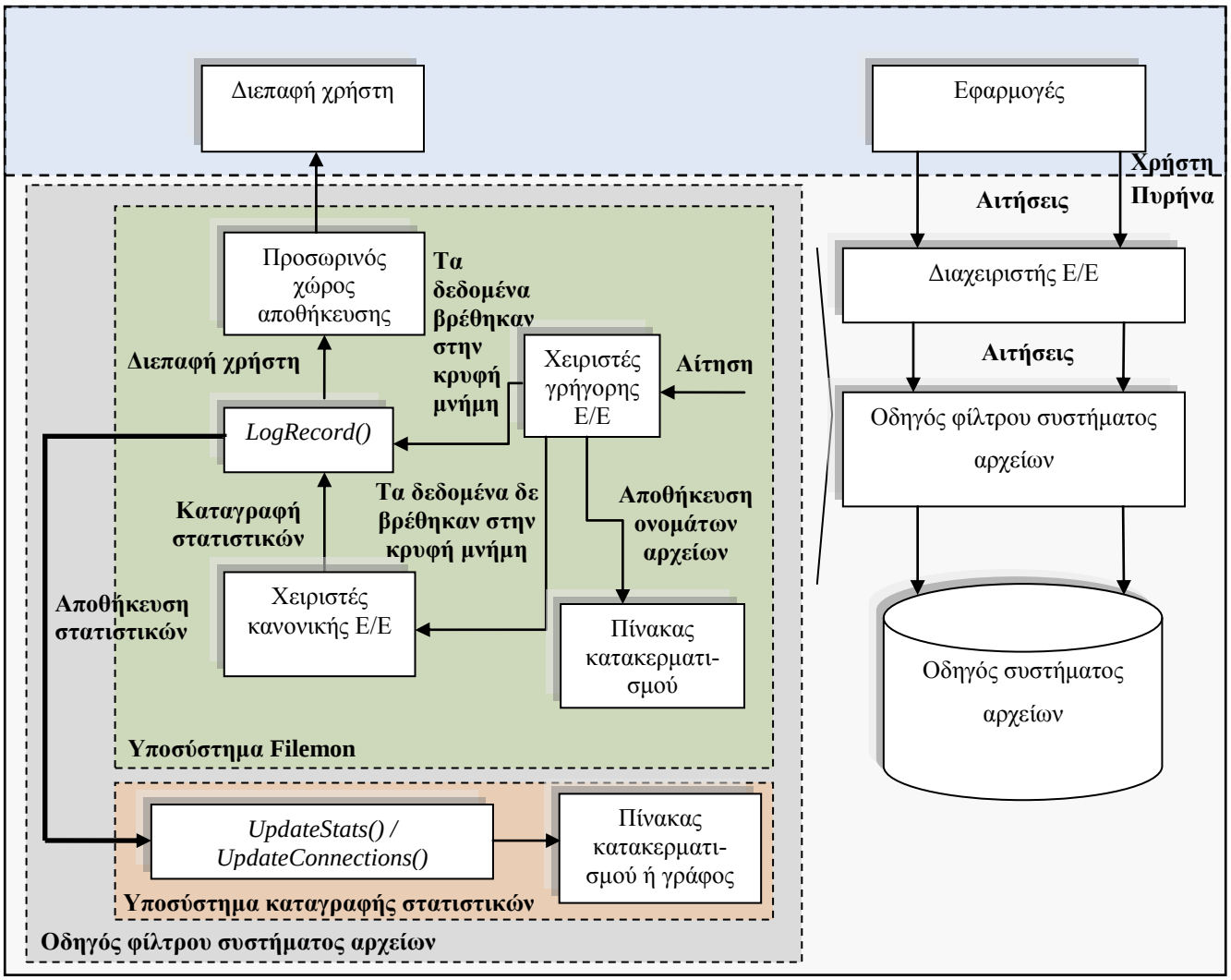
Ο συνδετικός κρίκος μεταξύ του υπάρχοντος κώδικα του Filemon και του υποσυστήματος καταγραφής στατιστικών είναι οι ρουτίνες διαχείρισης Ε/Ε. Στα πλαίσια της παρούσας εργασίας τροποποιήσαμε τις ρουτίνες που σχετίζονται με τη διαχείριση των αιτήσεων εγγραφής και ανάγνωσης, έτσι ώστε να είναι δυνατή η καταγραφή των αρχείων στα οποία έγιναν οι αιτήσεις και των αντίστοιχων στατιστικών τους.

Όπως ήδη αναφέραμε, η κύρια μέθοδος για τη μεταβίβαση μιας αίτησης Ε/Ε από τον διαχειριστή Ε/Ε σε έναν οδηγό συσκευής είναι η γρήγορη Ε/ Ε (Fast I/O). Είναι σημαντικό λοιπόν ο οδηγός φίλτρο να παρέχει λειτουργίες για γρήγορη Ε/Ε, οι οποίες δηλώνονται στο αντικείμενο του οδηγού στη συνάρτηση *DriverEntry()* η οποία καλείται από τον διαχειριστή Ε/Ε κατά τη φόρτωση του οδηγού.

Σε περίπτωση που τα δεδομένα δε βρίσκονται στην κρυφή μνήμη, ο οδηγός συσκευής ενημερώνει τον διαχειριστή Ε/Ε, ο οποίος καταφεύγει στη χρήση των παραδοσιακών μεθόδων δημιουργίας μιας αίτησης IRP. Αφού δημιουργήσει την αίτηση IRP, ο διαχειριστής Ε/Ε καλεί τις

ρουτίνες του οδηγού που είναι υπεύθυνες για τη διαχείρισή της. Ο οδηγός φίλτρου πρέπει να υλοποιεί και να δηλώνει στο αντικείμενο του οδηγού τις ρουτίνες που διαχειρίζονται κάθε τύπο IRP και όχι μόνο τους τύπους IRP που τον ενδιαφέρουν αποκλειστικά. Ο κύριος λόγος που συμβαίνει αυτό είναι ότι ο οδηγός φίλτρο θα πρέπει να προωθεί όλες τις αιτήσεις IRP προς τον κατώτερο πραγματικό οδηγό στη διαστρωμάτωση, έτσι ώστε οι αιτήσεις τελικά να εξυπηρετούνται.

Και στις δύο περιπτώσεις, είτε στη γρήγορη αίτηση E/E, είτε στην αποστολή πακέτου αίτησης IRP, η καταγραφή των πληροφοριών των αιτήσεων γίνεται μέσω της συνάρτησης *LogRecord()*.



Σχήμα 5.1 - Διαχείριση των αιτήσεων E/E από τον οδηγό του Filemon. Η αίτηση εξυπηρετείται είτε με το μηχανισμό της γρήγορης E/E, είτε με τη δημιουργία πακέτων IRP. Τα στατιστικά καταγράφονται με τη βοήθεια της ρουτίνας LogRecord(), η οποία αποτελεί το συνδετικό κρίκο μεταξύ του Filemon και του υποσυστήματος καταγραφής στατιστικών.

5.1.1 – Συναρτήσεις διαχείρισης της γρήγορης Ε/Ε

Στην περίπτωση μιας γρήγορης αίτησης Ε/Ε, ο διαχειριστής Ε/Ε καλεί μια από τις ρουτίνες που περιλαμβάνονται στον οδηγό φίλτρου για τη διαχείριση των αιτήσεων γρήγορης Ε/Ε. Για κάθε είδος αίτησης υπάρχει και μια διαφορετική ρουτίνα. Εκείνες που μας ενδιαφέρουν περισσότερο είναι οι *FilemonFastIoRead()* και *FilemonFastIoWrite(),* οι οποίες καλούνται σε περίπτωση που έρθει αίτηση γρήγορης Ε/Ε για ανάγνωση και εγγραφή αντίστοιχα. Εκεί, αφού ολοκληρωθεί η αίτηση από τον πραγματικό οδηγό στον οποίο αρχικά απευθυνόταν μέσω των συναρτήσεων *FastIoRead()* και *FastIoWrite()* αντίστοιχα, καταγράφονται οι πληροφορίες της αίτησης με την κλήση της *logRecord()*.

Όνομα Συνάρτησης	**Λειτουργία**
FilemonFastIoRead()	Καλεί τη συνάρτηση *FastIoRead* που παρέχει το αντικείμενο οδηγού της πραγματικής συσκευής για την ολοκλήρωση της αίτησης Ε/Ε. και στη συνέχεια εισάγει στον πίνακα κατακερματισμού το αντικείμενο αρχείο, το οποίο αφορά η αίτηση, και τη διαδρομή καταλόγων που το προσδιορίζει.
FilemonFastIoWrite()	Καλεί τη συνάρτηση *FastIoWrite* που παρέχει το αντικείμενο οδηγού της πραγματικής συσκευής για την ολοκλήρωση της αίτησης Ε/Ε και καταγράφει την αίτηση εγγραφής στον προσωρινό χώρο αποθήκευσης έτσι ώστε να εμφανιστεί από το πρόγραμμα διεπαφής. Επίσης εισάγει το αντικείμενο αρχείο, το οποίο αφορά η αίτηση, και τη διαδρομή καταλόγων που το προσδιορίζει στον πίνακα κατακερματισμού, αν δεν υπάρχει ήδη.

Πίνακας 5.1 – Σύντομη περιγραφή της λειτουργίας των συναρτήσεων για τη διαχείριση της γρήγορης Ε/Ε.

5.1.2 – Συναρτήσεις διαχείρισης των αιτήσεων IRP

Η κύρια ρουτίνα που διαχειρίζεται τις αιτήσεις IRP του Filemon είναι η *FilemonDispatch()*. Αρχικά, εξετάζει αν η αίτηση σχετίζεται με τη γραφική διεπαφή του Filemon. Εάν ναι, καλεί τη

ρουτίνα *FilemonDeviceRoutine* () για να την εξυπηρετήσει. Διαφορετικά, εάν η αίτηση απευθύνεται σε μία συσκευή υλικού καλεί τη ρουτίνα *FilemonHookRoutine()*.

Η *FilemonHookRoutine()* είναι η ρουτίνα που είναι υπεύθυνη για τον έλεγχο και την καταγραφή όλων των αιτήσεων που πραγματοποιούνται από τις εφαρμογές χρήστη στο σύστημα αρχείων που παρακολουθείται. Ελέγχει το είδος της αίτησης και αν πρόκειται για μια αίτηση ανάγνωσης ή εγγραφής προς το σύστημα αρχείων η ενέργεια καταγράφεται με κλήση της συνάρτησης *LogRecord()*.

Όνομα Συνάρτησης	**Λειτουργία**
FilemonDispatch()	Η *FilemonDispatch* είναι η κύρια ρουτίνα διαχείρισης των αιτήσεων IRP που λαμβάνει το Filemon. Ελέγχει αν η αίτηση έχει προέλθει από το πρόγραμμα της διεπαφής ή αν πρόκειται για αίτηση Ε/Ε στο σύστημα αρχείων που παρακολουθείται. Στη συνέχεια, καλεί την κατάλληλη ρουτίνα για να την εξυπηρετήσει.
FilemonDeviceRoutine()	Διαχειρίζεται τις αιτήσεις IRP που προέρχονται από το πρόγραμμα της διεπαφής. Αφού ελέγξει τον τύπο του IRP για να ανακαλήψει το είδος της αίτησης, εκτελεί τις κατάλληλες ενέργεις για να την εξυπηρετήσει.
FilemonHookRoutine()	Η συνάρτηση αυτή διαχειρίζεται τις αιτήσεις Ε/Ε που απευθύνονται στο σύστημα αρχείων που παρακολουθείται. Αρχικά ελέγχεται το είδος της αίτησης και καταγράφεται στο χώρο προσωρινής αποθήκευσης ώστε να εμφανιστεί από το πρόγραμμα της διεπαφής χρήστη. Στη συνέχεια, η αίτηση προωθείται στον πραγματικό οδηγό συσκευής στον οποίο απευθυνόταν με κλήση της ρουτινας *IoCallDriver*.

Πίνακας 5.2 – Σύντομη περιγραφή της λειτουργίας των συναρτήσεων για τη διαχείριση των αιτήσεων IRP.

5.2 – Υποσύστημα καταγραφής στατιστικών και συνδέσεων

Στην ενότητα αυτή παρουσιάζουμε τα δύο υποσυστήματα καταγραφής στατιστικών και συνδέσεων που υλοποιήσαμε. Το πρώτο χρησιμοποιεί έναν πίνακα κατακερματισμού στον οποίο

αποθηκεύει στατιστικά της χρήσης των αρχείων στο σύστημα, ενώ το δεύτερο έναν γράφο συσχετίσεων στον οποίο καταγράφονται συνδέσεις μεταξύ των αρχείων.

5.2.1 – Υποσύστημα καταγραφής στατιστικών

Ένα σημαντικό θέμα για την αποδοτική λειτουργία του συστήματος είναι η επιλογή της κατάλληλης δομής για την αποτελεσματική και γρήγορη αποθήκευση και ανάκτηση των δεδομένων. Στην περίπτωση της τυχαίας ή σειριακής τοποθέτησης των δεδομένων σε μια δομή, όλες οι εργασίες (όπως εισαγωγή, αναζήτηση, διαγραφή) απαιτούν χρόνο ανάλογο με το πλήθος των αποθηκευμένων δεδομένων (*O(n)*, αν *n* είναι το πλήθος των δεδομένων). Η κατάσταση βελτιώνεται όταν τα δεδομένα αποθηκευτούν με κάποιον δομημένο τρόπο. Ο στόχος του πίνακα κατακερματισμού είναι να επιτευχθεί σταθερός χρόνος πρόσβασης, ανεξάρτητος από το πλήθος των δεδομένων. Ας υποθέσουμε ότι μπορούμε να υπολογίσουμε τη θέση στην οποία θα έπρεπε να βρίσκεται ένα αντικείμενο αν ήταν αποθηκευμένο στη δομή, τότε, ανεξάρτητα από το πλήθος των δεδομένων που έχουμε, έχουμε πετύχει να έχουμε σταθερό χρόνο πρόσβασης στα δεδομένα.

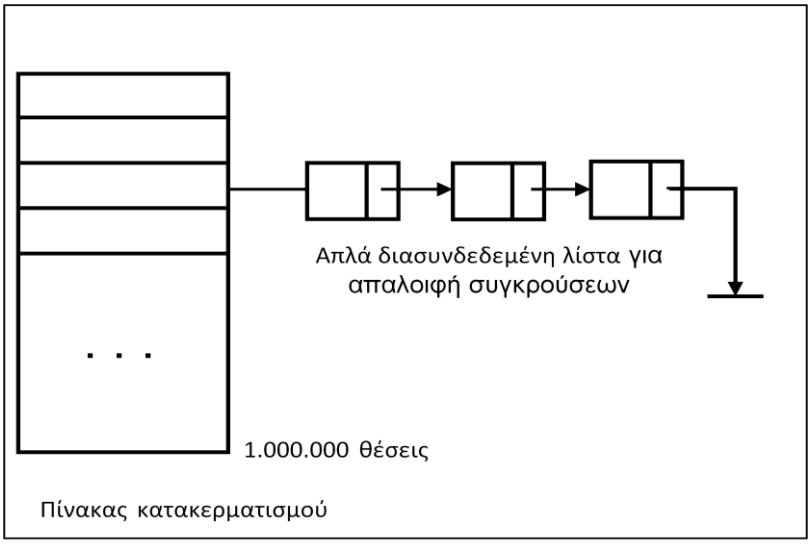
```
ULONG HASHSTATOBJECT(
      char *fileName
      )
{
      int i, length;

      #if defined(_IA64_)
            ULONG_PTR hash = 0;
      #else
            ULONG hash = 0;
      #endif
      int c;

      length = strlen( fileName);
      for (i = 0; i < length; i++) {
            c = fileName[i];
            hash = c + (hash << 6) + (hash << 16) - hash;
      }
      return (hash%NUMHASHSTAT);
}
```

Σχήμα 5.2 - Η συνάρτηση κατακερματισμού που επιλέξαμε. Πρόκειται για μια συνάρτηση κατακερματισμού γενικής χρήσης η οποία έχει διαπιστωθεί πως έχει καλή απόδοση προκαλώντας την καλύτερη κατανομή των κλειδιών και λιγότερες διασπάσεις. Έχει αποδειχθεί επίσης ότι έχει καλή διασπορά.

Τον υπολογισμό της θέσης στον πίνακα όπου βρίσκεται κάθε αντικείμενο που εισέρχεται σε αυτόν τον πετυχαίνουμε με τη χρήση μιας συνάρτησης κατακερματισμού. Η συνάρτηση κατακερματισμού που επιλέξαμε είναι μια γενική συνάρτηση κατακερματισμού, που

χρησιμοποιείται σε διάφορα συστήματα όπως για παράδειγμα στο BerkeleyDB (Σχήμα 5.2) και χρησιμοποιεί ως κλειδί το όνομα του αρχείου που θέλουμε να εισάγουμε στον πίνακα.

Ωστόσο, η συνάρτηση κατακερματισμού δεν εγγυάται ότι δύο αντικείμενα δε θα εισέλθουν στην ίδια θέση. Για το λόγο αυτό, κάθε θέση του πίνακα κατακερματισμού αποτελείται από μια απλά συνδεδεμένη λίστα ώστε να αποφεύγονται οι συγκρούσεις. Η δομή αυτή διακρίνεται στο Σχήμα 5.3. Να σημειώσουμε στο σημείο αυτό πως το προκαθορισμένο μέγεθος του πίνακα επιλέχθηκε να είναι 1.000.000 θέσεις.



Σχήμα 5.3 - Διακρίνεται η δομή του πίνακα κατακερματισμού. Κάθε θέση του πίνακα περιέχει μια απλά διασυνδεδεμένη λίστα για την αποφυγή τυχόν συγκρούσεων

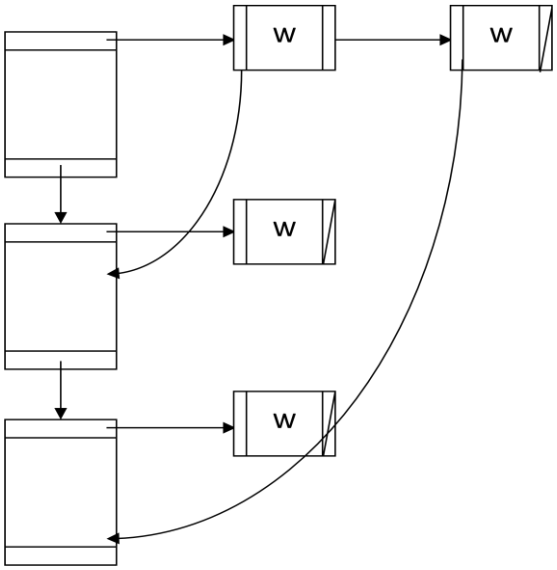
Τα αντικείμενα που αποθηκεύουμε στον πίνακα κατακερματισμού περιλαμβάνουν τη διαδρομή του αρχείου στο οποίο έγινε η αίτηση (File Path), το αντικείμενο του αρχείου, καθώς και δύο μετρητές για τις αναγνώσεις και τις εγγραφές που έχουν γίνει σε αυτό. Πιστεύουμε ότι η καταγραφή των συνολικών εγγραφών και αναγνώσεων που γίνονται στα αρχεία θα αποτυπώνει ποια από αυτά χρησιμοποιεί συχνότερα ο χρήστης. Οι μετρητές αυτοί μπορούν να αποτελέσουν ένα κριτήριο κατάταξης των αποτελεσμάτων της αναζήτησης, έτσι ώστε να εμφανίζονται πρώτα τα αρχεία που χρησιμοποιεί πιο συχνά ο χρήστης.

Έχοντας περιγράψει τη δομή που χρησιμοποιήσαμε θα αναφέρουμε σύντομα τον τρόπο με τον οποίο τη διαχειριζόμαστε. Η κύρια ρουτίνα διαχείρισης του πίνακα κατακερματισμού είναι η *UpdateStats()*, η οποία καλείται από τη *logRecord()* κάθε φορά που γίνεται μια νέα αίτηση εγγραφής ή ανάγνωσης σε κάποιο αρχείο. Η ρουτίνα *UpdateStats()* αρχικά ελέγχει αν το αρχείο έχει εισαχθεί

στον πίνακα. Αν έχει εισαχθεί, τότε αυξάνει τον μετρητή στον οποίο αναφέρεται η αίτηση που έγινε, διαφορετικά εισάγει το αρχείο στον πίνακα και αρχικοποιεί τους μετρητές με μηδέν. Εκτός από τη ρουτίνα αυτή, υπάρχουν και άλλες ρουτίνες διαχείρισης του πίνακα κατακερματισμού οι οποίες διαγράφουν εγγραφές του ή αποθηκεύουν τις εγγραφές που περιέχει σε αρχείο όταν ζητηθεί από το χρήστη.

5.2.2 – Υποσύστημα καταγραφής συνδέσεων

Το δεύτερο υποσύστημα που υλοποιούμε καταγράφει τις συνδέσεις μεταξύ των αρχείων σε έναν γράφο. Σκοπός είναι να δημιουργηθεί ένα περιβάλλον εικονικών συνδέσεων στο οποίο θα μπορούν να τρέχουν οι αλγόριθμοι κατάταξης όπως ο PageRank και ο HITS. Ο γράφος που χρησιμοποιήσαμε εδώ υλοποιείται με λίστες ακμών. Κάθε κόμβος του γράφου περιλαμβάνει το όνομα και τη διαδρομή ενός αρχείου (path), καθώς και την τελευταία χρονική στιγμή όπου έγινε μια αίτηση εγγραφής ή ανάγνωσης στο αρχείο αυτό. Οι ακμές περιλαμβάνουν βάρη και κάθε φορά που δύο συσχετιζόμενα αρχεία συνδέονται αντί να δημιουργηθεί μια νέα ακμή αυξάνεται το βάρος της ήδη υπάρχουσας. Η δομή αυτή φαίνεται στο Σχήμα 5.4. Να σημειώσουμε εδώ πως ο γράφος δημιουργείται κατά τη φόρτωση του οδηγού, στη ρουτίνα *DriverEntry()*.



Σχήμα 5.4 - Διακρίνεται η δομή του γράφου συσχετίσεων. Οι κόμβοι του γράφου συνδέονται μεταξύ τους με λίστες ακμών. Κάθε κόμβος της λίστας ακμών περιέχει το βάρος της σύνδεσης το οποίο αυξάνεται κάθε φορά που δύο ήδη συνδεδεμένοι κόμβοι συνδέονται.

Η βασική ιδέα που χρησιμοποιούμε εδώ είναι η εξής: δύο αρχεία που χρησιμοποιούνται μέσα σε ένα συγκεκριμένο χρονικό διάστημα συνδέονται μεταξύ τους. Αν υπήρχε ήδη μια ακμή που

συνέδεε τα δύο αυτά αρχεία στο γράφο, δε δημιουργείται νέα, αλλά αυξάνεται το βάρος της ήδη υπάρχουσας. Για τη συντήρηση του γράφου, καθορίζεται ένα χρονικό διάστημα στο τέλος του οποίου μειώνονται στο μισό τα βάρη των ακμών. Ορίζουμε λοιπόν τα εξής χρονικά διαστήματα:

- Παράθυρο χρόνου συσχετίσεων: Αν δύο αρχεία χρησιμοποιηθούν μέσα σε αυτό το χρονικό διάστημα θεωρούμε ότι συσχετίζονται.

- Παράθυρο ενημέρωσης ακμών: Στο τέλος αυτού του χρονικού διαστήματος εκτελείται η λειτουργία συντήρησης στο γράφο που θα περιγράψουμε στη συνέχεια.

Εκτός από τη δομή του γράφου, χρησιμοποιούμε και μία δομή ουράς ως βοηθητική. Η ουρά ουσιαστικά μας πληροφορεί για το ποια αρχεία ανήκουν στο ίδιο παράθυρο με κάποιο αρχείο που προσπελάστηκε πρόσφατα. Έτσι, κάθε φορά που γίνεται αίτηση σε ένα αρχείο δε χρειάζεται να εξετάσουμε όλα τα αρχεία του γράφου για να βρούμε με ποια συνδέεται, αλλά ένα μικρό υποσύνολό τους που διατηρούμε στην ουρά. Καθώς ο χρόνος προχωράει και η ουρά γεμίζει με νέα αρχεία, τα αρχεία που προσπελάστηκαν παλαιότερα απομακρύνονται.

Στο σημείο αυτό παρουσιάζουμε τον βασικό αλγόριθμο που εκτελείται κάθε φορά που γίνεται αίτηση σε ένα αρχείο (Σχήμα 5.5):

1. Αρχικά, ελέγχουμε αν το αρχείο στο οποίο έγινε η αίτηση υπάρχει ήδη στο γράφο και αν ναι ενημερώνουμε το χρόνο πιο πρόσφατης χρήσης του.

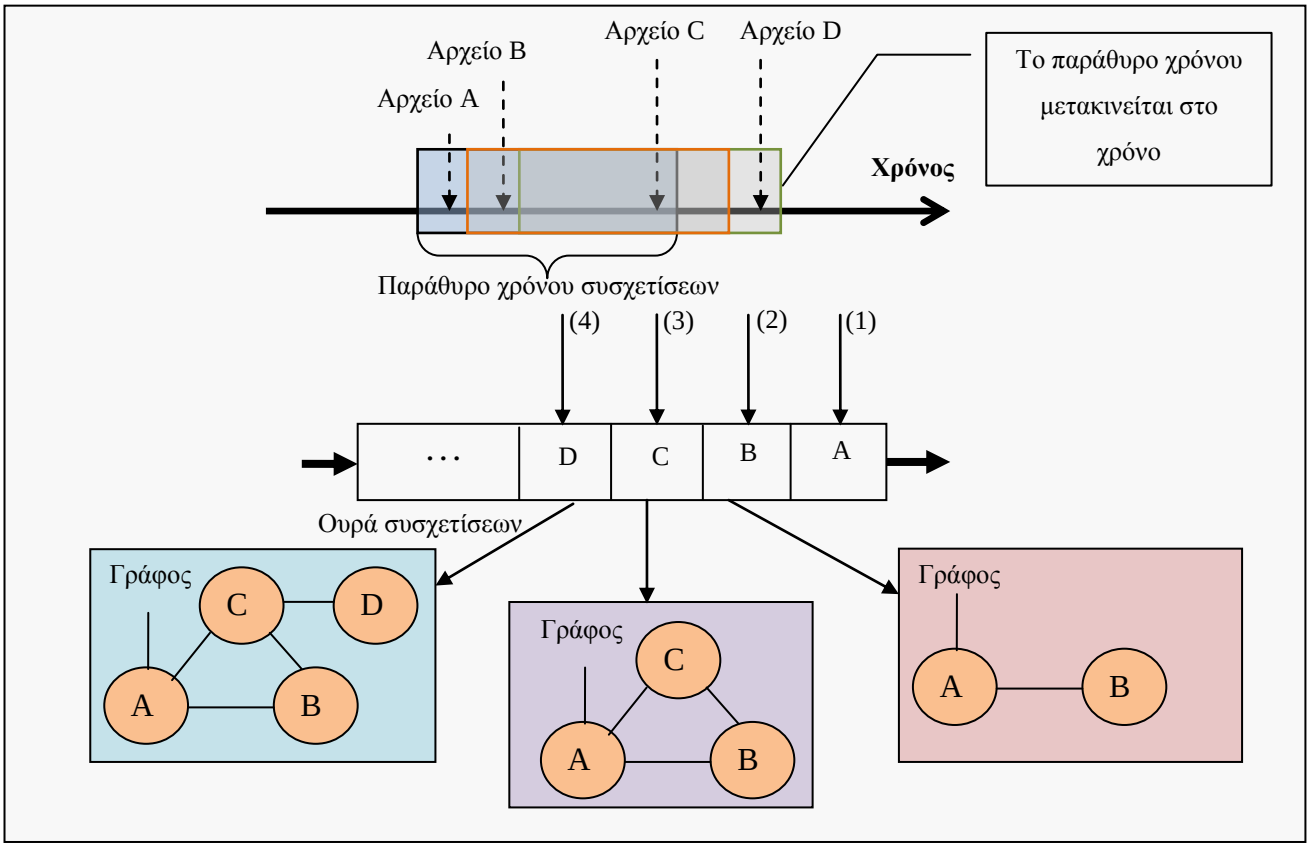
2. Στη συνέχεια, διατρέχουμε την ουρά και εξετάζουμε αν το τρέχον αρχείο της ουράς είναι στο ίδιο παράθυρο με το νέο αρχείο. Αν ναι, τα δύο αρχεία εισέρχονται στο γράφο, αν δεν έχουν εισαχθεί ήδη, και συνδέονται με μια ακμή ή αυξάνεται το βάρος της υπάρχουσας αν είναι ήδη συνδεδεμένα. Διαφορετικά, αν τα δύο αρχεία δε συνδέονται μεταξύ τους, απομακρύνουμε το τρέχον αρχείο από την ουρά.

3. Το νέο αρχείο εισέρχεται στην ουρά. Μια ειδική περίπτωση προκύπτει όταν το νέο αρχείο βρίσκεται ήδη στην ουρά. Τότε, εξαγάγουμε από την ουρά το ήδη υπάρχον αρχείο και το επανεισαγάγουμε στο τέλος της χρησιμοποιώντας την νέα χρονική στιγμή της αίτησης.

Ο παραπάνω αλγόριθμος εκτελείται στη ρουτίνα *updateConnections(),* η οποία είναι η βασική ρουτίνα διαχείρισης του γράφου και καλείται από την *logRecord()* του Filemon κάθε φορά που γίνεται αίτηση ανάγνωσης η εγγραφής σε κάποιο αρχείο. Στο σημείο αυτό είναι σημαντικό να σημειώσουμε πως, αφού ολοκληρωθεί η εκτέλεση του παραπάνω αλγορίθμου, ελέγχουμε αν έχει

περάσει το χρονικό διάστημα για την ενημέρωση των ακμών. Στην περίπτωση που πρέπει να γίνει ενημέρωση, καλείται η ρουτίνα *decreaseWeight()*, η οποία είναι υπεύθυνη για την ενημέρωση των ακμών του γράφου.

Κατά την ενημέρωση των ακμών του γράφου, μειώνουμε τα βάρη από όλες τις ακμές στο μισό και αν σε μία ακμή το βάρος γίνει μηδέν τη διαγράφουμε. Επίσης, αν ένας κόμβος μείνει χωρίς ακμές τότε τον διαγράφουμε από το γράφο απελευθερώνοντας τη μνήμη που είχαμε δεσμεύσει γι' αυτόν.



Σχήμα 5.5 - Διακρίνεται ο αλγόριθμος που χρησιμοποιούμε για την εισαγωγή ενός αρχείου στο γράφο. Δύο αρχεία συνδέονται μεταξύ τους όταν χρησιμοποιούνται μέσα σε ένα χρονικό διάστημα, το παράθυρο χρόνου συσχετίσεων. Το παράθυρο αυτό μετακινείται στο χρόνο. Ας θεωρήσουμε ότι αρχικά ο γράφος και η ουρά είναι άδεια. Γίνεται μια αίτηση στο αρχείο A, το οποίο εισάγεται απευθείας στην ουρά. Στη συνέχεια, ο χρήστης χρησιμοποιεί το αρχείο B μέσα στο ίδιο παράθυρο χρόνου με το A. Αμέσως, γίνεται μια προσπέλαση της ουράς και βρίσκουμε ότι το B συνδέεται με το A, οπότε τα δύο αρχεία εισάγονται στο γράφο και συνδέονται μεταξύ τους. Έπειτα το B εισάγεται κι αυτό στην ουρά. Ακολούθως, γίνεται μια αίτηση στο αρχείο C. Αφού προσπελαστεί η ουρά, το αρχείο C εισέρχεται στο γράφο, συνδέεται με τα A και B, αφού βρίσκεται στο ίδιο παράθυρο χρόνου με αυτά και εισάγεται στην ουρά. Τέλος γίνεται αίτηση στο αρχείο D. Αφού προσπελαστεί η ουρά, προκύπτει ότι το D δε συνδέεται με το A και το B, αφού δε βρίσκονται στο ίδιο παράθυρο. Αμέσως το A και το B απομακρύνονται από την ουρά. Ωστόσο, το αρχείο C βρίσκεται στο ίδιο παράθυρο με το D, οπότε το D εισάγεται στον γράφο και συνδέεται με αυτό.

Τέλος, εκτός από τις παραπάνω ρουτίνες διαχείρισης του γράφου, υπάρχουν και άλλες οι οποίες υλοποιούν λειτουργίες του γράφου, όπως δημιουργία, διαγραφές, εισαγωγές κόμβων, σύνδεση κόμβων, αναζήτηση κόμβου και εκτύπωση των περιεχομένων του γράφου.

5.3 – Περίληψη

Για την υλοποίηση του συστήματος μας χρησιμοποιήσαμε τον κώδικα του Filemon, τον οποίο τροποποιήσαμε σύμφωνα με τις ανάγκες μας. Συγκεκριμένα για να καταγράψουμε τα αρχεία και τα στατιστικά τους, τροποποιήσαμε τις ρουτίνες που διαχειρίζονται αιτήσεις εγγραφής και ανάγνωσης.

Το πρώτο υποσύστημα καταγραφής στατιστικών έχει ως κύρια δομή αποθήκευσης έναν πίνακα κατακερματισμού. Με τον πίνακα κατακερματισμού επιτυγχάνουμε σταθερό χρόνο πρόσβασης στα δεδομένα που περιλαμβάνει ανεξάρτητα από το πλήθος τους. Τα αντικείμενα που αποθηκεύονται στον πίνακα αποτελούνται από το όνομα, τη διαδρομή του αρχείου στο οποίο έγινε η αίτηση, το αντικείμενο του αρχείου και δύο μετρητές για τις αναγνώσεις και εγγραφές που έγιναν σε αυτό. Το δεύτερο υποσύστημα χρησιμοποιεί ως κύρια δομή ένα γράφο συσχετίσεων. Ο γράφος υλοποιείται με λίστες ακμών. Κάθε κόμβος του αναπαριστά ένα αρχείο και οι ακμές του χαρακτηρίζονται από βάρη. Τα αρχεία που σχετίζονται μεταξύ τους συνδέονται με μια ακμή. Εκτός από το γράφο χρησιμοποιούμε μια ουρά που μας βοηθά στη διαδικασία επιλογής των αρχείων που συνδέονται μεταξύ τους. Ανά τακτά χρονικά διαστήματα γίνεται μείωση των βαρών και διαγραφή κόμβων που πληρούν συγκεκριμένες προϋποθέσεις.

Κεφάλαιο 6

Πειράματα

Για τη σύγκριση των δύο διαφορετικών σχεδιασμών του συστήματος κάναμε μετρήσεις που αφορούν την απόδοσή τους. Στα πειράματα που εκτελέσαμε εξετάζουμε τα συστήματα που υλοποιήσαμε από δύο σκοπιές. Πρώτον, εξετάζουμε την επιβάρυνση του υπόλοιπου συστήματος, αυτήν δηλαδή που καταλαβαίνει ο χρήστης καθώς εργάζεται στο σύστημα και δεύτερον εξετάζουμε την επιβάρυνση των δομών των συστημάτων που υλοποιήθηκαν, δηλαδή του πίνακα κατακερματισμού και του γράφου. Η εξέταση αυτή θα μας βοηθήσει να συγκρίνουμε τις δύο δομές μεταξύ τους, να εξηγήσουμε πού οφείλεται η επιβάρυνση που εισάγει στο σύστημα η κάθε μια και τέλος να αποφασίσουμε ποια είναι η καλύτερη. Τα πειράματα έγιναν για διαφορετικές παραμέτρους που αφορούν το μέγεθος του πίνακα κατακερματισμού στη μια περίπτωση και το χρονικό παράθυρο συσχετίσεων ή ενημέρωσης ακμών του γράφου στην άλλη. Τα αποτελέσματά τους παρουσιάζονται σε γραφικές παραστάσεις και αναλύονται.

6.1 – Περιβάλλον εκτέλεσης πειραμάτων

Οργανώνουμε τα πειράματα που εκτελούμε σε δύο κατηγορίες. Η πρώτη κατηγορία περιλαμβάνει τα πειράματα που αναφέρονται στην απόδοση του συστήματος που υλοποιήθηκε κατά τη χρήση του σε προσωπικό υπολογιστή. Τα πειράματα που ανήκουν σε αυτή την κατηγορία επιλέχτηκαν με σκοπό να αποδώσουν μια άποψη για την επιβάρυνση που προκαλούν τα δύο συστήματα (καταγραφή στατιστικών και συνδέσεων) στις καθημερινές εργασίες του χρήστη. Για το σκοπό αυτό, χρησιμοποιούμε το εργαλείο PCMARK05. Το PCMARK05 είναι ένα εργαλείο που προσομοιώνει καθημερινές εργασίες, όπως την εκκίνηση του λειτουργικού συστήματος και διάφορων εφαρμογών, τη διαδικασία ελέγχου για ιούς, την εγγραφή σε αρχεία και τέλος τη γενική χρήση του προσωπικού υπολογιστή. Στόχος του είναι να μετρήσει το ρυθμό μεταφοράς δεδομένων κατά τις παραπάνω εργασίες.

Επειδή το PCMARK05 χρησιμοποιεί ένα προσωρινό αρχείο για να πάρει μετρήσεις του ρυθμού μεταφοράς δεδομένων, δε μας δίνει πλήρη εικόνα για την απόδοση του συστήματος σε συνθήκες υψηλού φόρτου. Έτσι, συμπεριλαμβάνουμε στην πρώτη κατηγορία άλλες δύο μεθόδους. Η πρώτη βασίζεται σε ένα εργαλείο microbenchmark, το οποίο δημιουργήσαμε για τις ανάγκες αξιολόγησης των δύο σχεδιασμών. Δημιουργεί προεπιλεγμένο πλήθος αρχείων, εκτελεί αιτήσεις εγγραφής και μετρά το χρόνο που χρειάστηκε να ολοκληρωθεί η διαδικασία. Με αυτόν τον τρόπο εξετάζουμε πως ανταποκρίνονται οι δομές των συστημάτων καταγραφής στατιστικών και συνδέσεων όταν διαχειρίζονται μεγάλο αριθμό αρχείων και πόσο επιβαρύνουν το σύστημα. Η δεύτερη μέθοδος που ακολουθούμε περιλαμβάνει τη μεταγλώττιση του πυρήνα 2.4.37 του Linux, κατά την οποία μετράμε το χρόνο που πέρασε μέχρι να ολοκληρωθεί.

Η δεύτερη κατηγορία περιλαμβάνει τα πειράματα που εξετάζουν την απόδοση του συστήματος καταγραφής στατιστικών και συνδέσεων σε ένα περιβάλλον διακομιστή αποθήκευσης. Το εργαλείο που χρησιμοποιούμε είναι το bonnie++. Πρόκειται για ένα εργαλείο το οποίο μετράει τη ρυθμαπόδοση του συστήματος αρχείων κατά την εκτέλεση διάφορων εργασιών σε αρχεία, όπως δημιουργία, διαγραφή, εγγραφή και ανάγνωση.

Το σύστημα που χρησιμοποιήσαμε για την εκτέλεση των πειραμάτων περιλαμβάνει τον επεξεργαστή Pentium 4 – 3.4GHz με κρυφή μνήμη 1024KB, 2048MB μνήμη Ram και σκληρό δίσκο SATA χωρητικότητας 500GB και ταχύτητας 7200rpm. Στο σύστημα είναι εγκατεστημένο το λειτουργικό σύστημα Windows XP Professional με το Service Pack 3.

6.2 – Microbenchmark

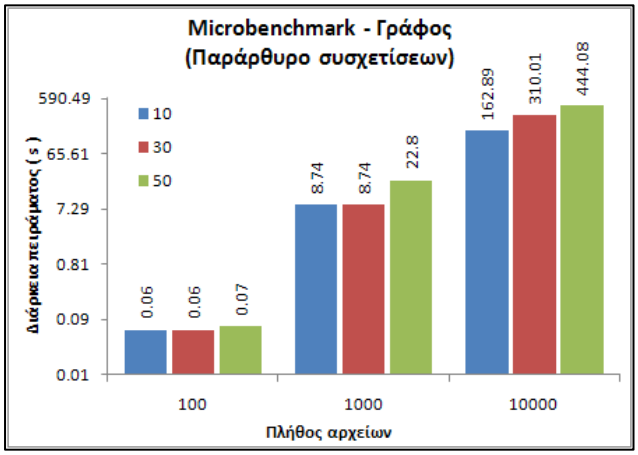
Για τις ανάγκες των πειραμάτων δημιουργήσαμε ένα εργαλείο το οποίο δημιουργεί ένα πλήθος αρχείων (100, 1.000 και 10.000 κάθε φορά). Το εργαλείο γράφει σε κάθε αρχείο που δημιουργεί από ένα χαρακτήρα και μετράει το χρόνο που χρειάστηκε να ολοκληρωθεί η παραπάνω διαδικασία. Αυτό μας επιτρέπει να έχουμε μια καλύτερη εικόνα όσον αφορά την επιβάρυνση κάθε δομής και του συστήματος στην περίπτωση που χρησιμοποιείται ένας αρκετά μεγάλος αριθμός αρχείων. Όταν χρησιμοποιείται ο γράφος ως δομή αποθήκευσης σε μεγάλο φόρτο τείνει να εισάγει σημαντικές καθυστερήσεις σε σχέση με τον πίνακα κατακερματισμού. Αυτό έχει σαν αποτέλεσμα σημαντική μείωση των επιδόσεων του συστήματος.

6.2.1 – Επιβάρυνση του συστήματος

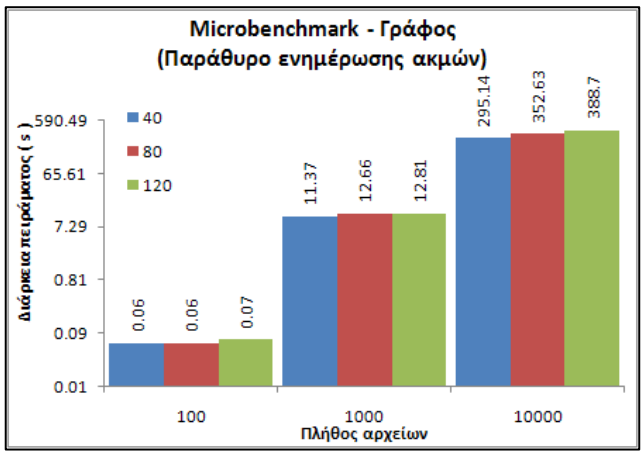
Η διάρκεια εκτέλεσης του πειράματος μας δείχνει πόσο έχει επιβαρυνθεί το σύστημα. Όταν η δομή φορτώνεται με μεγαλύτερο αριθμό αρχείων, γίνεται πιο αργή με αποτέλεσμα να επιβαρύνει και τη συνολική λειτουργία του συστήματος, οπότε και το πείραμα αργεί να ολοκληρωθεί.

Αρχικά εξετάζουμε το γράφο αλλάζοντας την τιμή του παραθύρου συσχετίσεων και κρατώντας σταθερή την τιμή του παραθύρου ενημέρωσης ακμών. Καθώς μεγαλώνει το πλήθος των αρχείων παρατηρούμε μια αύξηση του χρόνου ολοκλήρωσης του πειράματος. Μεγάλο παράθυρο συσχετίσεων σημαίνει περισσότερες συνδέσεις μεταξύ αρχείων, συνεπώς επιβαρύνεται περισσότερο η δομή, η οποία με τη σειρά της επηρεάζει το σύστημα κάνοντας το πιο αργό (Σχήμα 6.1 - α). Ίδια συμπεριφορά ακολουθούν και τα αποτελέσματα όταν μεταβάλλουμε το παράθυρο ενημέρωσης ακμών ενώ έχουμε μια συγκεκριμένη τιμή για το παράθυρο συσχετίσεων (Σχήμα 6.1 - β). Αντίθετα από την δομή του γράφου, ο πίνακας κατακερματισμού έχει διαφορετική συμπεριφορά. Ναι μεν αυξάνεται ο χρόνος διεκπεραίωσης του πειράματος για μεγαλύτερο πλήθος αρχείων, αλλά για μεγαλύτερα μεγέθη πίνακα πετυχαίνουμε καλύτερο χρόνο για το ίδιο πλήθος αρχείων. Για παράδειγμα, όταν δημιουργούμε 10.000 αρχεία, χρησιμοποιώντας πίνακα με 1.000.000 θέσεις, ο χρόνος ολοκλήρωσης είναι μικρότερος απ’ ότι όταν χρησιμοποιούμε πίνακα με 500.000 ή 1024 θέσεις (Σχήμα 6.1 - γ).

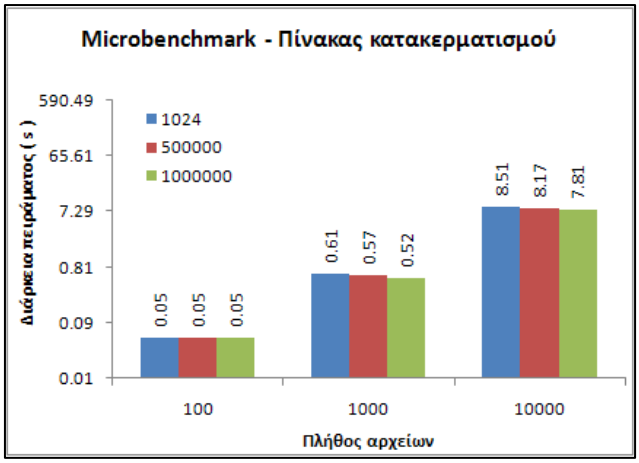
Αξιοσημείωτη είναι η διαφορά που παρουσιάζει η διάρκεια του πειράματος όταν έχουμε ως δομή αποθήκευσης τον πίνακα κατακερματισμού σε σχέση με εκείνη του γράφου. Για μικρό πλήθος αρχείων η διαφορά αυτή δεν είναι εμφανής. Όσο μεγαλώνει όμως ο αριθμός των αρχείων ο χρόνος που πετυχαίνει ο πίνακας κατακερματισμού, γύρω στα 8 δευτερόλεπτα, είναι εμφανώς πολύ καλύτερος από εκείνον των 444 ή των 388 δευτερολέπτων του γράφου. Όταν ο γράφος πρέπει να διαχειριστεί ένα αρκετά μεγάλο πλήθος αρχείων γίνεται πολύ αργός σε σχέση με τον πίνακα κατακερματισμού, μειώνοντας την απόδοση του συστήματος, γι’ αυτό και βλέπουμε αυτή τη μεγάλη διαφορά στο χρόνο εκτέλεσης του πειράματος.



(α)



(β)



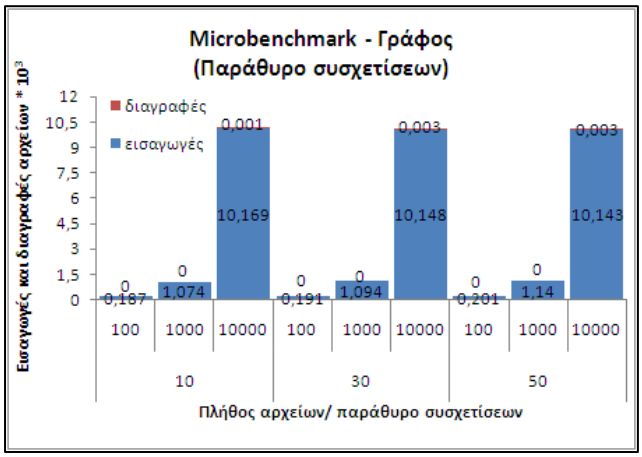
(γ)

Σχήμα 6.1 - Εξετάζουμε το χρόνο ολοκλήρωσης του πειράματος (δημιουργία 100, 1.000 και 10.000 αρχεία και εγγραφή ενός χαρακτήρα σε κάθε αρχείο), ***(α)*** *χρησιμοποιώντας τρεις διαφορετικές τιμές για το παράθυρο συσχετίσεων (10, 30 και 50 δευτερόλεπτα κάθε φορά) και σταθερό το παράθυρο ενημέρωσης ακμών (60 δευτερόλεπτα),* ***(β)*** *χρησιμοποιώντας τρεις διαφορετικές τιμές για το παράθυρο ενημέρωσης ακμών (40, 80 και 120 δευτερόλεπτα κάθε φορά) και σταθερό το παράθυρο συσχετίσεων (30 δευτερόλεπτα)* ***(γ)*** *χρησιμοποιώντας διαφορετικά μεγέθη του πίνακα κατακερματισμού (1024, 500.000 και 1.000.000). Όσο μικρότερος χρόνος τόσο καλύτερα*

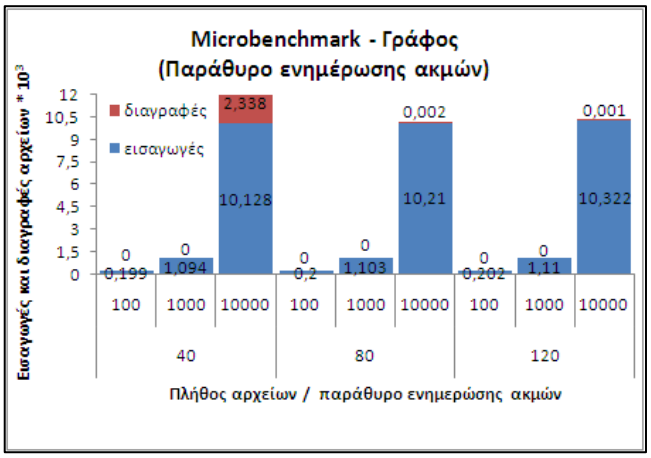
6.2.2 – Επιβάρυνση πίνακα κατακερματισμού και γράφου

Στα υπόλοιπα πειράματα αυτού του είδους μετράμε τις εισαγωγές κόμβων, τις αναγνώσεις και εγγραφές αρχείων στο γράφο και στον πίνακα κατακερματισμού. Επιπλέον, για τον γράφο μετράμε και το πλήθος των συνδέσεων (ακμών) που έχουν δημιουργηθεί ή ενημερωθεί στη διάρκεια του πειράματος και το πλήθος τω κόμβων που διαγράφηκαν. Τα αρχεία που εισάγονται και στις δύο δομές αποτελούνται όχι μόνο από τα αρχεία που δημιουργούνται από το microbenchmark αλλά και από εκείνα που χρησιμοποιεί το λειτουργικό σύστημα κάθε φορά.

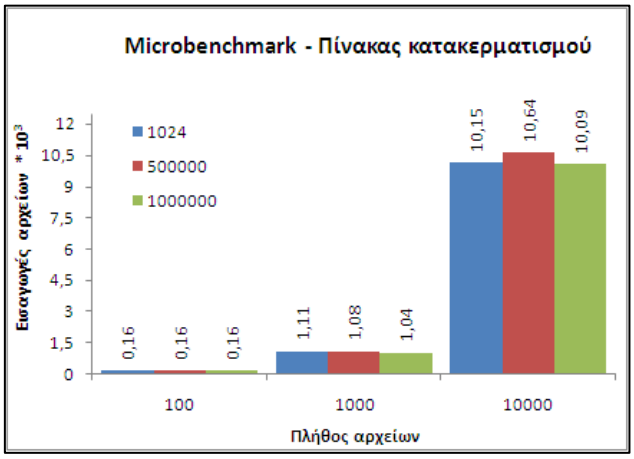
Στην περίπτωση του γράφου όπου αλλάζουμε την τιμή του παραθύρου συσχετίσεων, βλέπουμε ότι οι κόμβοι που εισήχθησαν στο γράφο κατά τη διάρκεια του πειράματος αυξάνονται όταν μεγαλώνει το παράθυρο συσχετίσεων. Κι αυτό διότι για μεγάλο παράθυρο συσχετίσεων, προκύπτουν περισσότερα αρχεία που σχετίζονται μεταξύ τους, οπότε έχουμε περισσότερες εισαγωγές στο γράφο (Σχήμα 6.2 - α). Το ίδιο παρατηρείται όταν μεταβάλλουμε τη τιμή του παραθύρου ενημέρωσης των ακμών, αφού για μεγάλο παράθυρο αργεί να γίνει ενημέρωση των βαρών των ακμών και επομένως διαγραφή των αντίστοιχων κόμβων (Σχήμα 6.2 - β).



(α)



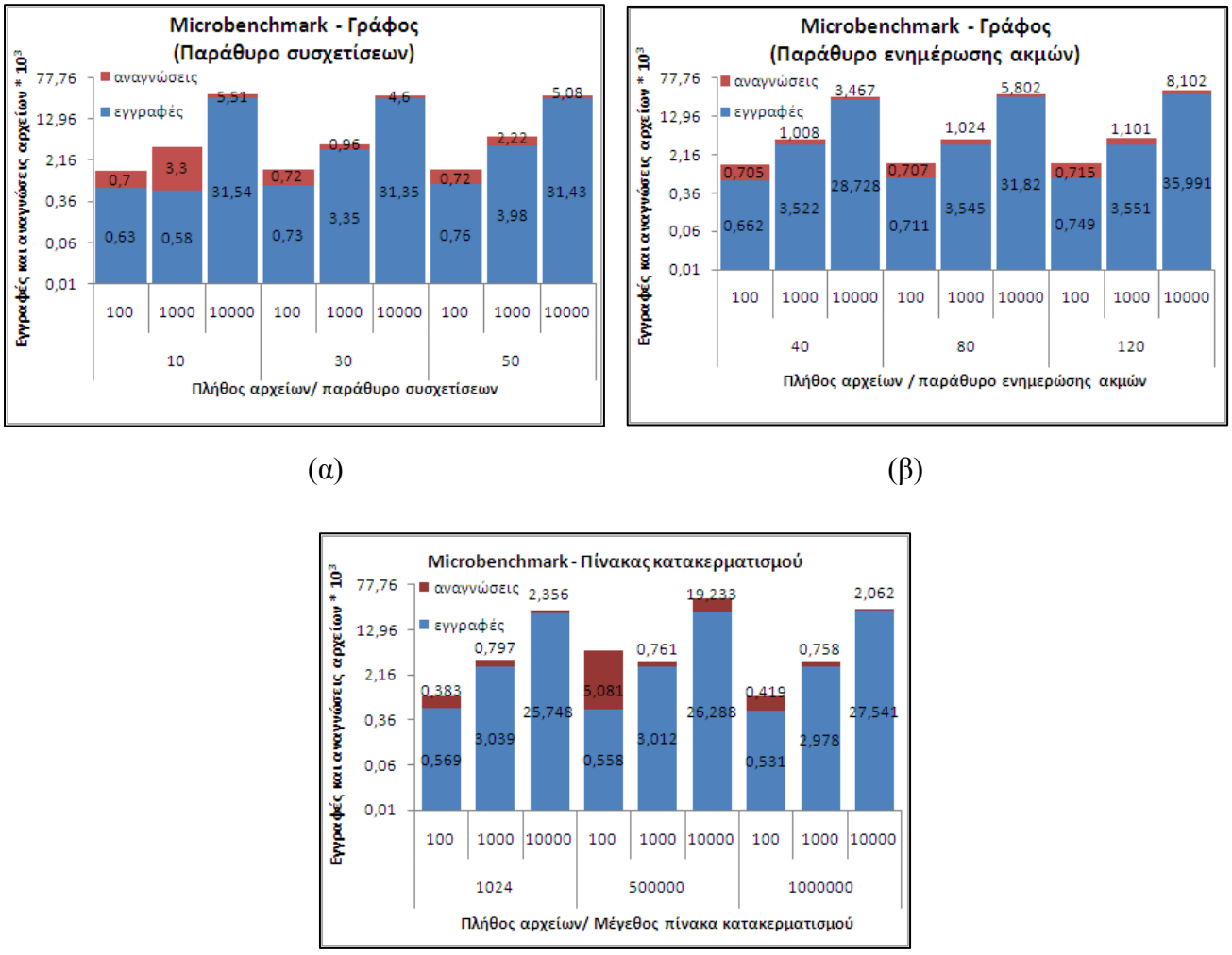
(β)



(γ)

Σχήμα 6.2 - Εξετάζουμε το πλήθος αρχείων που εισήχθησαν σε κάθε δομή κατά τη διάρκεια εκτέλεσης του πειράματος (δημιουργία 100, 1.000 και 10.000 αρχεία και εγγραφή ενός χαρακτήρα σε κάθε αρχείο), ***(α)*** *χρησιμοποιώντας τρεις διαφορετικές τιμές για το παράθυρο συσχετίσεων (10, 30 και 50 δευτερόλεπτα κάθε φορά) και σταθερό το παράθυρο ενημέρωσης ακμών (60 δευτερόλεπτα),* ***(β)*** *χρησιμοποιώντας τρεις διαφορετικές τιμές για το παράθυρο ενημέρωσης ακμών (40, 80 και 120 δευτερόλεπτα κάθε φορά) και σταθερό το παράθυρο συσχετίσεων (30 δευτερόλεπτα)* ***(γ)*** *χρησιμοποιώντας διαφορετικά μεγέθη του πίνακα κατακερματισμού (1024, 500.000 και 1.000.000).*

Από την άλλη μεριά, οι εισαγωγές αρχείων που έγιναν στον πίνακα κατακερματισμού αφορούν όλα τα αρχεία στα οποία έγιναν αιτήσεις εγγραφής ή ανάγνωσης (Σχήμα 6.2 - γ). Παρατηρούμε ότι ο αριθμός των κόμβων που εισήχθησαν, τόσο στο γράφο, όσο και στον πίνακα κατακερματισμού, δεν διαφέρουν κατά πολύ (Σχήμα 6.2). Επίσης ο χρόνος εκτέλεσης του πειράματος (Σχήμα 6.1), είναι αρκετά μεγαλύτερος στο γράφο απ' ότι στον πίνακα κατακερματισμού ενώ ο όγκος δεδομένων που έχουν οι δύο δομές είναι σχεδόν ίδιος. Δεδομένων των παραπάνω ο γράφος διαχειρίζεται πιο αργά την πληροφορία που αποθηκεύει.



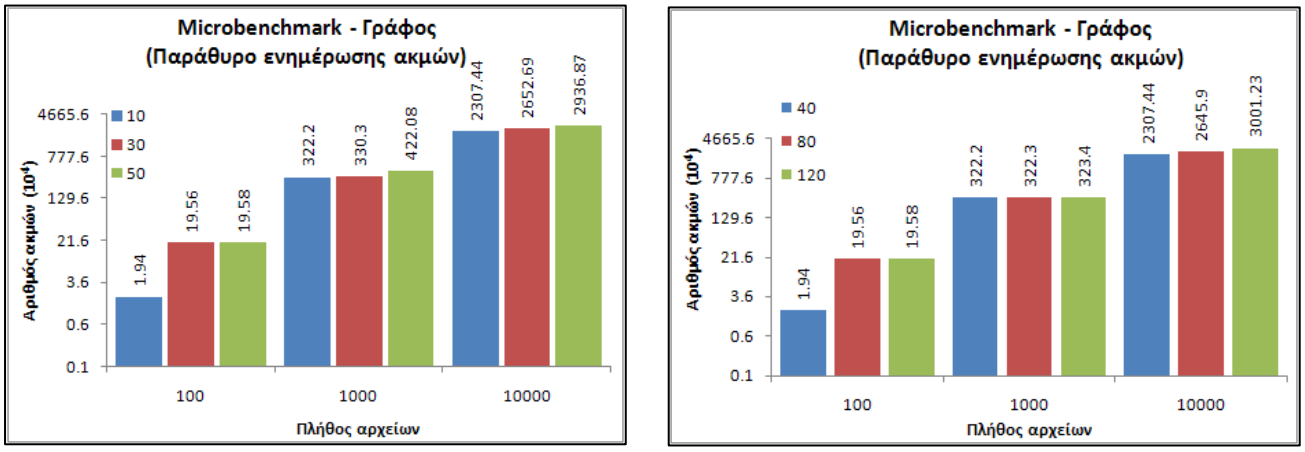
(α) (β)

(γ)

Σχήμα 6.3 - Εξετάζουμε το πλήθος των εγγραφών και αναγνώσεων στα αρχεία κάθε δομής κατά τη διάρκεια εκτέλεσης του πειράματος (δημιουργία 100, 1.000 και 10.000 αρχεία και εγγραφή ενός χαρακτήρα σε κάθε αρχείο), ***(α)*** *χρησιμοποιώντας τρεις διαφορετικές τιμές για το παράθυρο συσχετίσεων (10, 30 και 50 δευτερόλεπτα κάθε φορά) και σταθερό το παράθυρο ενημέρωσης ακμών (60 δευτερόλεπτα),* ***(β)*** *χρησιμοποιώντας τρεις διαφορετικές τιμές για το παράθυρο ενημέρωσης ακμών (40, 80 και 120 δευτερόλεπτα κάθε φορά) και σταθερό το παράθυρο συσχετίσεων (30 δευτερόλεπτα)* ***(γ)*** *χρησιμοποιώντας διαφορετικά μεγέθη του πίνακα κατακερματισμού (1024, 500.000 και 1.000.000).*

Οι εγγραφές και αναγνώσεις που γίνονται στα αρχεία που έχουν εισαχθεί στο γράφο και στον πίνακα κατακερματισμού, προκύπτουν τόσο από το εργαλείο που δημιουργήσαμε (microbenchmark) όσο και από τις αιτήσεις του λειτουργικού συστήματος στα αρχεία που χρησιμοποιεί. Γι' αυτό ο αριθμός τους είναι αυξημένος σε σχέση με τον αριθμό των αρχείων που δημιουργούμε (Σχήμα 6.3). Επίσης, ο αριθμός τους, είναι λίγο μεγαλύτερος στο γράφο από τον πίνακα κατακερματισμού γιατί το πείραμα με το γράφο διαρκεί περισσότερο οπότε καταγράφονται περισσότερες εγγραφές και αναγνώσεις.

Τέλος, εξετάζουμε για τη δομή του γράφου πόσες συσχετίσεις αρχείων γίνονται. Στην πρώτη περίπτωση, που μεταβάλλουμε το παράθυρο συσχετίσεων, ο αριθμός των ακμών του γράφου αυξάνεται, αφού αυξάνοντας το παράθυρο έχουμε περισσότερες συσχετίσεις μεταξύ αρχείων (Σχήμα 6.4 - α). Την ίδια συμπεριφορά παρουσιάζει ο γράφος και όταν κρατάμε σταθερό το παράθυρο συσχετίσεων και μεγαλώνουμε την τιμή του παραθύρου ενημέρωσης ακμών (Σχήμα 6.4 - β).



(α) (β)

Σχήμα 6.4 - Εξετάζουμε το πλήθος των ακμών που δημιουργήθηκαν ή ενημερώθηκαν κατά τη διάρκεια εκτέλεσης του πειράματος (δημιουργία 100, 1.000 και 10.000 αρχεία και εγγραφή ενός χαρακτήρα σε κάθε αρχείο), ***(α)*** *χρησιμοποιώντας τρεις διαφορετικές τιμές για το παράθυρο συσχετίσεων (10, 30 και 50 δευτερόλεπτα κάθε φορά) και σταθερό το παράθυρο ενημέρωσης ακμών (60 δευτερόλεπτα),* ***(β)*** *χρησιμοποιώντας τρεις διαφορετικές τιμές για το παράθυρο ενημέρωσης ακμών (40, 80 και 120 δευτερόλεπτα κάθε φορά) και σταθερό το παράθυρο συσχετίσεων (30 δευτερόλεπτα).*

6.3 – PCMARK05

Θέλοντας να αξιολογήσουμε την επιβάρυνση που αντιλαμβάνεται ο χρήστης καθώς εργάζεται στον προσωπικό του υπολογιστή χρησιμοποιούμε το PCMARK05. Το PCMARK05 προσομοιώνει διάφορες εργασίες και μετράει τον ρυθμό μεταφοράς δεδομένων στο σύστημα, κατά τη διάρκεια εκτέλεσης του πειράματος, για τις περιπτώσεις εκκίνησης του λειτουργικού συστήματος,

εκκίνησης εφαρμογών, γενικής χρήσης, ελέγχου για ιούς και τέλος για την περίπτωση εγγραφής σε κάποιο αρχείο. Για να έχουμε πιο ακριβή αποτελέσματα, επαναλαμβάνουμε κάθε πείραμα 5 φορές και παίρνουμε το μέσο όρο.

Εκτελούμε τα πειράματα για τον γράφο και τον πίνακα κατακερματισμού. Για το γράφο αρχικά κρατάμε σταθερό το παράθυρο συσχετίσεων (30 δευτερόλεπτα) και μεταβάλλουμε το παράθυρο ενημέρωσης ακμών (60, 80 και 120 δευτερόλεπτα). Στη συνέχεια αυξάνουμε το παράθυρο συσχετίσεων (10, 30 και 60 δευτερόλεπτα) και κρατάμε σταθερό το παράθυρο ενημέρωσης ακμών (80 δευτερόλεπτα). Για τον πίνακα κατακερματισμού χρησιμοποιούμε 1.024, 500.000 και 1.000.000 θέσεις κάθε φορά. Σε κάποιες περιπτώσεις τα αποτελέσματα για διαφορετικές παραμέτρους δεν παρουσιάζουν μεγάλες αποκλίσεις, γι' αυτό και στα διαγράμματα απεικονίζεται μια αντιπροσωπευτική τιμή.

Είναι σημαντικό επίσης να αναφέρουμε ότι πριν την εκτέλεση του κάθε πειράματος, «ζεσταίνουμε» το σύστημα κάνοντας αιτήσεις σε αρχεία ώστε οι δομές των συστημάτων καταγραφής στατιστικών και συνδέσεων να αποκτήσουν κάποια αρχικά δεδομένα.

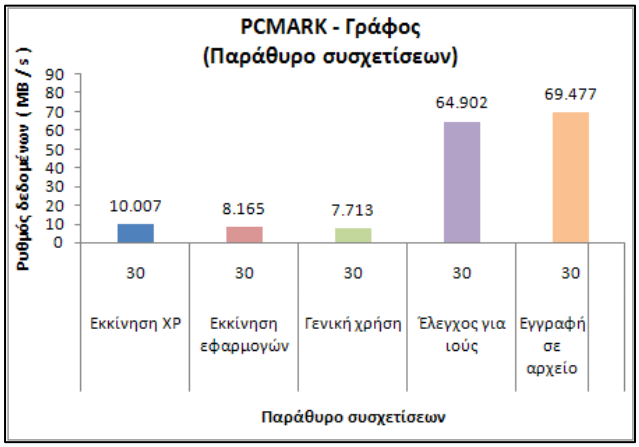
6.3.1 – Επιβάρυνση του συστήματος

Ο ρυθμός μεταφοράς δεδομένων αναφέρεται στην ταχύτητα με την οποία τα δεδομένα μεταφέρονται από και προς τον δίσκο. Αρχικά εξετάζουμε τη συμπεριφορά του συστήματος όταν γίνεται χρήση του γράφου ως δομή αποθήκευσης. Χρησιμοποιούμε διαφορετικές τιμές για το μέγεθος του παραθύρου συσχετίσεων και κρατάμε σταθερή την τιμή του παραθύρου ενημέρωσης των ακμών. Παρατηρούμε ότι ο ρυθμός μεταφοράς δεδομένων δεν έχει μεγάλες αποκλίσεις για τις διαφορετικές τιμές του παραθύρου συσχετίσεων (Σχήμα 6.5 - α).

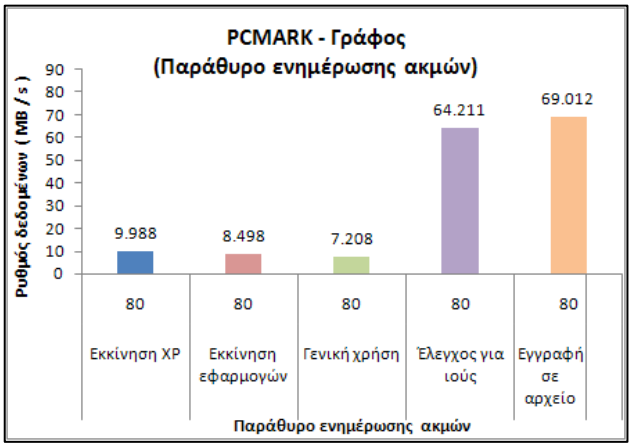
Μεταβάλλοντας το παράθυρο ενημέρωσης ακμών και κρατώντας σταθερή την τιμή του παραθύρου συσχετίσεων αυτή τη φορά, από τις μετρήσεις παίρνουμε σχεδόν ίδια αποτελέσματα (Σχήμα 6.5 - β). Για το λόγο αυτό, παρουσιάζουμε μια αντιπροσωπευτική τιμή για κάθε περίπτωση στα διαγράμματα που ακολουθούν.

Παρόμοια πειράματα εκτελούμε για να εξετάσουμε τη συμπεριφορά του συστήματος όταν γίνεται χρήση του πίνακα κατακερματισμού ως δομή αποθήκευσης. Σύμφωνα με τα αποτελέσματα του πειράματος, για διαφορετικά μεγέθη του πίνακα κατακερματισμού ο ρυθμός είναι σχεδόν ίδιος (Σχήμα 6.5 - γ). Απ' όλες τις παραπάνω μετρήσεις διαπιστώνουμε ότι υπάρχει μια μικρή διαφορά στο ρυθμό μεταφοράς δεδομένων μεταξύ του πίνακα κατακερματισμού και του γράφου, με τον

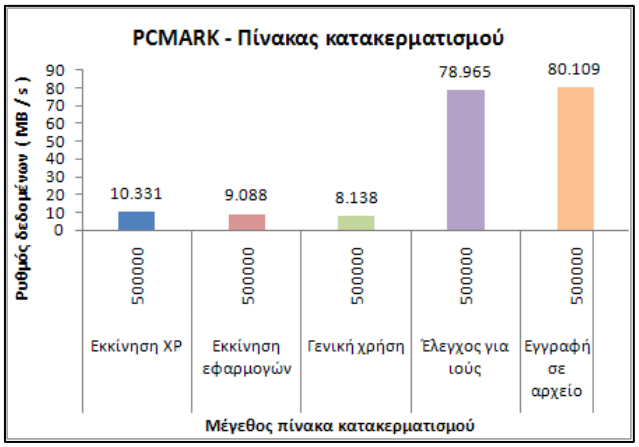
πίνακα κατακερματισμού να πετυχαίνει λίγο καλύτερα αποτελέσματα στο ρυθμό μεταφοράς δεδομένων. Η διαφορά δεν είναι ιδιαίτερα εμφανής γιατί δεν εισάγεται μεγάλο πλήθος αρχείων.



(α)



(β)



(γ)

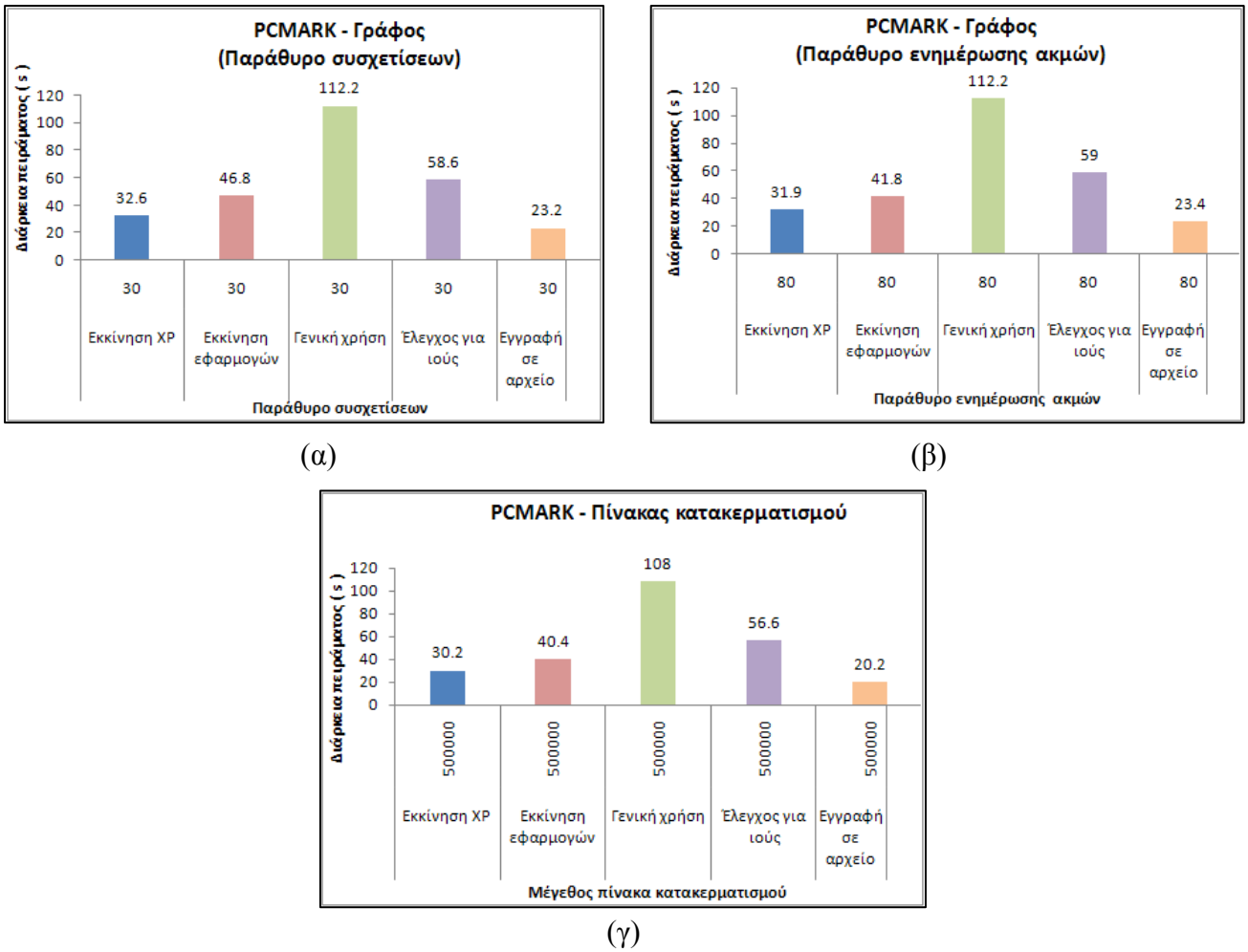
Σχήμα 6.5 - Εξετάζουμε τον ρυθμό μεταφοράς δεδομένων (όσο μεγαλύτερος τόσο καλύτερα), ***(α)*** *χρησιμοποιώντας παράθυρο συσχετίσεων 30 δευτερόλεπτα και παράθυρο ενημέρωσης ακμών 60 δευτερόλεπτα,* ***(β)*** *χρησιμοποιώντας παράθυρο συσχετίσεων 30 δευτερόλεπτα και παράθυρο ενημέρωσης ακμών 80 δευτερόλεπτα,* ***(γ)*** *χρησιμοποιώντας πίνακα κατακερματισμού 500.000 θέσεων, για τις περιπτώσεις εκκίνησης του λειτουργικού συστήματος, εκκίνησης εφαρμογών, γενικής χρήσης, διαδικασίας ελέγχου για ιούς και εγγραφής σε αρχείο.*

Στη συνέχεια εξετάζουμε το χρόνο διάρκειας κάθε πειράματος που επίσης αφορά στην επιβάρυνση του συστήματος. Αρχικά χρησιμοποιούμε το γράφο για να αποθηκεύσουμε τις συνδέσεις μεταξύ των αρχείων. Μεταβάλλοντας την τιμή του παραθύρου συσχετίσεων εκτελούμε τα πειράματα για όλες τις παραπάνω περιπτώσεις και βλέπουμε ότι οι χρόνοι είναι στις ίδιες τιμές όσο μεγαλώνει το παράθυρο συσχετίσεων (Σχήμα 6.6 - α).

Το ίδιο συμβαίνει και όταν κρατάμε σταθερό το παράθυρο συσχετίσεων και αλλάζουμε το παράθυρο ενημέρωσης ακμών όπως φαίνεται στο παραπάνω διάγραμμα (Σχήμα 6.6 - β). Όσο πιο

αργά γίνεται η ενημέρωση, τόσο πιο αργά σβήνονται ακμές και κόμβοι από το γράφο γεγονός που καθιστά αρκετά επιβαρυντική τη δομή για τη λειτουργία του συστήματος.

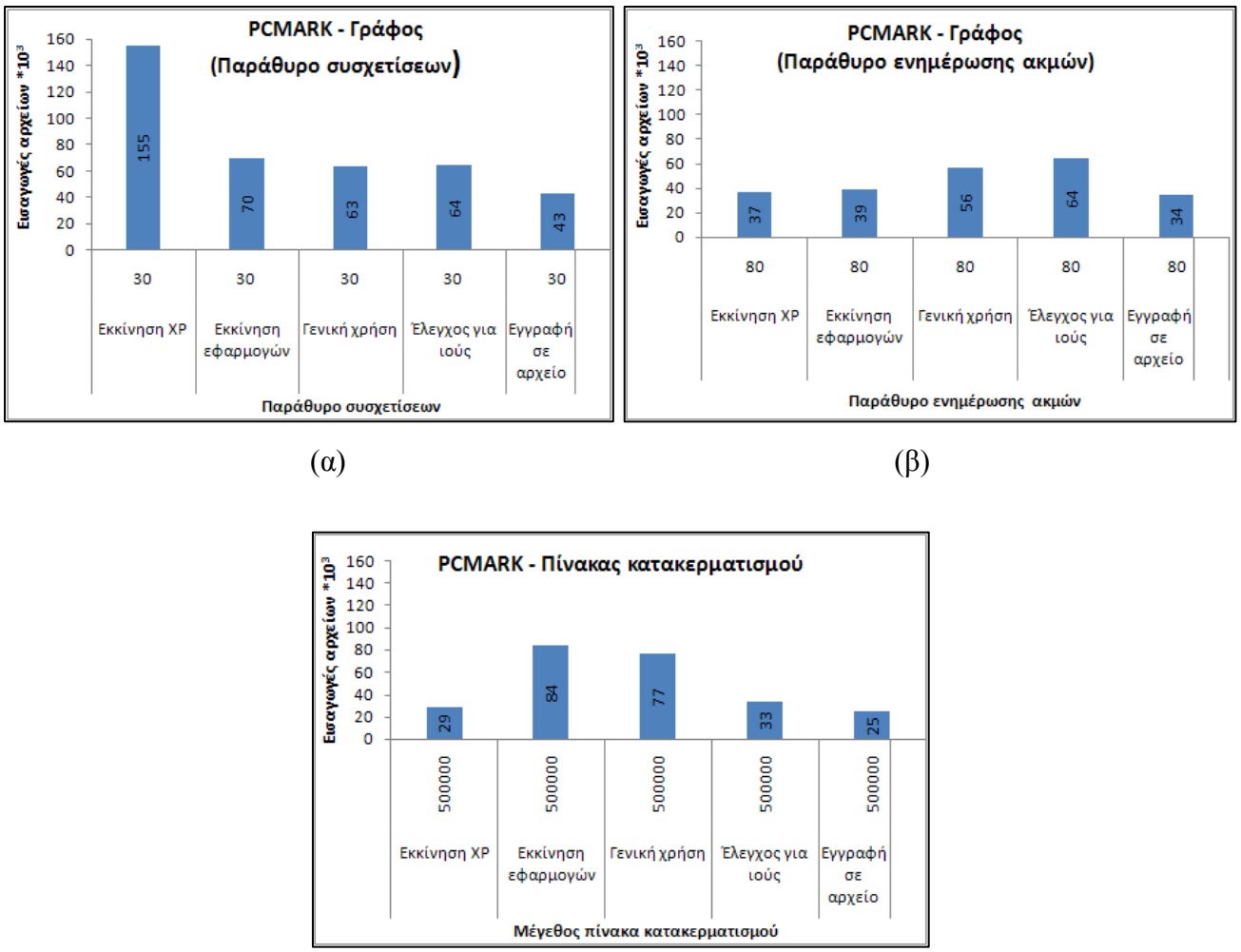
Ακολούθως, χρησιμοποιούμε το σύστημα με δομή αποθήκευσης τον πίνακα κατακερματισμού. Οι μετρούμενοι χρόνοι είναι σχεδόν ίσοι όσο αυξάνει το μέγεθος, και είναι καλύτεροι από εκείνους που παίρνουμε όταν ως δομή αποθήκευσης χρησιμοποιήσαμε το γράφο (Σχήμα 6.6 - γ). Έτσι σε κάθε διάγραμμα παρουσιάζεται μόνο μια τιμή για κάθε περίπτωση.



Σχήμα 6.6 - Εξετάζουμε τη διάρκεια εκτέλεσης κάθε πειράματος (όσο μικρότερη τόσο καλύτερα), ***(α)*** *χρησιμοποιώντας παράθυρο συσχετίσεων 30 δευτερόλεπτα και παράθυρο ενημέρωσης ακμών 60 δευτερόλεπτα,* ***(β)*** *χρησιμοποιώντας παράθυρο συσχετίσεων 30 δευτερόλεπτα και παράθυρο ενημέρωσης ακμών 80 δευτερόλεπτα* ***(γ)*** *χρησιμοποιώντας πίνακα κατακερματισμού 500.000 θέσεων, για τις περιπτώσεις εκκίνησης του λειτουργικού συστήματος, εκκίνησης εφαρμογών, γενικής χρήσης, διαδικασίας ελέγχου για ιούς και εγγραφής σε αρχείο.*

6.3.2 – Επιβάρυνση πίνακα κατακερματισμού και γράφου

Ένας άλλος παράγοντας που θα ήταν χρήσιμο να εξετάσουμε είναι ο αριθμός αρχείων που έχουν εισαχθεί στον πίνακα κατακερματισμού και στο γράφο καθώς και οι εγγραφές και αναγνώσεις που έγιναν στα αρχεία αυτά.



(α) (β)

(γ)

Σχήμα 6.7 - Εξετάζουμε το πλήθος των αρχείων που εισήχθησαν στο γράφο κατά τη διάρκεια του πειράματος, ***(α)*** *χρησιμοποιώντας παράθυρο συσχετίσεων 30 δευτερόλεπτα και παράθυρο ενημέρωσης ακμών 60 δευτερόλεπτα,* ***(β)*** *χρησιμοποιώντας παράθυρο συσχετίσεων 30 δευτερόλεπτα και παράθυρο ενημέρωσης ακμών 80 δευτερόλεπτα* ***(γ)*** *χρησιμοποιώντας πίνακα κατακερματισμού 500.000 θέσεων, για τις περιπτώσεις εκκίνησης του λειτουργικού συστήματος, εκκίνησης εφαρμογών, γενικής χρήσης, διαδικασίας ελέγχου για ιούς και εγγραφής σε αρχείο.*

Στον γράφο, με σταθερό το παράθυρο ενημέρωσης ακμών και μεταβάλλοντας εκείνο των συσχετίσεων, κατά τη διάρκεια κάθε πειράματος το πλήθος των αρχείων που εισήχθησαν δεν έχει μεγάλη διαφορά γι’ αυτό και απεικονίζουμε μια ενδεικτική τιμή (Σχήμα 6.7 - α). Η ίδια

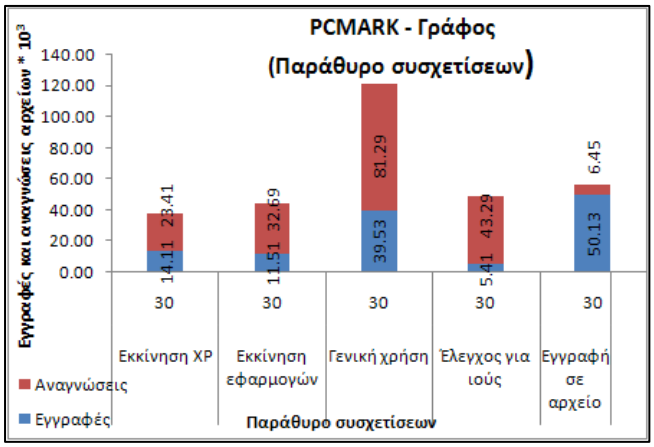
συμπεριφορά παρατηρείται και στο γράφο όπου το παράθυρο ενημέρωσης ακμών μεταβάλλεται ενώ το παράθυρο συσχετίσεων μένει ίδιο (Σχήμα 6.7 - β).

Αντίστοιχα, στα πειράματα που έγιναν χρησιμοποιώντας ως δομή τον πίνακα κατακερματισμού, ο αριθμός των αρχείων που έχουν εισαχθεί στον πίνακα σε κάθε ένα από τα πειράματα δεν έχει μεγάλες αποκλίσεις για διαφορετικά μεγέθη (Σχήμα 6.7 - γ).

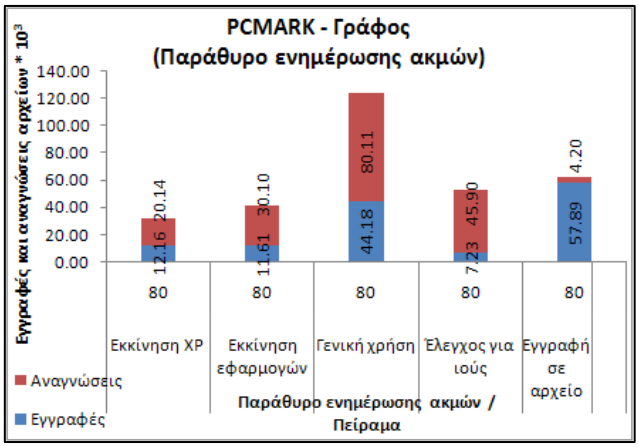
Στη διαμόρφωση των αποτελεσμάτων του πλήθους των αρχείων στο γράφο και στον πίνακα κατακερματισμού επιδρά τόσο η κατάσταση του συστήματος κάθε φορά όσο και ο χρόνος που διήρκησε το πείραμα. Όσο μεγαλύτερη διάρκεια είχε, τόσο περισσότερες αιτήσεις και εισαγωγές έγιναν. Έτσι, στις περιπτώσεις των μετρήσεων με το γράφο τα αρχεία που εισήχθησαν είναι λίγο περισσότερα από εκείνα του πίνακα κατακερματισμού.

Μια παρατήρηση που κάνουμε είναι το γεγονός ότι το πλήθος των αρχείων που εισάγονται στην εκάστοτε δομή δεν επηρεάζει με τον ίδιο τρόπο τις δύο δομές. Πιο συγκεκριμένα, φαίνεται ότι ο γράφος επηρεάζεται πολύ περισσότερο και όσο αυξάνεται ο αριθμός των αρχείων που εισάγονται, τόσο η δομή γίνεται πιο βαριά με αποτέλεσμα να επηρεάζεται η απόδοση του συστήματος. Αντίθετα, ο πίνακας κατακερματισμού δεν επηρεάζεται σχεδόν καθόλου, ειδικά όταν χρησιμοποιούμε πίνακα μεγάλου μεγέθους, διότι οι εσωτερικές λίστες είναι μικρότερες.

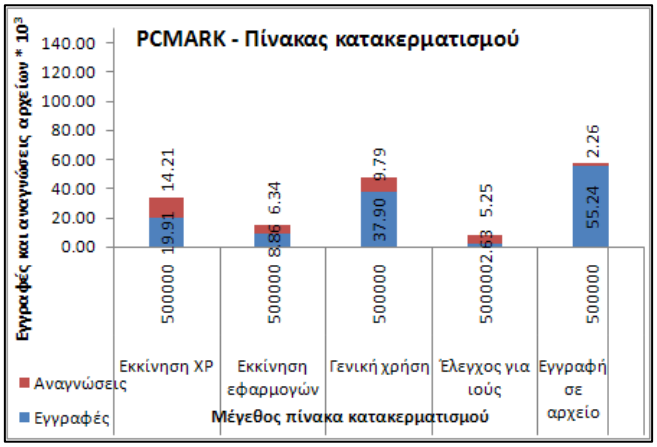
Οι εγγραφές και αναγνώσεις που γίνονται στα αρχεία που έχουν εισαχθεί είτε στον πίνακα κατακερματισμού είτε στο γράφο εξαρτώνται από την κατάσταση του συστήματος κάθε φορά αλλά και από το χρόνο που έκανε το πείραμα να ολοκληρωθεί. Επομένως παρατηρείται ότι οι εγγραφές και αναγνώσεις που γίνονται όταν ως δομή αποθήκευσης χρησιμοποιείται ο γράφος είναι περισσότερες από εκείνες όταν γίνεται χρήση του πίνακα κατακερματισμού. Καταγράφουμε το πλήθος των εγγραφών και των αναγνώσεων που έγιναν συνολικά κατά τη διάρκεια του πειράματος. Χρησιμοποιώντας ως δομή αποθήκευσης τον γράφο με διαφορετικές παραμέτρους στο παράθυρο συσχετίσεων, οι εγγραφές και αναγνώσεις που έγιναν για τις διαφορετικές τιμές του παραθύρου είναι σχεδόν ίσες (Σχήμα 6.8 - α). Το ίδιο συμβαίνει και όταν αυξάνουμε το παράθυρο ενημέρωσης ακμών (Σχήμα 6.8 - β) αλλά και το μέγεθος του πίνακα κατακερματισμού (Σχήμα 6.8 - γ). Για το λόγο αυτό, στα γραφήματα απεικονίζονται αντιπροσωπευτικές τιμές.



(α)



(β)



(γ)

Σχήμα 6.8 - Εξετάζουμε το πλήθος των εγγραφών και αναγνώσεων αρχείων κατά τη διάρκεια του πειράματος, ***(α)*** *χρησιμοποιώντας παράθυρο συσχετίσεων 30 δευτερόλεπτα και παράθυρο ενημέρωσης ακμών 60 δευτερόλεπτα,* ***(β)*** *χρησιμοποιώντας παράθυρο συσχετίσεων 30 δευτερόλεπτα και παράθυρο ενημέρωσης ακμών 80 δευτερόλεπτα* ***(γ)*** *χρησιμοποιώντας πίνακα κατακερματισμού 500.000 θέσεων, για τις περιπτώσεις εκκίνησης του λειτουργικού συστήματος, εκκίνησης εφαρμογών, γενικής χρήσης, διαδικασίας ελέγχου για ιούς και εγγραφής σε αρχείο.*

6.4 – Μεταγλώττιση του Linux

Ένας άλλος τρόπος σύγκρισης της απόδοσης του γράφου και του πίνακα κατακερματισμού είναι παίρνοντας μετρήσεις από τη μεταγλώττιση του πυρήνα 2.4.37 του Linux. Κρατάμε το χρόνο διάρκειας της μεταγλώττισης, τον αριθμό των κόμβων που εισήχθησαν σε κάθε δομή και διαγράφησαν από τον γράφο, καθώς και τις αναγνώσεις και εγγραφές που έγιναν στα αρχεία.

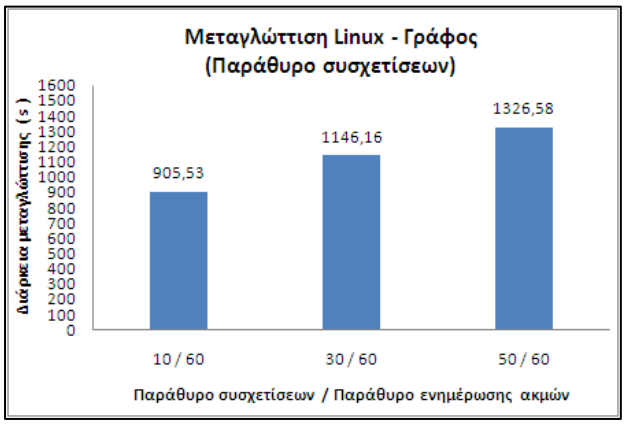
Για τη μεταγλώττιση του πυρήνα του Linux χρησιμοποιείται το περιβάλλον Cygwin. Πριν εκτελέσουμε τη μεταγλώττιση ρυθμίζουμε τον πυρήνα με την εντολή:

$ make menuconfig

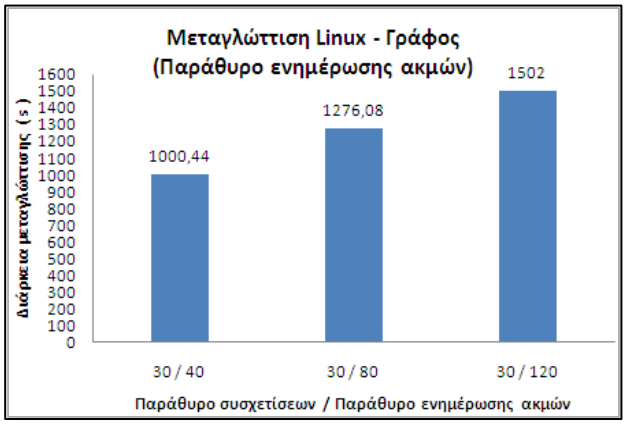
Σε όλες τις περιπτώσεις έχουμε συμπεριλάβει στον πυρήνα τα ίδια χαρακτηριστικά. Στη συνέχεια η μεταγλώττιση γίνεται με την ακόλουθη εντολή:

$ time sh –c ‘make’

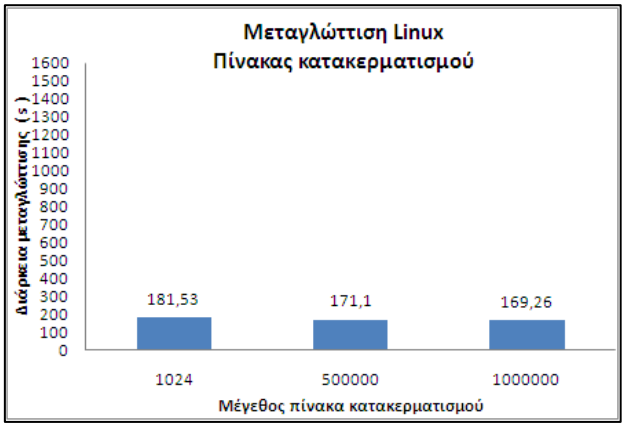
6.4.1 – Επιβάρυνση του συστήματος



(α)



(β)



(γ)

Σχήμα 6.9 - Εξετάζουμε το χρόνο ολοκλήρωσης μεταγλώττισης του Linux 2.4.37 (όσο μικρότερος χρόνος τόσο καλύτερα), ***(α)*** *χρησιμοποιώντας τρεις διαφορετικές τιμές για το παράθυρο συσχετίσεων (10, 30 και 50 δευτερόλεπτα κάθε φορά) και σταθερό το παράθυρο ενημέρωσης ακμών (60 δευτερόλεπτα),* ***(β)*** *χρησιμοποιώντας τρεις διαφορετικές τιμές για το παράθυρο ενημέρωσης ακμών (40, 80 και 120 δευτερόλεπτα κάθε φορά) και σταθερό το παράθυρο συσχετίσεων (30 δευτερόλεπτα)* ***(γ)*** *χρησιμοποιώντας διαφορετικά μεγέθη του πίνακα κατακερματισμού (1024, 500.000 και 1.000.000).*

Η διάρκεια εκτέλεσης του πειράματος είναι μεγαλύτερη στην περίπτωση που χρησιμοποιούμε τον γράφο. Μάλιστα καθώς αυξάνεται το παράθυρο συσχετίσεων ο χρόνος τείνει να

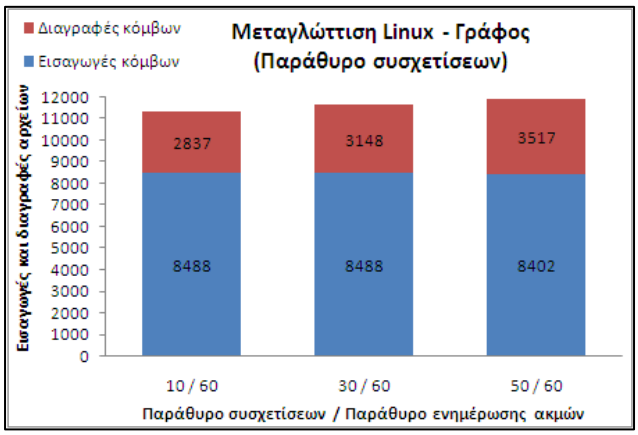
μεγαλώνει αφού έχουμε περισσότερες συσχετίσεις αρχείων και επομένως επιβαρύνεται ο γράφος και κατ' επέκταση το σύστημα (Σχήμα 6.9 - α). Επίσης, αν αυξήσουμε το παράθυρο ενημέρωσης ακμών, τόσο πιο αραιά γίνεται η μείωση βαρών των ακμών και η διαγραφή κόμβων, οπότε ο γράφος γίνεται αρκετά βαρύς επηρεάζοντας το υπόλοιπο σύστημα και ο χρόνος ολοκλήρωσης της μεταγλώττισης αυξάνεται (Σχήμα 6.9 - β).

Αντίθετα, με τη χρήση του πίνακα κατακερματισμού η μεταγλώττιση ολοκληρώνεται σε λιγότερο χρόνο σε σχέση με το γράφο. Επιπλέον, ο χρόνος που διαρκεί η μεταγλώττιση μειώνεται καθώς αυξάνει το μέγεθος του πίνακα, διότι οι εσωτερικές λίστες του πίνακα χρησιμοποιούνται λιγότερο (Σχήμα 6.9 - γ).

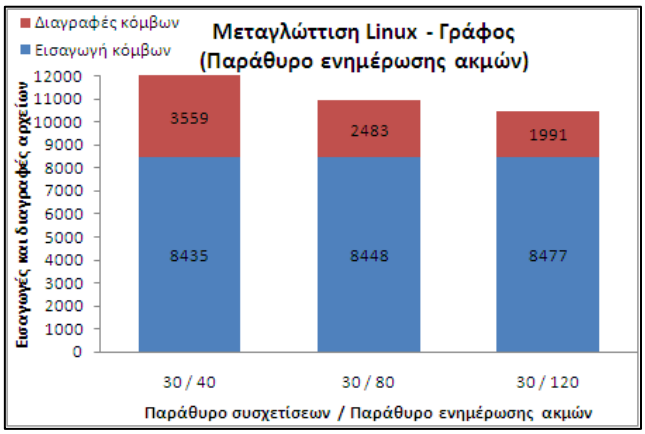
6.4.2 – Επιβάρυνση πίνακα κατακερματισμού και γράφου

Στη συνέχεια, μετράμε τις διαγραφές κόμβων στο γράφο, τις εισαγωγές κόμβων, τις αναγνώσεις και εγγραφές αρχείων και στις δύο δομές. Οι εισαγωγές κόμβων στο γράφο (Σχήμα 6.10 – α και β) είναι σχεδόν ίδιες με εκείνες στον πίνακα κατακερματισμού (Σχήμα 6.10 - γ), αφού και στις δύο περιπτώσεις για τη μεταγλώττιση του πυρήνα χρησιμοποιούνται τα ίδια αρχεία. Ωστόσο, υπάρχει μια μικρή διαφορά στα αρχεία που χρησιμοποιεί το λειτουργικό σύστημα κάθε φορά. Οι διαγραφές κόμβων μειώνονται καθώς μεγαλώνει το παράθυρο ενημέρωσης ακμών, αφού η μείωση των βαρών γίνεται λιγότερες φορές (Σχήμα 6.10 - β).

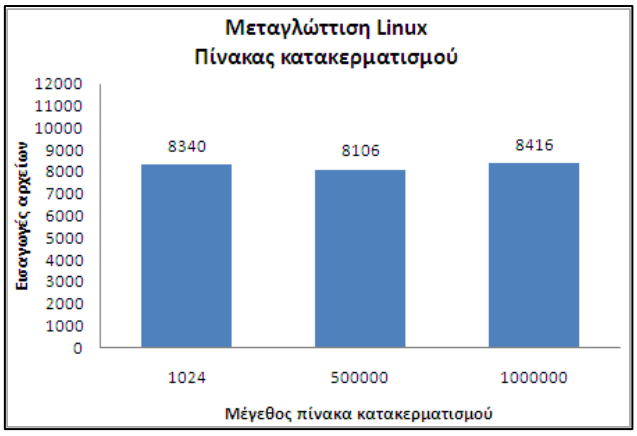
Το πλήθος των αρχείων που εισάγονται στο γράφο και στον πίνακα κατακερματισμού είναι σχεδόν ίδιο. Οπότε, επειδή με τον ίδιο φόρτο, η μεταγλώττιση με το γράφο αργεί περισσότερο φαίνεται ότι ο πίνακας κατακερματισμού δεν επιβαρύνει το σύστημα τόσο πολύ όσο ο γράφος.



(α)



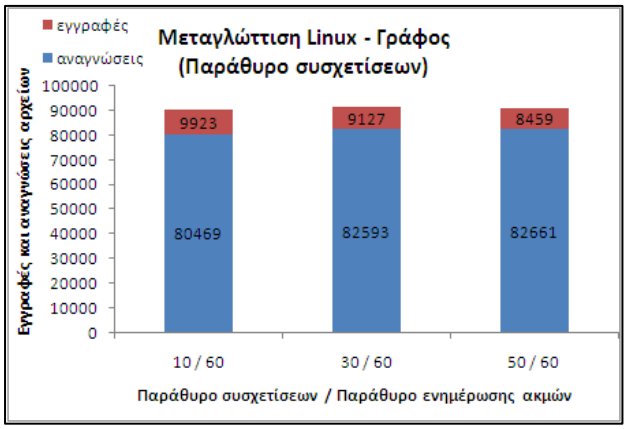
(β)



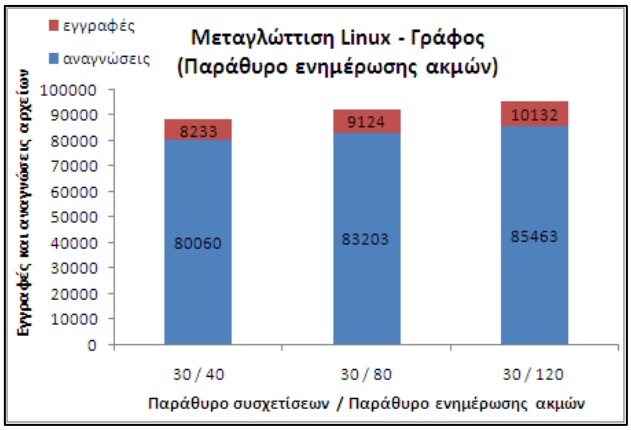
(γ)

*Σχήμα 6.10 - Εξετάζουμε το πλήθος διαγραφών και εισαγωγών στις δομές , **(α)** χρησιμοποιώντας τρεις διαφορετικές τιμές για το παράθυρο συσχετίσεων (10, 30 και 50 δευτερόλεπτα κάθε φορά) και σταθερό το παράθυρο ενημέρωσης ακμών (60 δευτερόλεπτα), **(β)** χρησιμοποιώντας τρεις διαφορετικές τιμές για το παράθυρο ενημέρωσης ακμών (40, 80 και 120 δευτερόλεπτα κάθε φορά) και σταθερό το παράθυρο συσχετίσεων (30 δευτερόλεπτα) **(γ)** χρησιμοποιώντας διαφορετικά μεγέθη του πίνακα κατακερματισμού (1024, 500.000 και 1.000.000).*

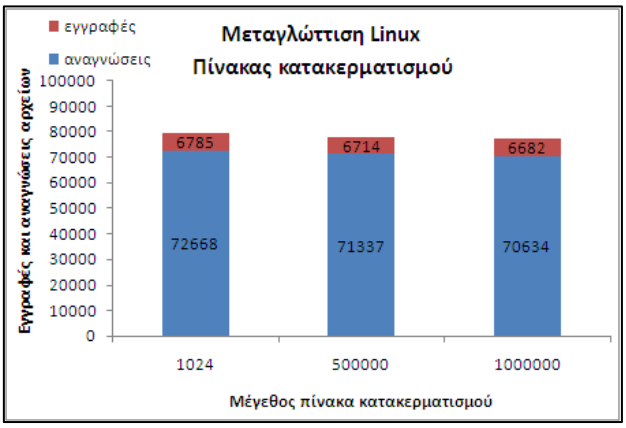
Οι εγγραφές και οι αναγνώσεις στα αρχεία που έχουν εισαχθεί στον γράφο και στον πίνακα κατακερματισμού εξαρτώνται από τα αρχεία που βρίσκονται σε κάθε δομή αλλά και από τις αιτήσεις του συστήματος κάθε φορά. Το πλήθος των εγγραφών είναι μεγαλύτερο στο γράφο (Σχήμα 6.11 - α και β) γιατί η διάρκεια μεταγλώττισης είναι μεγαλύτερη από εκείνη του πίνακα κατακερματισμού (Σχήμα 6.11 - γ).



(α)



(β)



(γ)

Σχήμα 6.11 - Εξετάζουμε το πλήθος εγγραφών και αναγνώσεων στα αρχεία που εισάγονται στις δομές, ***(α)*** *χρησιμοποιώντας τρεις διαφορετικές τιμές για το παράθυρο συσχετίσεων (10, 30 και 50 δευτερόλεπτα κάθε φορά) και σταθερό το παράθυρο ενημέρωσης ακμών (60 δευτερόλεπτα),* ***(β)*** *χρησιμοποιώντας τρεις διαφορετικές τιμές για το παράθυρο ενημέρωσης ακμών (40, 80 και 120 δευτερόλεπτα κάθε φορά) και σταθερό το παράθυρο συσχετίσεων (30 δευτερόλεπτα)* ***(γ)*** *χρησιμοποιώντας διαφορετικά μεγέθη του πίνακα κατακερματισμού (1024, 500.000 και 1.000.000).*

6.5 – Bonnie++

Για να αξιολογήσουμε την επιβάρυνση ενός διακομιστή αποθήκευσης κατά την χρήση των συστημάτων καταγραφής στατιστικών και συνδέσεων χρησιμοποιούμε το bonnie++. Εκτελούμε το bonnie++ ως εξής:

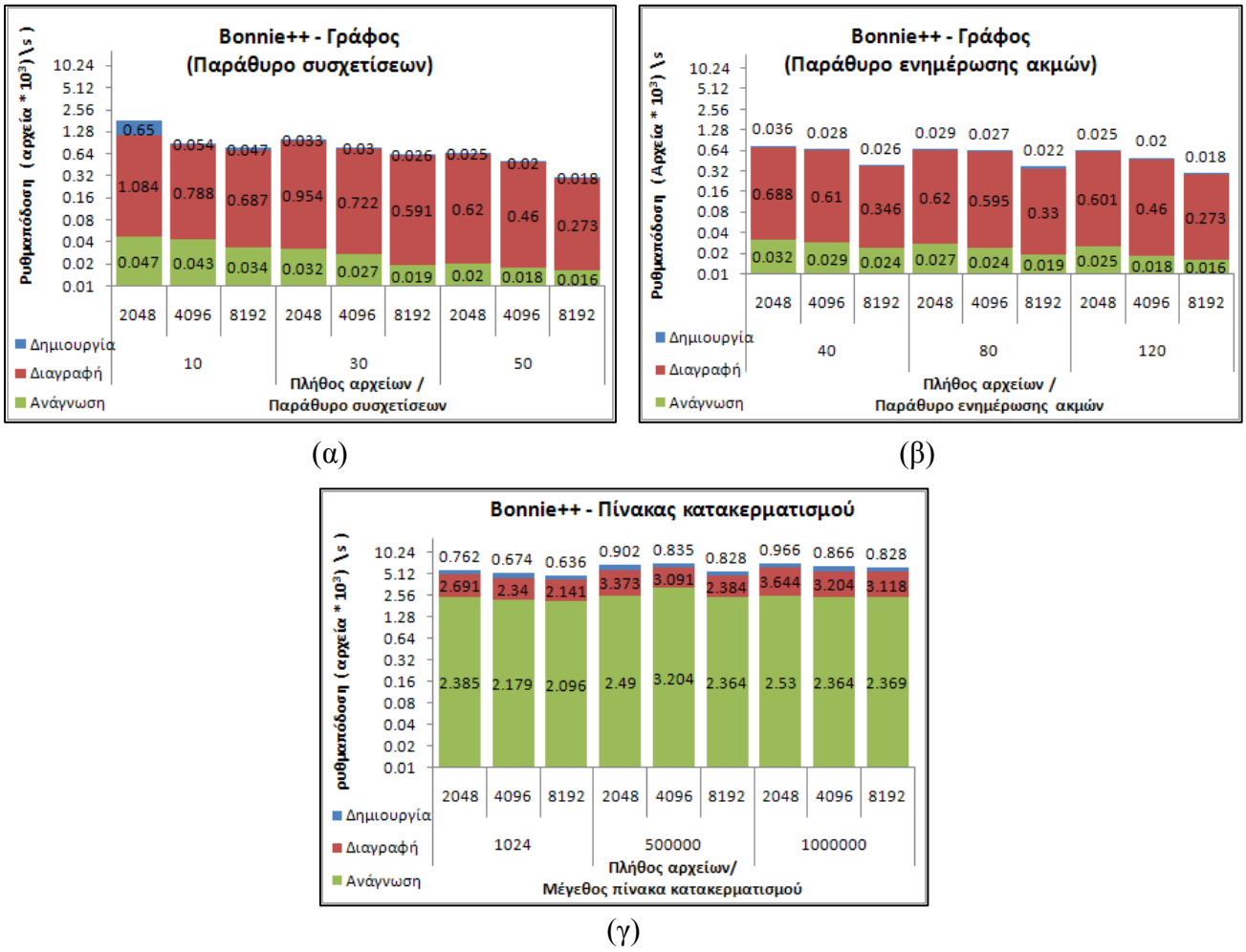
```
bonnie++ -s 0 -f -n x:1:1:1 -u 0
```

Η παραπάνω εντολή εκτελεί το bonnie++ και ενεργοποιεί το πείραμα δημιουργίας αρχείων με το διακόπτη (-n x:1:1:1). Στη θέση του x τοποθετούμε τους αριθμούς 2, 4 και 8 οι οποίοι πληροφορούν

το bonnie++ να δημιουργήσει 2048, 4096 και 8192 αρχεία αντίστοιχα. Επειδή στα πειράματα που εκτελούμε μας ενδιαφέρει να εξετάσουμε πως συμπεριφέρεται το σύστημα κατά τη δημιουργία και χρήση πολλών αρχείων απενεργοποιούμε τα υπόλοιπα πειράματα που εκτελεί το bonnie++ με τους διακόπτες –s 0 και –f. Τέλος, με το διακόπτη –u 0 καθορίζουμε ότι κατά την εκτέλεση του bonnie++ θα χρησιμοποιηθεί ο λογαριασμός του διαχειριστή (Administrator).

Τα αποτελέσματα του πειράματος μας δίνουν τη ρυθμαπόδοση του συστήματος κατά τη δημιουργία, ανάγνωση και διαγραφή αρχείων και το χρόνο που διήρκησε το πείραμα.

6.5.1 – Επιβάρυνση του συστήματος



(α) (β)

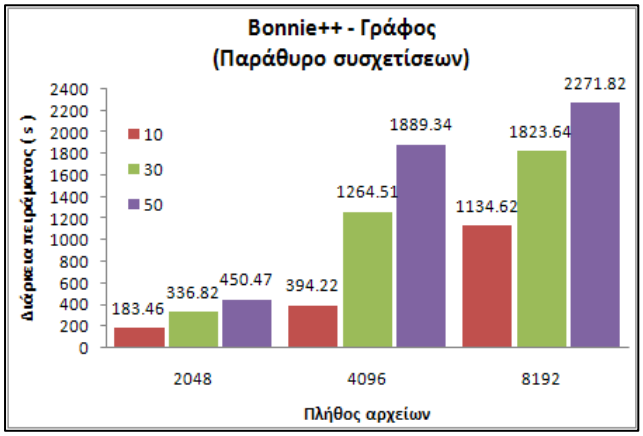
(γ)

Σχήμα 6.12 - Εξετάζουμε τη ρυθμαπόδοση του συστήματος (όσο μεγαλύτερη ρυθμαπόδοση τόσο καλύτερα), ***(α)*** *χρησιμοποιώντας τρεις διαφορετικές τιμές για το παράθυρο συσχετίσεων (10, 30 και 50 δευτερόλεπτα κάθε φορά) και σταθερό το παράθυρο ενημέρωσης ακμών (60 δευτερόλεπτα),* ***(β)*** *χρησιμοποιώντας τρεις διαφορετικές τιμές για το παράθυρο ενημέρωσης ακμών (40, 80 και 120 δευτερόλεπτα κάθε φορά) και σταθερό το παράθυρο συσχετίσεων (30 δευτερόλεπτα)* ***(γ)*** *χρησιμοποιώντας διαφορετικά μεγέθη του πίνακα κατακερματισμού (1024, 500.000 και 1.000.000).*

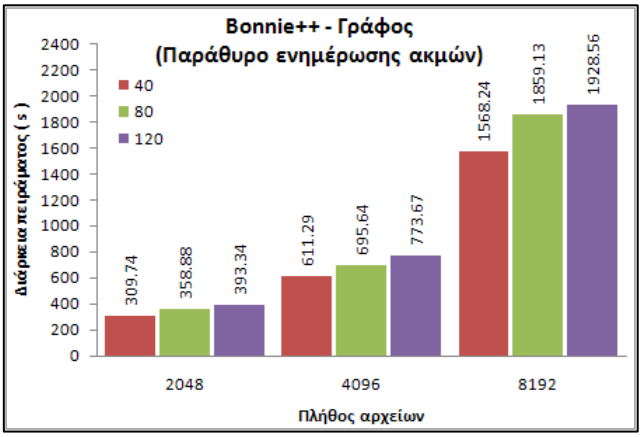
Αρχικά εκτελέσαμε τα πειράματα τρέχοντας το σύστημα που χρησιμοποιεί τον γράφο συσχετίσεων. Αυξάνοντας το παράθυρο συσχετίσεων και το πλήθος των αρχείων που διαχειρίζεται το Bonnie++, παρατηρούμε μείωση της ρυθμαπόδοσης του συστήματος (Σχήμα 6.12 - α). Αυτό συμβαίνει επειδή όσο αυξάνεται το παράθυρο συσχετίσεων τόσο περισσότερα αρχεία εισάγονται στο γράφο οπότε γίνεται πιο βαρύς. Η μείωση αυτή παρατηρείται και όταν αυξάνουμε το μέγεθος του παραθύρου ενημέρωσης ακμών (Σχήμα 6.12 - β), αφού οι διαγραφές κόμβων γίνονται σε μεγαλύτερο χρονικό διάστημα. Οπότε ο γράφος διαχειρίζεται μεγάλο όγκο δεδομένων για μεγαλύτερο χρόνο.

Στη συνέχεια, τρέξαμε τα ίδια πειράματα έχοντας αυτή τη φορά ενεργοποιημένο το σύστημα που χρησιμοποιεί τον πίνακα κατακερματισμού. Στην περίπτωση αυτή, παρατηρούμε πάλι ότι η ρυθμαπόδοση μειώνεται όσο αυξάνεται ο αριθμός των αρχείων που δημιουργεί το bonnie++. Ωστόσο, αυτή η μείωση είναι πολύ μικρότερη σε σχέση με πριν που χρησιμοποιήσαμε τον γράφο. Επίσης, παρατηρούμε ότι όσο μεγαλύτερο είναι το μέγεθος του πίνακα τόσο καλύτερη είναι η ρυθμαπόδοση (Σχήμα 6.12 - γ). Άρα, μια παρατήρηση που μπορούμε να κάνουμε είναι ότι ο πίνακας κατακερματισμού επιβαρύνει πολύ λιγότερο το σύστημα από το γράφο, και όσο το μέγεθός του αυξάνεται, η επιβάρυνση αυτή ελαττώνεται.

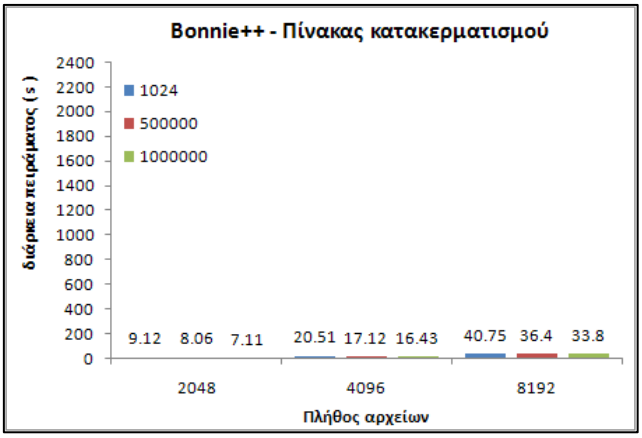
Ακολούθως, αξιολογούμε το χρόνο που διαρκεί το κάθε πείραμα. Το πείραμα ολοκληρώνεται πολύ πιο γρήγορα όταν χρησιμοποιείται ο πίνακας κατακερματισμού αντί του γράφου. Συγκεκριμένα, όσο αυξάνεται το μέγεθος του πίνακα κατακερματισμού, τόσο πιο γρήγορα ολοκληρώνεται το πείραμα (Σχήμα 6.13 - γ). Από την άλλη μεριά, ο γράφος, για κάθε πλήθος αρχείων που χρησιμοποιεί το Bonnie++, όσο μεγαλώνει το παράθυρο συσχετίσεων ή το παράθυρο ενημερώσεων ακμών, τόσο πιο αργός γίνεται, με αποτέλεσμα να επιβαρύνεται το σύστημα και να διαρκεί περισσότερο το πείραμα (Σχήμα 6.13 – α και β).



(α)



(β)

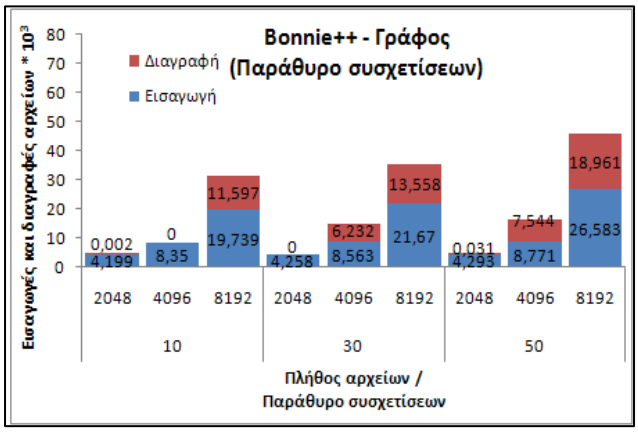


(γ)

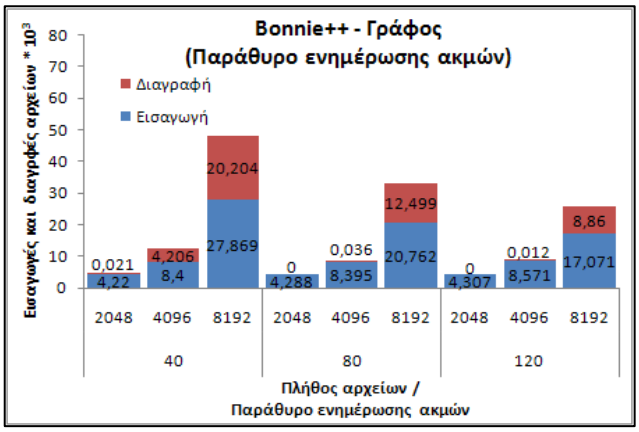
Σχήμα 6.13 - Εξετάζουμε το χρόνο εκτέλεσης του πειράματος (όσο μικρότερος χρόνος τόσο καλύτερα), ***(α)*** *χρησιμοποιώντας τρεις διαφορετικές τιμές για το παράθυρο συσχετίσεων (10, 30 και 50 δευτερόλεπτα κάθε φορά) και σταθερό το παράθυρο ενημέρωσης ακμών (60 δευτερόλεπτα),* ***(β)*** *χρησιμοποιώντας τρεις διαφορετικές τιμές για το παράθυρο ενημέρωσης ακμών (40, 80 και 120 δευτερόλεπτα κάθε φορά) και σταθερό το παράθυρο συσχετίσεων (30 δευτερόλεπτα)* ***(γ)*** *χρησιμοποιώντας διαφορετικά μεγέθη του πίνακα κατακερματισμού (1024, 500.000 και 1.000.000).*

Το πείραμα ολοκληρώνεται πολύ πιο γρήγορα όταν χρησιμοποιείται ο πίνακας κατακερματισμού αντί του γράφου. Συγκεκριμένα, όσο αυξάνεται το μέγεθος του πίνακα κατακερματισμού, τόσο πιο γρήγορα ολοκληρώνεται το πείραμα (Σχήμα 6.13 - γ). Από την άλλη μεριά, ο γράφος, για κάθε πλήθος αρχείων που χρησιμοποιεί το Bonnie++, όσο μεγαλώνει το παράθυρο συσχετίσεων ή το παράθυρο ενημερώσεων ακμών, τόσο πιο αργός γίνεται, με αποτέλεσμα να επιβαρύνεται το σύστημα και να διαρκεί περισσότερο το πείραμα (Σχήμα 6.13 – α και β).

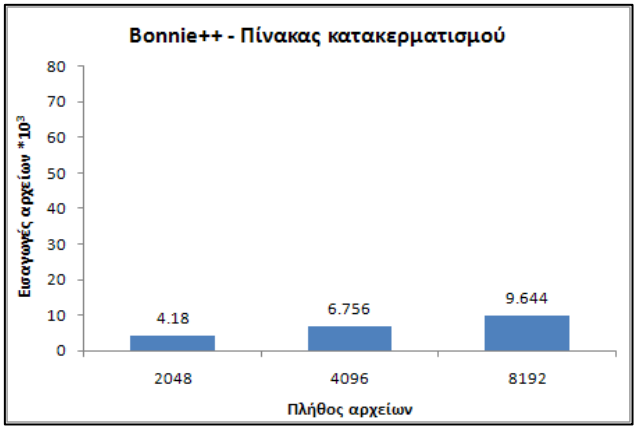
6.5.2 – Επιβάρυνση πίνακα κατακερματισμού και γράφου



(α)



(β)



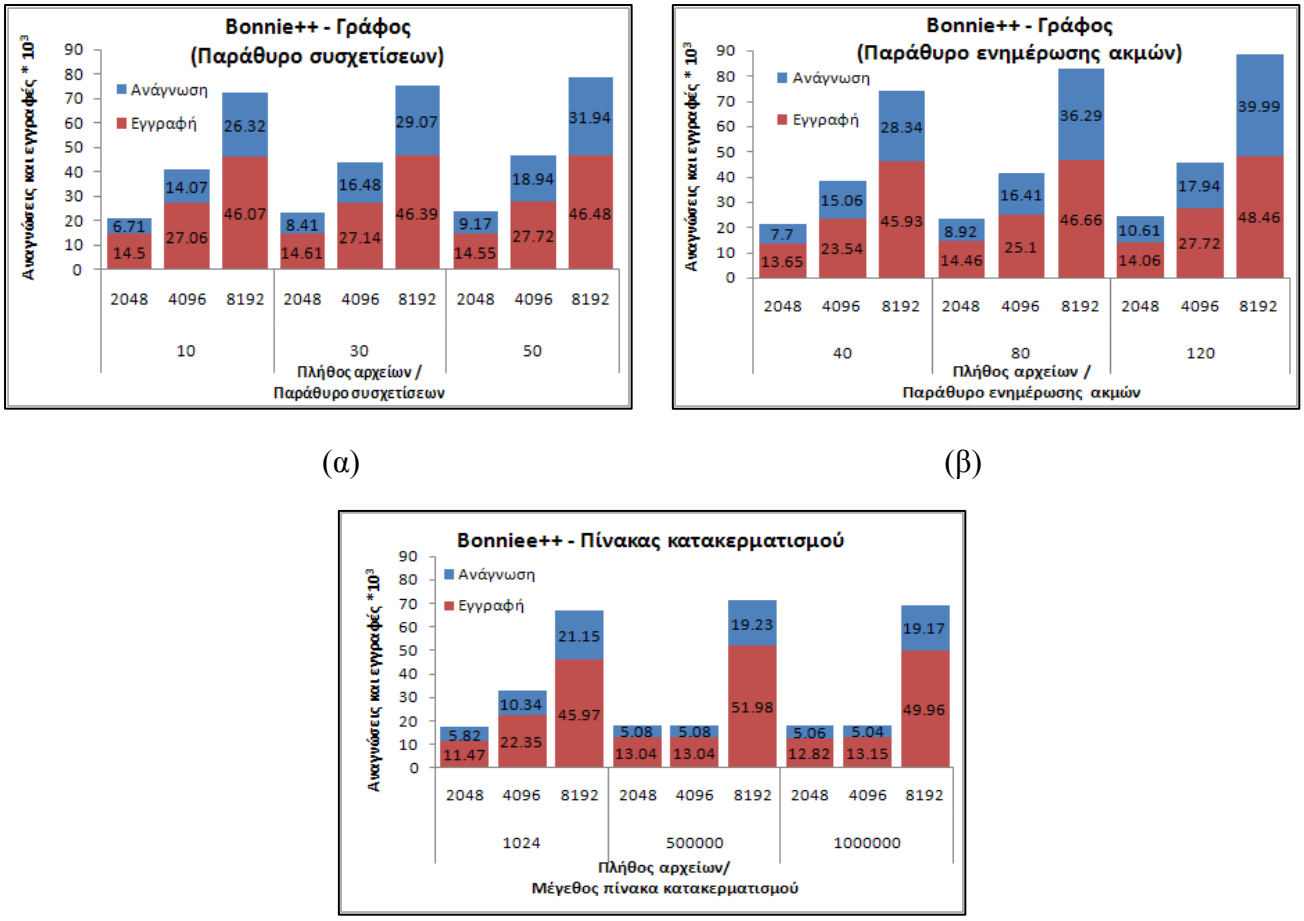
(γ)

Σχήμα 6.14 - Εξετάζουμε τις εισαγωγές και διαγραφές αρχείων, ***(α)*** *χρησιμοποιώντας τρεις διαφορετικές τιμές για το παράθυρο συσχετίσεων (10, 30 και 50 δευτερόλεπτα κάθε φορά) και σταθερό το παράθυρο ενημέρωσης ακμών (60 δευτερόλεπτα),* ***(β)*** *χρησιμοποιώντας τρεις διαφορετικές τιμές για το παράθυρο ενημέρωσης ακμών (40, 80 και 120 δευτερόλεπτα κάθε φορά) και σταθερό το παράθυρο συσχετίσεων (30 δευτερόλεπτα)* ***(γ)*** *χρησιμοποιώντας διαφορετικά μεγέθη του πίνακα κατακερματισμού (1024, 500.000 και 1.000.000).*

Στη συνέχεια, μετράμε τις διαγραφές κόμβων στο γράφο, τις εισαγωγές κόμβων, τις αναγνώσεις και εγγραφές αρχείων και στις δύο δομές, έτσι ώστε να δούμε με πιο τρόπο επιβαρύνονται οι δύο δομές που χρησιμοποιούμε. Στο γράφο ο αριθμός των εισαγωγών, για συγκεκριμένο πλήθος αρχείων του Bonnie++, αυξάνεται όσο μεγαλώνει το παράθυρο συσχετίσεων (Σχήμα 6.14 - α). Όταν το παράθυρο ενημέρωσης ακμών αυξάνεται, οι διαγραφές κόμβων μειώνονται αφού η ενημέρωση των βαρών γίνεται ανά μεγαλύτερα διαστήματα και οι εισαγωγές μειώνονται επειδή τα αρχεία παραμένουν μεγαλύτερο διάστημα στο γράφο (Σχήμα 6.14 - β). Οι

εισαγωγές αρχείων στο γράφο είναι περισσότερες από εκείνες στον πίνακα κατακερματισμού (Σχήμα 6.14 - γ).

Αυτό συμβαίνει γιατί καθώς το Bonnie++ δημιουργεί και γράφει τα αρχεία σειριακά, αυτά εισάγονται στο γράφο, αλλά μέχρι να ολοκληρωθεί η διαδικασία κάποια διαγράφονται λόγω της ενημέρωσης των ακμών. Έτσι, όταν στη συνέχεια τα ίδια αρχεία διαβάζονται με τυχαία σειρά από το Bonnie++ κάποια από αυτά θα πρέπει να εισαχθούν εκ νέου στο γράφο. Στον πίνακα κατακερματισμού όμως δεν τίθεται τέτοιο θέμα αφού όποιο αρχείο εισαχθεί δε διαγράφεται.

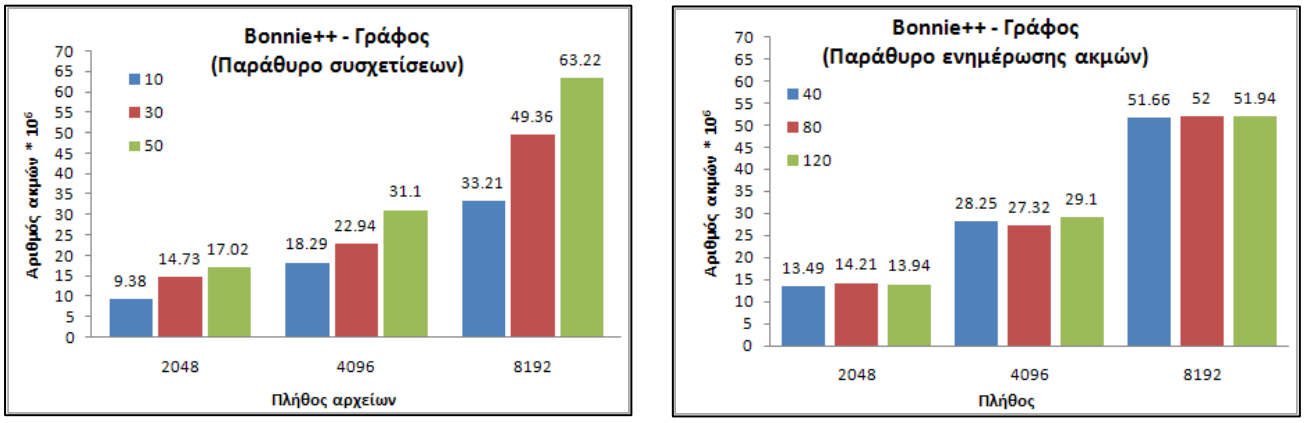


Σχήμα 6.15 - Εξετάζουμε τον αριθμό εγγραφών και αναγνώσεων στα αρχεία των δύο δομών, ***(α)*** *χρησιμοποιώντας τρεις διαφορετικές τιμές για το παράθυρο συσχετίσεων (10, 30 και 50 δευτερόλεπτα κάθε φορά) και σταθερό το παράθυρο ενημέρωσης ακμών (60 δευτερόλεπτα),* ***(β)*** *χρησιμοποιώντας τρεις διαφορετικές τιμές για το παράθυρο ενημέρωσης ακμών (40, 80 και 120 δευτερόλεπτα κάθε φορά) και σταθερό το παράθυρο συσχετίσεων (30 δευτερόλεπτα)* ***(γ)*** *χρησιμοποιώντας διαφορετικά μεγέθη του πίνακα κατακερματισμού (1024, 500.000 και 1.000.000).*

Οι εγγραφές και οι αναγνώσεις είναι περισσότερες στο γράφο απ' ότι στον πίνακα κατακερματισμού. Το γεγονός αυτό εξηγείται, καθώς τα πειράματα κατά τη χρήση του γράφου διήρκησαν περισσότερο από τα πειράματα που έγιναν όταν έτρεχε το σύστημα με τον πίνακα

κατακερματισμού, με αποτέλεσμα να καταγραφούν περισσότερες αιτήσεις προς τα αρχεία που χρησιμοποιεί το λειτουργικό σύστημα, ή άλλες εφαρμογές που έτρεχαν στο σύστημα (Σχήμα 6.15).

Για τη δομή του γράφου εξετάζουμε ακόμη πόσες ακμές δημιουργήθηκαν και ενημερώθηκαν. Στην πρώτη περίπτωση, που μεταβάλλουμε το παράθυρο συσχετίσεων, ο αριθμός των ακμών του γράφου αυξάνεται αφού αυξάνοντας το παράθυρο έχουμε περισσότερες συσχετίσεις μεταξύ αρχείων (Σχήμα 6.16 - α). Το γεγονός αυτό επιβαρύνει τον γράφο και έχει ως συνέπεια την επιβάρυνση του συστήματος, όπως φαίνεται και στο Σχήμα 6.12 - α. Όταν κρατάμε σταθερό το παράθυρο συσχετίσεων και μεγαλώνουμε την τιμή του παραθύρου ενημέρωσης ακμών (Σχήμα 6.16 - β), ο αριθμός των συσχετίσεων είναι σχεδόν ο ίδιος καθώς μεταβάλλεται το παράθυρο ενημέρωσης ακμών.



(α) (β)

*Σχήμα 6.16 - Εξετάζουμε το πλήθος των ακμών που δημιουργήθηκαν κατά τη διάρκεια εκτέλεσης του πειράματος **(α)** χρησιμοποιώντας τρεις διαφορετικές τιμές για το παράθυρο συσχετίσεων (10, 30 και 50 δευτερόλεπτα κάθε φορά) και σταθερό το παράθυρο ενημέρωσης ακμών (60 δευτερόλεπτα), **(β)** χρησιμοποιώντας τρεις διαφορετικές τιμές για το παράθυρο ενημέρωσης ακμών (40, 80 και 120 δευτερόλεπτα κάθε φορά) και σταθερό το παράθυρο συσχετίσεων (30 δευτερόλεπτα).*

6.6 – Περίληψη

Για την αξιολόγηση των διαφορετικών σχεδιασμών του συστήματος που υλοποιήσαμε, κάναμε μετρήσεις που εξετάζουν την επιβάρυνση των δομών του και την επιβάρυνση που αντιλαμβάνεται ο χρήστης καθώς εργάζεται στο σύστημα. Εξετάζουμε την απόδοση του συστήματος που υλοποιήσαμε κατά τη χρήση του σε προσωπικό υπολογιστή και σε διακομιστή. Για την αξιολόγηση της χρήσης σε προσωπικό υπολογιστή χρησιμοποιήσαμε το εργαλείο PCMAPK05 το οποίο προσομοιώνει καθημερινές λειτουργίες του χρήστη και μετρά το ρυθμό μεταφοράς

δεδομένων. Επίσης δημιουργήσαμε ένα εργαλείο microbenchmark. Το microbenchmark σε αντίθεση με το PCMAPK05 δημιουργεί προεπιλεγμένο πλήθος αρχείων, δίνοντας μια εκτενέστερη εικόνα για την απόδοση του συστήματος σε συνθήκες υψηλού φόρτου. Ακόμη, μετρήσαμε τη διάρκεια μεταγλώττισης του πυρήνα 2.4.37 του Linux για τους διαφορετικούς σχεδιασμούς τους συστήματος. Για την αξιολόγηση της χρήσης σε διακομιστή χρησιμοποιήσαμε το εργαλείο bonnie++. Το bonnie++ μετρά τη ρυθμαπόδοση του συστήματος αρχείων κατά την εκτέλεση διάφορων εργασιών σε αρχεία, όπως δημιουργία, διαγραφή, εγγραφή και ανάγνωση. Μελετήσαμε τα αποτελέσματα των πειραμάτων και διαπιστώσαμε ότι όταν χρησιμοποιείται ο πίνακας κατακερματισμού ως δομή αποθήκευσης των στατιστικών χρήσης των αρχείων έχουμε καλύτερα αποτελέσματα σε σχέση με το γράφο. Πετυχαίνει καλύτερο ρυθμό μεταφοράς δεδομένων, καλύτερη ρυθμαπόδοση στο σύστημα αρχείων και διαχειρίζεται αποτελεσματικότερα μεγάλο πλήθος αρχείων.

Κεφάλαιο 7

Συμπεράσματα

Οι αυξημένες απαιτήσεις των χρηστών για μια γρήγορη και επιτυχημένη αναζήτηση των αρχείων σε προσωπικούς υπολογιστές και διακομιστές αποθήκευσης, δείχνουν ότι τα ήδη υπάρχοντα συστήματα αναζήτησης δεν καλύπτουν σε ικανοποιητικό βαθμό τις ανάγκες τους. Έχοντας λοιπόν ως κίνητρο την ανάγκη των χρηστών για ένα πιο αποτελεσματικό εργαλείο αναζήτησης σε συστήματα αρχείων, προσεγγίσαμε το πρόβλημα με έναν διαφορετικό τρόπο απ' ότι είχε γίνει μέχρι τώρα.

Κατά την άποψή μας η καταγραφή των στατιστικών της χρήσης των αρχείων σε έναν πίνακα κατακερματισμού βελτιώνει την αναζήτησή τους από το χρήστη, αφού ο πίνακας κατακερματισμού επιτρέπει την ταχύτατη ενημέρωση και ανάκτησή τους. Επίσης, τα αρχεία με τις περισσότερες εγγραφές και αναγνώσεις, δηλαδή εκείνα που χρησιμοποιούνται συχνότερα από το χρήστη, θα μπορούν να έρχονται στις πρώτες θέσεις των αποτελεσμάτων της αναζήτησης.

Έτσι, κατά την παρούσα εργασία αναπτύξαμε ένα σύστημα το οποίο καταγράφει τα στατιστικά της χρήσης των αρχείων σε έναν πίνακα κατακερματισμού. Για να διαπιστώσουμε αν ο πίνακας κατακερματισμού είναι καταλληλότερη δομή από το γράφο συσχετίσεων που χρησιμοποιείται σε άλλα συστήματα αναζήτησης, υλοποιήσαμε ένα δεύτερο σύστημα που καταγράφει συνδέσεις μεταξύ των αρχείων που χρησιμοποιεί ο χρήστης σε έναν γράφο συσχετίσεων.

Στη συνέχεια, εκτελέσαμε πειράματα χρησιμοποιώντας τις δύο διαφορετικές δομές αποθήκευσης, διαπιστώνοντας ότι το σύστημα που χρησιμοποιεί τον πίνακα κατακερματισμού συμπεριφέρεται καλύτερα από το σύστημα που χρησιμοποιεί το γράφο και επιβαρύνει ελάχιστα το σύστημα. Αντίθετα, ο γράφος συσχετίσεων σε μεγάλο φόρτο επιβαρύνεται πολύ γρήγορα με αποτέλεσμα να επιβαρύνει και το ίδιο το σύστημα.

Κλείνοντας, θα θέλαμε να σημειώσουμε ότι ο πίνακας κατακερματισμού που χρησιμοποιήσαμε είναι στατικός και χρησιμοποιεί απλές συνδεδεμένες λίστες για την αντιμετώπιση των συγκρούσεων. Από τα αποτελέσματα των πειραμάτων διαπιστώσαμε ότι όσο αυξάνεται το μέγεθος του πίνακα τόσο αυξάνεται και η απόδοση του συστήματος. Θεωρούμε, λοιπόν, σημαντικό να τονίσουμε ότι το σύστημά μας θα μπορούσε στο μέλλον να επεκταθεί και να χρησιμοποιηθεί ως δομή ένας επεκτάσιμος πίνακας κατακερματισμού.

Αναφορές

[1] Mark E. Russinovich and David E. Sollomon. Microsoft Windows Internals. Microsoft Press, U.S., 2005.

[2] Walter Oney. Programming the Microsoft Windows Driver Model. Microsoft Press, U.S., 2003

[3] Craig A. N. Soules, and Gregory N. Ganger. Connections: Using Context to Enhance File Search. SOSP `05, Brigton, UK, 2005.

[4] Sam Shah, Craig A. N. Soules, Gregory R. Ganger and Brian D. Noble. Using Provenance to Aid in Personal File Search. Usenix `07 Annual Technical Conference, Santa Clara, CA, 2007.

[5] Matthew Richrdson, Amit Prakash, Eric Brill. Beyond PageRank: Machine Learning for Static Ranking. WWW 2006, Edinburgh, Scotland, 2006.

[6] Michael Mesnier, Eno Thereska, Gregory R. Ganger, Daniel Ellard and Margo Seltzer. File classification in self-* storage systems. ICAC-04, New York, NY, 2004.

[7] Paula A. Farina, A Comparison of Two Desktop Search Engines: Google Desktop Search (Beta) vs. Windows XP Search, 21st Computer Science Seminar SA3-T3-1

[8] Apple spotlight search. http://developer.apple.com/macosx/tiger/spotlight.html.

[9] Yahoo! desktop search. http://info.yahoo.com/privacy/in/yahoo/desktopsearch/.

[10] Windows search, http://www.microsoft.com/windows/products/winfamily/desktopsearch /default.mspx .

[11] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. Journal of the ACM, 46(5): 604-632. ACM, September 1999

[12] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. Computer Networks and ISDN Systems, 30(1-7): 107-117, 1998.

[13] Google Desktop, http://www.google.com.

[14] The Lemur Toolkit, http://lemurproject.com.

[15] Terrier Information Retrieval Platform, http://ir.dcs.gla.ac.uk/terrier.