ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

Χρήση της Απομακρυσμένης Μνήμης ως Αποθηκευτικού Χώρου του Συστήματος Διαχείρισης Ιδεατής Μνήμης

Γιώργος Δραμιτινός

Μεταπτυχιακή Εργασία

Ηράκλειο, Ιούνιος 1996

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

Χρήση της Απομακρυσμένης Μνήμης ως Αποθηκευτικού Χώρου του Συστήματος Διαχείρισης Ιδεατής Μνήμης

Εργασία που υποβλήθηκε από τον
Γιώργο Δραμιτινό
ως μερική εκπλήρωση των απαιτήσεων
για την απόκτηση
ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΔΙΠΛΩΜΑΤΟΣ ΕΙΔΙΚΕΥΣΗΣ

Συγγραφέας:

Γιώργος Δραμιτινός
Τμήμα Επιστήμης Υπολογιστών

Εισηγητική Επιτροπή:

Ευάγγελος Μαρκάτος, Επίκουρος Καθηγητής, Επόπτης

Χρήστος Νικολάου, Αναπληρωτής Καθηγητής, Μέλος

Απόστολος Τραγανίτης, Αναπληρωτής Καθηγητής, Μέλος

Δεκτή:

Πάνος Κωνσταντόπουλος, Αναπληρωτής Καθηγητής
Πρόεδρος Επιτροπής Μεταπτυχιακών Σπουδών

Ηράκλειο, Ιούνιος 1996

Χρήση της Απομακρυσμένης Μνήμης ως Αποθηκευτικού Χώρου του Συστήματος Διαχείρισης Ιδεατής Μνήμης

Γιώργος Δραμιτινός

Μεταπτυχιακή Εργασία

Τμήμα Επιστήμης Υπολογιστών
Πανεπιστήμιο Κρήτης

Περίληψη

Παραδοσιακά τα λειτουργικά συστήματα χρησιμοποιούν μαγνητικούς δίσκους για την αποθήκευση σελίδων από το σύστημα διαχείρισης ιδεατής μνήμης παρά το γεγονός ότι το κόστος μιας προσπέλασης στο δίσκο, μετρημένο σε κύκλους επεξεργαστή, συνεχίζει να αυξάνεται.

Στην εργασία αυτή μελετάται η χρήση της μνήμης των μηχανών ενός τοπικού δικτύου ως αποθηκευτικού χώρου του συστήματος διαχείρισης ιδεατής μνήμης. Περιγράφουμε τη σχεδίαση, υλοποίηση και αξιολόγηση ενός σελιδοδιαχειριστή που χρησιμοποιεί την κύρια μνήμη των σταθμών εργασίας ενός τοπικού δικτύου σαν μια συσκευή αποθήκευσης σελίδων γρηγορότερη από το δίσκο και που παρέχει τη δυνατότητα αδιάλειπτης λειτουργίας ακόμη και στην περίπτωση βλάβης ενός μηχανήματος του δικτύου. Ο σελιδοδιαχειριστής μας έχει υλοποιηθεί σαν οδηγός συσκευής, ο οποίος συνδέεται με το λειτουργικό σύστημα DEC OSF/1 χωρίς να απαιτεί καμιά αλλαγή στον κώδικα του πυρήνα και ουσιαστικά μετατρέπει το σύνολο της κύριας μνήμης του τοπικού δικτύου σε μια μεγάλη κρυφή μνήμη για τα τμήματα των δίσκων που χρησιμοποιούνται για αποθήκευση σελίδων. Χρησιμοποιώντας αρκετές εφαρμογές μετράμε την απόδοση της χρήσης του συστήματός μας πάνω από ένα δίκτυο Ethernet και βρίσκουμε ότι είναι καλύτερη από αυτή του τοπικού δίσκου. Αξιολογούμε την απόδοση διαφόρων τεχνικών αξιοπιστίας και αποδεικνύουμε τη βιωσιμότητά τους ακόμη και πάνω από δίκτυα χαμηλού ρυθμού μεταγωγής δεδομένων, όπως το Ethernet. Προσομοιώνουμε τη λειτουργία του συστήματός μας πάνω από δίκτυα υψηλού ρυθμού μεταγωγής δεδομένων και αξιολογούμε την απόδοσή του.

Συμπεραίνουμε ότι τα οφέλη από τη χρήση της μνήμης ολόκληρου του δικτύου για την αποθήκευση σελίδων είναι αξιόλογα σήμερα και πρόκειται να γίνουν ακόμη πιο σημαντικά στο μέλλον.

Επόπτης: Ευάγγελος Μαρκάτος
Επίκουρος Καθηγητής
Τμήμα Επιστήμης Υπολογιστών
Πανεπιστήμιο Κρήτης.

Reliable Paging to Remote Main Memory in a Workstation Cluster

George Dramitinos

Master of Science Thesis

Department of Computer Science
University of Crete

Abstract

Traditional operating systems use magnetic disks as paging devices, even though the cost of a disk transfer measured in processor cycles continues to increase.

In this paper we explore the use of remote main memory for paging. We describe the design, implementation and evaluation of a pager that uses main memory of remote workstations in a workstation cluster as a faster−than−disk paging device and provides reliability in case of single workstation failures. Our pager has been implemented as a block device driver linked to the DEC OSF/1 operating system, without any modifications to the kernel code and effectively turns the cluster's main memory into a global cache for its swap partitions. Using several test applications we measure the performance of remote memory paging over an Ethernet interconnection network and find it to be faster than traditional disk paging. We evaluate the performance of various reliability policies and prove their feasibility even over low bandwidth networks, like Ethernet. We use our system over simulated high bandwidth networks and evaluate its performance.

We conclude that the benefits of reliable remote memory paging in workstation clusters are significant today and will probably increase in the near future.

Advisor: Evangelos Markatos
Assistant Professor
Computer Science Department
University of Crete.

Ευχαριστίες

Ευχαριστώ θερμά τον επόπτη καθηγητή μου κ. Ευάγγελο Μαρκάτο που μου έδωσε την δυνατότητα να ασχοληθώ με το θέμα της εργασίας αυτής και με καθοδήγησε σε όλη τη διάρκειά της. Ιδιαίτερα τον ευχαριστώ για τις αναρίθμητες ώρες συζητήσεων σχετικά με την εργασία και γιατί πάντα ήταν έτοιμος να προσφέρει οποιαδήποτε βοήθεια χρειαζόμουν.

Ο κ. Χρήστος Νικολάου και ο κ. Απόστολος Τραγανίτης, μέλη της εξεταστικής επιτροπής, έκαναν εκτενή και ουσιώδη σχόλια σχετικά με την αρχική έκδοση του κειμένου της εργασίας. Οι υποδείξεις τους συνέβαλαν ουσιαστικά στη βελτίωση της παρουσίασης. Ευχαριστώ επίσης τον κ. Σαράντο Καπιδάκη, τον κ. Χρήστο Νικολάου, τον κ. Μανόλη Κατεβαίνη και την κ. Κατερίνα Χρονάκη για τις πολύτιμες υποδείξεις τους σε παλαιότερες εκδόσεις αυτού του κειμένου.

Ενα μεγάλο ευχαριστώ αξίζει στον Κοσμά Παπαχρήστο που με την εμπειρία του σε λειτουργικά συστήματα είχε πάντα ενδιαφέρουσες παρατηρήσεις σχετικά με την υλοποίηση του σελιδοδιαχειριστή. Επίσης στο Σωτήρη Ιωαννίδη για τις εκτενείς συζητήσεις για τη χρήση της συνολικής μνήμης ενός τοπικού δικτύου σαν κρυφή μνήμη για τους δίσκους.

Ο Αλέξανδρος Λαμπρινίδης και ο Απόστολος Ζάρρας έκαναν χρήσιμες παρατηρήσεις σε πρώιμα στάδια της εργασίας αυτής. Επίσης θα ήθελα να ευχαριστήσω τα μέλη της ομάδας των ΠΛΕΙΑΔΩΝ και της ομάδας της Αρχιτεκτονικής Υπολογιστών και Συστημάτων VLSI για την υποστήριξή τους σε όλη τη διάρκεια της εργασίας μου. Ιδιαίτερα ευχαριστώ τους : Πηνελόπη Κωνσταντά–Φανουράκη, Τίτο Σαρειδάκη, Μανόλη Μαραζάκη, Μαρία Καραβασίλη και Γιώργο Γεωργιαννάκη.

Ακόμα οφείλω να ευχαριστήσω τους γονείς μου, Πέτρο και Βούλα, για τη συμπαράστασή τους και την αμέριστη υποστήριξη που μου έδωσαν καθόλη τη διάρκεια των σπουδών μου.

Επίσης θα ήθελα να ευχαριστήσω το Τμήμα Επιστήμης Υπολογιστών του Πανεπιστημίου Κρήτης και το Ινστιτούτο Πληροφορικής του Ιδρύματος Τεχνολογίας και Ερευνας για την υλικοτεχνική και οικονομική υποστήριξη που μου παρείχαν κατά τη διάρκεια των σπουδών μου.

Περιεχόμενα

Περίληψη **i**

Abstract **iii**

Ευχαριστίες **v**

Περιεχόμενα **vii**

Κατάλογος Πινάκων **ix**

Κατάλογος Σχημάτων **xi**

1 Εισαγωγή **1**
1.1 Το Πρόβλημα . . . 1
1.2 Η Αιτία του Προβλήματος . . . 1
1.3 Η Προτεινόμενη Λύση . . . 6
1.4 Η συνεισφορά αυτής της εργασίας . . . 7

2 Σχεδίαση **9**
2.1 Στόχοι Σχεδίασης . . . 9
2.2 Περιβάλλον Ανάπτυξης . . . 11
2.3 Ο Σελιδοδιαχειριστής . . . 11
2.4 Ο Εξυπηρετητής Μνήμης . . . 12

3 Υλοποίηση **13**
3.1 Η Υλοποίηση του Σελιδοδιαχειριστή . . . 13
3.1.1 Γενική Περιγραφή . . . 13
3.1.2 Η Διασύνδεση με το Σελιδοδιαχειριστή του DEC OSF/1 . . . 13
3.1.3 Η Διασύνδεση με το Δίκτυο . . . 16
3.1.4 Η Διασύνδεση με το Δίσκο . . . 19
3.2 Η Υλοποίηση του Εξυπηρετητή Μνήμης . . . 19
3.3 Θέματα Πολιτικής . . . 20
3.3.1 Επιλογή Εξυπηρετητών . . . 20
3.3.2 Συμπίεση . . . 22
3.4 Πειραματικά Αποτελέσματα . . . 22
3.4.1 Απόδοση πάνω από το Ethernet . . . 22
3.4.2 Απόδοση σε Σχέση με το Μέγεθος Εισόδου . . . 23
3.4.3 Απόδοση πάνω από Δίκτυα Μεγαλύτερου Ρυθμού Μεταγωγής Δεδομένων . . . 24
3.4.4 Η Καθυστέρηση ανά Σελίδα . . . 26
3.4.5 Η Χρήση Φορτωμένων Εξυπηρετητών . . . 27
3.4.6 Η Απόδοση πάνω από Φορτωμένο Ethernet . . . 27

4 Αξιοπιστία **29**
4.1 Εισαγωγή . . . 29
4.2 Στόχοι των Πολιτικών Αξιοπιστίας . . . 30
4.3 Συγκριτική Μελέτη Πολιτικών Αξιοπιστίας . . . 30
4.3.1 Η Τεχνική των Πολλαπλών Αντιγράφων . . . 30
4.3.2 Η Τεχνική της Ισοτιμίας . . . 31
4.3.3 Η Τεχνική της Μεταβλητής Ισοτιμίας . . . 31
4.3.4 Πειραματικά Αποτελέσματα . . . 34
4.4 Η Χρήση του Τοπικού Δίσκου για την Αύξηση της Αξιοπιστίας . . . 35
4.5 Προσαρμοζόμενη Σελιδοδιαχείριση . . . 36
4.5.1 Πειραματικά Αποτελέσματα . . . 37

5 Σχετική Εργασία **41**
5.1 Συστήματα που Βασίζονται σε Ειδικό Υλικό . . . 41
5.1.1 Επεκτάσεις Μνήμης με τη Χρήση του Διαύλου Εισόδου/Εξόδου . . . 41
5.1.2 eNVy . . . 41
5.1.3 Σύγκριση . . . 43
5.2 Συστήματα που Χρησιμοποιούν Εξωτερικούς Εξυπηρετητές Μνήμης . . . 44
5.2.1 Το Σύστημα του Πανεπιστημίου της Washington . . . 44
5.2.2 Η Χρήση Απομακρυσμένης Μνήμης σε Φορητούς Υπολογιστές . . . 44
5.2.3 Σύγκριση . . . 45
5.3 Η Χρήση της Απομακρυσμένης Μνήμης Τροποποιώντας την Εφαρμογή . . . 45
5.4 Η Χρήση Απομακρυσμένης Μνήμης σε Μηχανήματα Χωρίς Τοπικό Δίσκο . . . 46
5.5 NOW . . . 47
5.6 Υλοποίηση Κοινής Μνήμης σε ένα Δίκτυο Υπολογιστών . . . 47
5.7 Κατανεμημένα Συστήματα Αρχείων . . . 48
5.7.1 xFS . . . 48
5.7.2 PACA . . . 49
5.7.3 Σύγκριση . . . 49
5.8 Χρήση της απομακρυσμένης μνήμης σε πολυεπεξεργαστές . . . 49
5.9 RAIDs . . . 50
5.10 Συστήματα Αρχείων που Βασίζονται σε Ημερολόγιο . . . 50
5.11 Zebra . . . 52
5.12 Συστήματα Κατανεμημένης Μοιραζόμενης Μνήμης . . . 52

6 Συμπεράσματα **53**

A Ορολογία **55**

Βιβλιογραφία **57**

Κατάλογος Πινάκων

1.1 **Χαρακτηριστικά των επιπέδων της ιεραρχίας μνήμης.** 3

3.1 **Η διασύνδεση μιας συσκευής μπλοκ με το λειτουργικό σύστημα.** Το xxx είναι ένα όνομα που εξαρτάται από τη συσκευή. Στην περίπτωσή μας επιλέχθηκε το rp (από το Remote Pager). 15

3.2 **Η διασύνδεση μιας συσκευής τυχαίας προσπέλασης σαν συσκευή χαρακτήρων με το λειτουργικό σύστημα.** 19

Κατάλογος Σχημάτων

1.1 **Χρόνος εκτέλεσης της εφαρμογής QSORT σαν συνάρτηση του μεγέθους των δεδομένων εισόδου της σε μηχανήματα με 32 MB και 64 MB φυσικής μνήμης.** Μόλις το μέγεθος των δεδομένων εισόδου υπερβεί τα 18 MB αρχίζει η αποθήκευση σελίδων στο δίσκο στο μηχάνημα με τα 32 MB μνήμης με αποτέλεσμα την τεράστια αύξηση στο χρόνο εκτέλεσης. 2

1.2 **Χρόνος εκτέλεσης της εφαρμογής FFT σαν συνάρτηση του μεγέθους των δεδομένων εισόδου της σε μηχανήματα με 32 MB και 64 MB φυσικής μνήμης.** Μόλις το μέγεθος των δεδομένων εισόδου υπερβεί τα 18 MB αρχίζει η αποθήκευση σελίδων στο δίσκο στο μηχάνημα με τα 32 MB μνήμης με αποτέλεσμα την τεράστια αύξηση στο χρόνο εκτέλεσης. 2

1.3 **Η δομή της ιεραρχίας μνήμης.** 4

1.4 **Η δομή του δίσκου.** 5

1.5 **Η ελεύθερη μνήμη ενός τοπικού δικτύου 16 μηχανών με ολική μνήμη 800 MB κατά τη διάρκεια μιας εβδομάδας.** 7

2.1 **Σε κάθε χρονική στιγμή, τα μηχανήματα με μεγάλο φόρτο μνήμης ενεργούν σαν πελάτες ενώ τα μηχανήματα με αρκετή ελεύθερη μνήμη σαν εξυπηρετητές.** 10

3.1 **Η σύνδεση του σελιδοδιαχειριστή και η αλληλεπίδρασή του με το λειτουργικό σύστημα.** . . 14

3.2 **Τα στρώματα συναρτήσεων του κώδικα δικτύωσης.** 17

3.3 **Οι διαδοχικές αντιγραφές δεδομένων χρησιμοποιώντας το TCP/IP.** 18

3.4 **Η επιβάρυνση του λογισμικού κατά τη μεταφορά δεδομένων μεγέθους 8KB με τη χρήση του TCP/IP.** 18

3.5 **Η απόδοση εφαρμογών χρησιμοποιώντας είτε το δίσκο είτε την απομακρυσμένη μνήμη.** Για όλες τις εφαρμογές, η χρήση της απομακρυσμένης μνήμης δίνει πολύ καλύτερα αποτελέσματα από τη χρήση του δίσκου. Ολες οι εφαρμογές εκτελέστηκαν σε DEC Alpha 3000 model 300. Τα μεγέθη εισόδου ήταν 3000 στοιχεία για την `QSORT`, ένας πίνακας 1700 × 1700 για το `GAUSS`, ένας πίνακας 2100 × 2100 για τον `MVEC`, ένας πίνακας με 700 K στοιχεία για τον `FFT` και μια εικόνα 12 MB για το `FILTER`. 23

3.6 **Απόδοση της εφαρμογής `FFT` χρησιμοποιώντας είτε την απομακρυσμένη μνήμη είτε το δίσκο, σαν συνάρτηση του μεγέθους των δεδομένων εισόδου.** 24

3.7 **Απόδοση του `FFT` για διάφορες επιλογές τοποθέτησης σελίδων.** Παρουσιάζονται οι επιδόσεις της εφαρμογής πάνω από τον τοπικό δίσκο και το Ethernet. Επίσης παρουσιάζονται οι εκτιμούμενες επιδόσεις της εφαρμογής σε ένα δίκτυο με 10 φορές μεγαλύτερο ρυθμό μεταγωγής δεδομένων από το Ethernet και σε ένα σύστημα με αρκετή φυσική μνήμη ώστε να χωρά όλα τα δεδομένα της εφαρμογής. 25

4.1 **Η λειτουργία της μεθόδου της μεταβλητής ισοτιμίας.** 33

4.2 **Η απόδοση των εφαρμογών με τη χρήση διαφόρων τεχνικών αξιοπιστίας.** Για όλες τις εφαρμογές, η τεχνική της *μεταβλητής ισοτιμίας* δίνει πολύ καλύτερες επιδόσεις από το δίσκο προσθέτοντας μια πολύ μικρή επιβάρυνση σε σχέση με τη χρήση της απομακρυσμένης μνήμης χωρίς αξιοπιστία. 35

4.3 **Απόδοση της τεχνικής της *μεταβλητής ισοτιμίας* και της τεχνικής της *διεγγραφής* πάνω από Ethernet για διάφορες εφαρμογές.** Τα μεγέθη εισόδου είναι 3000 στοιχεία για την `QSORT`, ένας πίνακας 3000 στοιχείων για το `GAUSS`, ένας πίνακας 1700×1700 για τον `MVEC` και ένας πίνακας 700 K σημείων για τον `FFT`. 36

4.4 **Απόδοση του** `FFT` **χρησιμοποιώντας διάφορες τεχνικές αξιοπιστίας.** Το μέγεθος εισόδου είναι μια ακολουθία με 650K στοιχεία. . . . 38
4.5 **Απόδοση του** `GAUSS` **χρησιμοποιώντας διάφορες τεχνικές αξιοπιστίας.** Το μέγεθος εισόδου είναι ένας πίνακας με 1700 στοιχεία. . . . 38
4.6 **Τα οφέλη της ικανότητας προσαρμογής για την εφαρμογή FFT.** Το μέγεθος της εισόδου είναι 20 ακολουθίες των 650K στοιχείων. . . . 39
4.7 **Τα οφέλη της ικανότητας προσαρμογής για την εφαρμογή GAUSS.** Το μέγεθος της εισόδου είναι 20 πίνακες των 1600 στοιχείων. . . . 40

5.1 **Η επεκταμένη μνήμη στο δίαυλο VME.** . . . 42
5.2 **Η χρήση της επεκταμένης μνήμης αντί του δίσκου για την αποθήκευση σελίδων που δε χωρούν στην κύρια μνήμη.** . . . 42
5.3 **Η χρήση της επεκταμένης μνήμης σαν πιο αργή κύρια μνήμη.** . . . 43
5.4 **Διάγραμμα του συστήματος eNVy.** . . . 44
5.5 **Το σύστημα του πανεπιστημίου της Washington.** . . . 45
5.6 **Η τροποποίηση στο σύστημα διαχείρισης μνήμης του DEC OSF/1.** . . . 48
5.7 **Η ιεραρχία μνήμης ενός πολυεπεξεργαστή.** . . . 51

Στους γονείς μου,
Πέτρο και Βούλα

Κεφάλαιο 1

Εισαγωγή

1.1 Το Πρόβλημα

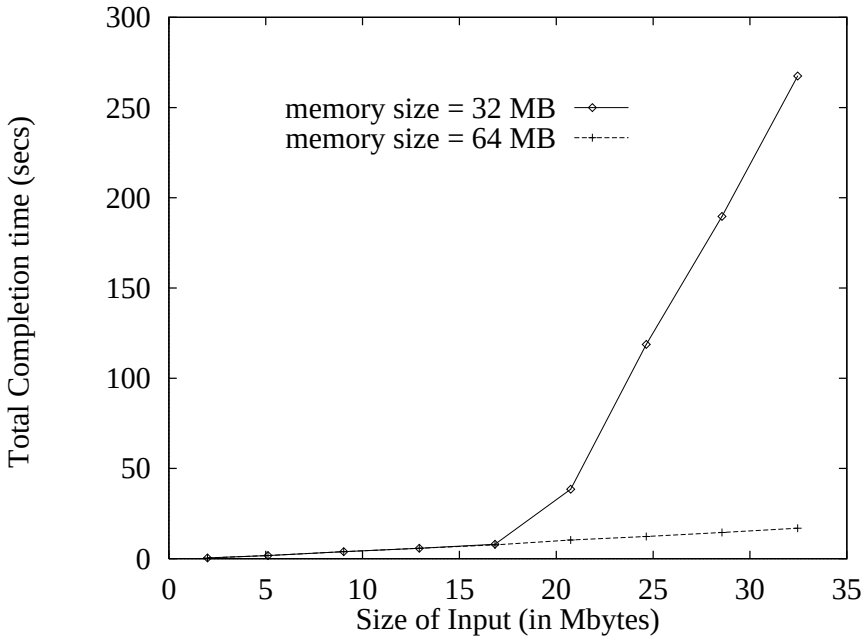
Η υπολογιστική ισχύς που παρέχουν οι μικροεπεξεργαστές αυξάνεται συνεχώς με μέσο ρυθμό μεγαλύτερο από 35% το χρόνο [HP90] κατά τα τελευταία 15 χρόνια. Επίσης το ποσό της μνήμης που μπορεί να προσπελάσει ένας μικροεπεξεργαστής έχει αυξηθεί δραματικά, φτάνοντας σήμερα σε χώρους διευθύνσεων των 64 bits [Sit92, SPA92]. Αυτοί οι δυο παράγοντες έχουν σαν αποτέλεσμα την ανάπτυξη εφαρμογών που προσπελαύνουν τεράστια ποσά μνήμης είτε για να λύσουν προβλήματα που έχουν αρκετά μεγάλο μέγεθος δεδομένων εισόδου (επιστημονικές εφαρμογές, συστήματα πολυμέσων) είτε για να αυξήσουν την απόδοσή τους (αρχεία και βάσεις δεδομένων απεικονισμένα στη μνήμη, γραφικά συστήματα).

Παρόλα αυτά, το μέγεθος της φυσικής μνήμης ενός σταθμού εργασίας δεν αυξάνεται με τους ίδιους ρυθμούς. Αυτό σημαίνει ότι όλο και πιο συχνά ο χρήστης αναγκάζεται να εκτελεί εφαρμογές που το σύνολο εργασίας τους δε χωρά στη διαθέσιμη κύρια μνήμη. Το φαινόμενο αυτό γίνεται ακόμη εντονότερο αν συνυπολογιστεί ο πολυπρογραμματισμός, ο οποίος μειώνει ακόμη περισσότερο το ποσό της διαθέσιμης φυσικής μνήμης ανά εφαρμογή. Το αποτέλεσμα του φαινομένου αυτού φαίνεται καθαρά στα σχήματα 1.1 και 1.2.

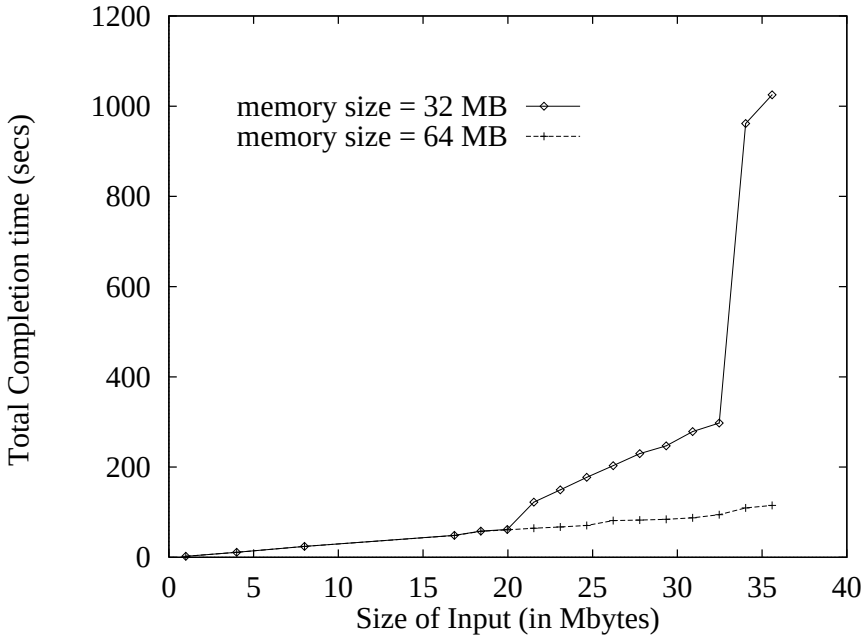
Ο χρόνος εκτέλεσης των εφαρμογών `QSORT` (ταξινόμηση πίνακα) και `FFT` (υλοποίηση γρήγορου μετασχηματισμού Fourier) αυξάνει απότομα από ένα μέγεθος δεδομένων εισόδου και μετά στο μηχάνημα που διαθέτει φυσική μνήμη 32 MB σε σχέση με το μηχάνημα που διαθέτει φυσική μνήμη 64 MB, αρκετή δηλαδή για να χωρέσει το σύνολο των δεδομένων. Ετσι, ενώ για μικρά μεγέθη δεδομένων εισόδου τα δυο μηχανήματα δίνουν την ίδια απόδοση, για μεγάλα μεγέθη ο χρόνος εκτέλεσης στο μηχάνημα με μνήμη 32 MB γίνεται ως και 15 φορές μεγαλύτερος από ό,τι στο μηχάνημα που διαθέτει 64 MB μνήμης.

1.2 Η Αιτία του Προβλήματος

Η δραματική αύξηση του χρόνου εκτέλεσης εφαρμογών που το σύνολο εργασίας τους δε χωρά στην κύρια μνήμη του υπολογιστή οφείλεται στη δομή του συστήματος διαχείρισης ιδεατής μνήμης, η οποία φαίνεται στο σχήμα 1.3. Η ιεραρχία μνήμης αποτελείται σε γενικές γραμμές από την κρυφή μνήμη, την κύρια μνήμη και το δίσκο, τα χαρακτηριστικά των οποίων φαίνονται στον πίνακα 1.1. Η κρυφή μνήμη είναι



Σχήμα 1.1: **Χρόνος εκτέλεσης της εφαρμογής QSORT σαν συνάρτηση του μεγέθους των δεδομένων εισόδου της σε μηχανήματα με 32 MB και 64 MB φυσικής μνήμης.** Μόλις το μέγεθος των δεδομένων εισόδου υπερβεί τα 18 MB αρχίζει η αποθήκευση σελίδων στο δίσκο στο μηχάνημα με τα 32 MB μνήμης με αποτέλεσμα την τεράστια αύξηση στο χρόνο εκτέλεσης.



Σχήμα 1.2: **Χρόνος εκτέλεσης της εφαρμογής FFT σαν συνάρτηση του μεγέθους των δεδομένων εισόδου της σε μηχανήματα με 32 MB και 64 MB φυσικής μνήμης.** Μόλις το μέγεθος των δεδομένων εισόδου υπερβεί τα 18 MB αρχίζει η αποθήκευση σελίδων στο δίσκο στο μηχάνημα με τα 32 MB μνήμης με αποτέλεσμα την τεράστια αύξηση στο χρόνο εκτέλεσης.

Τύπος μνήμης	Χρόνος πρόσβασης (σε κύκλους ρολογιού)	Χώρος αποθήκευσης (σε MB)	Κόστος (σε δραχμές ανά MB)
Κρυφή μνήμη	1	0.1 – 1	15000
Κύρια μνήμη	10	10 – 100	7000
Δίσκος	10^6	100 – 1000	70

Πίνακας 1.1: **Χαρακτηριστικά των επιπέδων της ιεραρχίας μνήμης.**

μια πολύ γρήγορη στατική μνήμη τυχαίας πρόσβασης με χρόνο προσπέλασης ίσο τυπικά με λίγους κύκλους ρολογιού, αλλά το μέγεθός της είναι πολύ μικρό λόγω του μεγάλου κόστους της ανά λέξη. Η κύρια μνήμη χρησιμοποιείται για την αποθήκευση δεδομένων που δε χωρούν στην κρυφή μνήμη. Είναι μια δυναμική μνήμη τυχαίας πρόσβασης με χρόνο προσπέλασης της τάξης των λίγων δεκάδων κύκλων ρολογιού και μέγεθος λίγων δεκάδων MB. Τέλος ο δίσκος χρησιμοποιείται για την αποθήκευση δεδομένων που δε χωρούν ούτε στην κύρια μνήμη. Το μέγεθός του είναι τυπικά ίσο με λίγες εκατοντάδες MB και ο χρόνος προσπέλασής του της τάξης των αρκετών ms, δηλαδή της τάξης των εκατομμυρίων κύκλων ρολογιού.

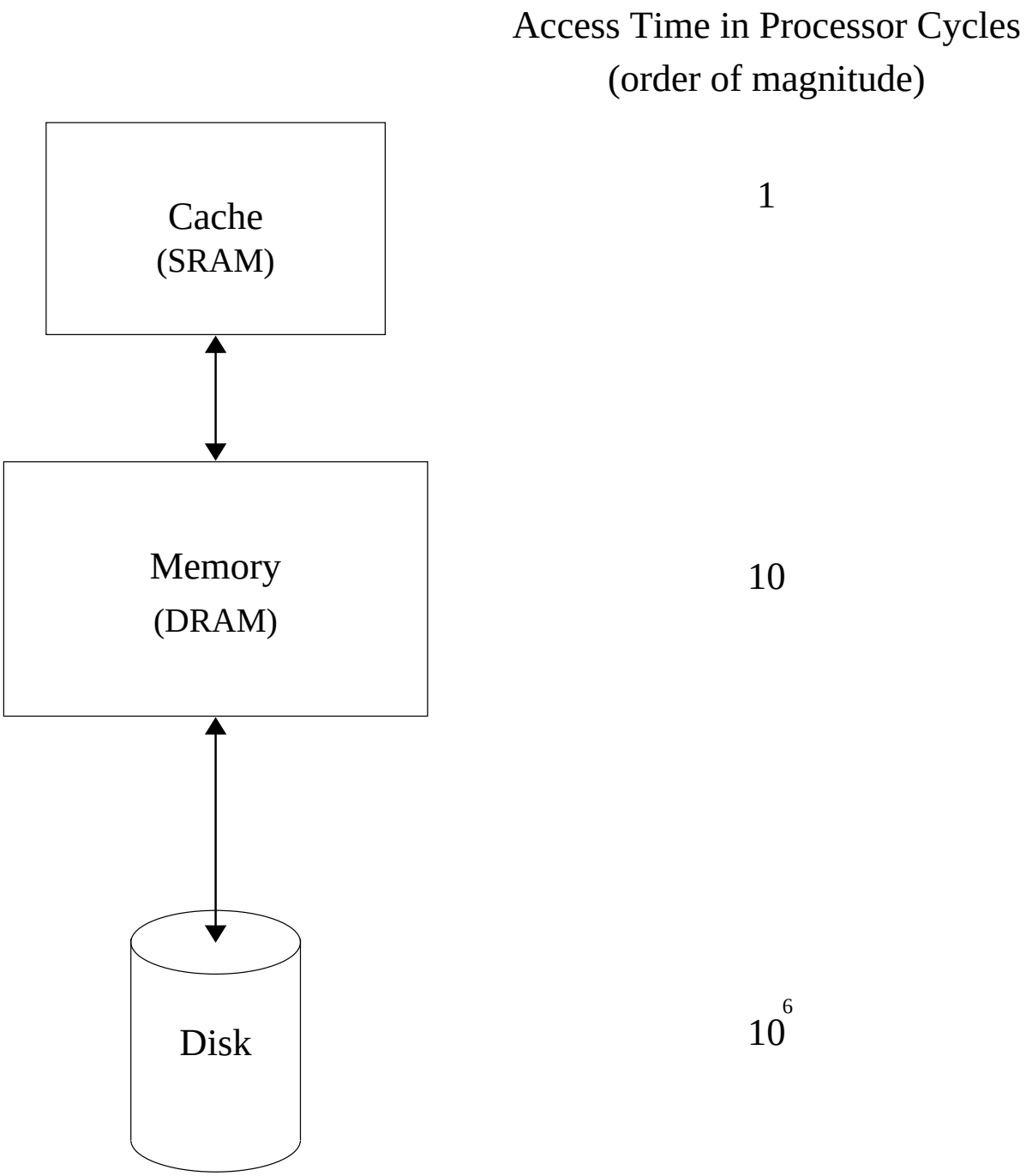
Παρατηρούμε ότι, ενώ η κύρια μνήμη παρέχει δυο τάξεις μεγέθους μεγαλύτερο αποθηκευτικό χώρο από την κρυφή μνήμη αυξάνοντας το χρόνο προσπέλασης στα δεδομένα κατά μια τάξη μεγέθους, ο δίσκος παρέχει μόλις μια τάξη μεγέθους μεγαλύτερο αποθηκευτικό χώρο από την κύρια μνήμη αυξάνοντας το χρόνο προσπέλασης στα δεδομένα κατά πέντε τάξεις μεγέθους. Είναι λοιπόν προφανές ότι εφαρμογές με σύνολο εργασίας που δε χωρά στη μνήμη αναγκάζονται να προσπελάσουν κάποια από τα δεδομένα τους από το δίσκο, με αποτέλεσμα τη δραματική αύξηση του χρόνου εκτέλεσής τους.

Το γεγονός αυτό μπορεί να οδηγήσει σε ένα φαινόμενο που ονομάζεται *παλινδρόμηση* κατά το οποίο η εκτελούμενη εφαρμογή κάνει συνεχώς λάθη σελίδας προσπαθώντας να προσπελάσει δεδομένα που δε βρίσκονται στην κύρια μνήμη και το σύστημα από ένα σημείο και μετά δεν κάνει τίποτα άλλο παρά μεταφέρει συνεχώς δεδομένα ανάμεσα στη μνήμη και το δίσκο. Η συχνότητα εμφάνισης του φαινομένου αυτού αυξάνεται με την πάροδο του χρόνου. Μερικές τάσεις που επικρατούν σήμερα στην αρχιτεκτονική υπολογιστών είναι οι εξής.

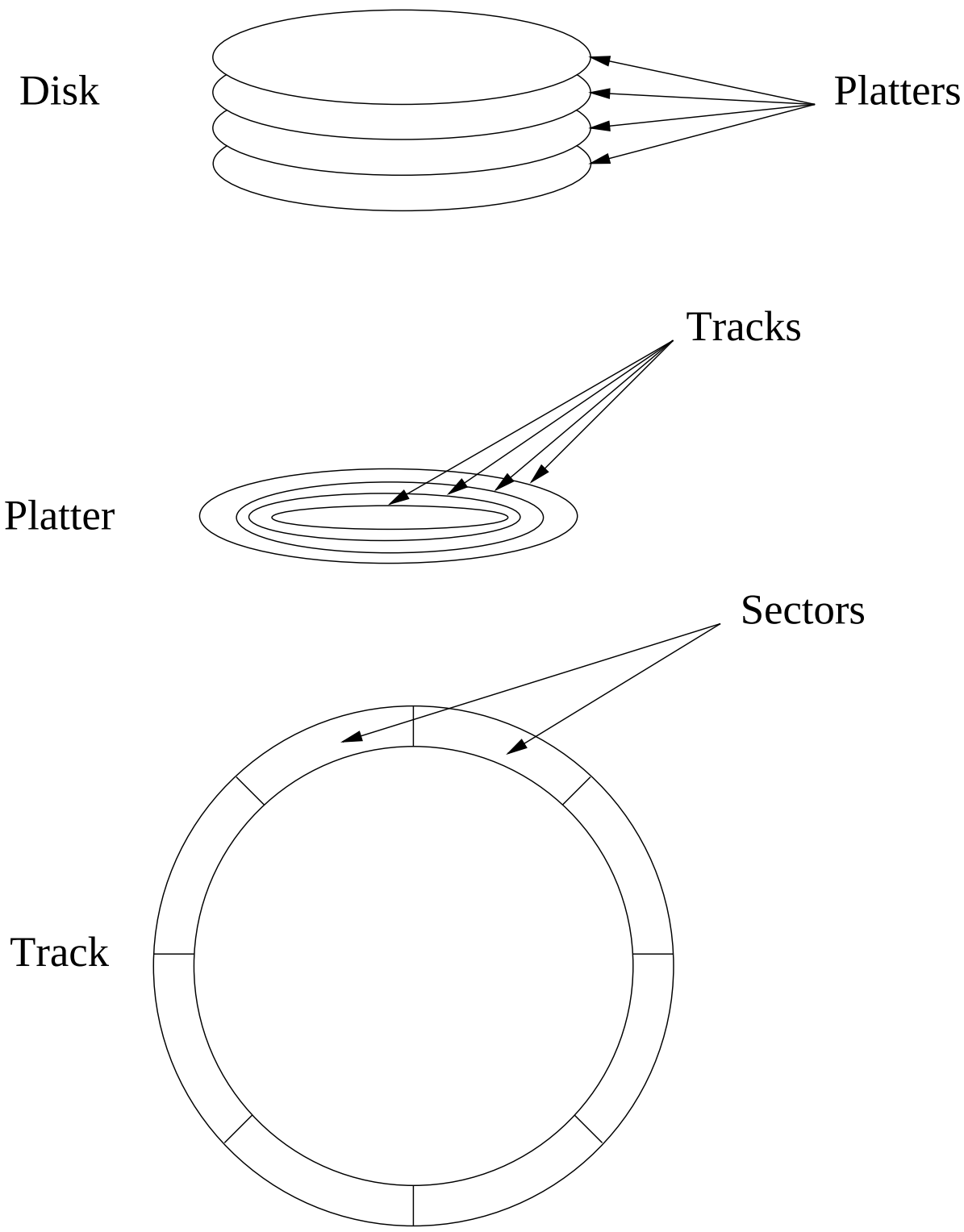
- Ο κύκλος ρολογιού μειώνεται ραγδαία.
- Το κόστος της κύριας μνήμης αποτελεί ολοένα και μεγαλύτερο μέρος του ολικού κόστους ενός σταθμού εργασίας [HP90].
- Ο χρόνος προσπέλασης στο δίσκο μειώνεται με πολύ αργό ρυθμό [HP90].
- Το μέγεθος της μνήμης που απαιτεί μια εφαρμογή για να δουλέψει αυξάνεται δραματικά.

Αυτό σημαίνει ότι αφενός όλο και περισσότερο ο χρήστης θα εκτελεί εφαρμογές που το σύνολο εργασίας τους δε θα χωρά στην κύρια μνήμη και αφετέρου ότι ο χρόνος προσπέλασης στο δίσκο, μετρημένος σε κύκλους ρολογιού, θα αυξάνεται συνεχώς, άρα το φαινόμενο της παλινδρόμησης θα γίνεται και συχνότερο και σοβαρότερο.

Ο λόγος για τον οποίο η προσπέλαση στο δίσκο είναι τόσο ακριβή οφείλεται στη δομή του που φαίνεται στο σχήμα 1.4. Για την προσπέλαση δεδομένων πρέπει αρχικά οι κεφαλές του δίσκου να μετακινηθούν μέχρι να φτάσουν στο σωστό ίχνος. Ο χρόνος αυτός ονομάζεται χρόνος αναζήτησης. Επίσης θα πρέπει να περιστραφεί αρκετά ο



Σχήμα 1.3: **Η δομή της ιεραρχίας μνήμης.**



Σχήμα 1.4: **Η δομή του δίσκου.**

δίσκος ώστε ο τομέας που περιέχει τα ζητούμενα δεδομένα να βρεθεί κάτω από την κεφαλή (χρόνος περιστροφής). Στη συνέχεια τα δεδομένα μεταφέρονται (γράφονται/διαβάζονται) από την κεφαλή. Επειδή ο δίσκος αποτελείται από μηχανικά μέρη, τόσο ο χρόνος αναζήτησης όσο και ο χρόνος περιστροφής είναι αρκετά σημαντικοί (της τάξης των αρκετών ms).

Για την αντιμετώπιση της μεγάλης καθυστέρησης της προσπέλασης στο δίσκο χρησιμοποιούνται διάφορες τεχνικές.

- *Χρησιμοποιούνται σελίδες μεγάλου μεγέθους ώστε ο χρόνος αναζήτησης και περιστροφής για τη μεταφορά της σελίδας από/προς το δίσκο να μοιράζεται σε πολλές προσπελάσεις στη σελίδα.* Η τεχνική αυτή ωφελεί προγράμματα που κάνουν πολλές προσπελάσεις ανά σελίδα (π.χ. εφαρμογές που προσπελαύνουν σειριακά μεγάλους πίνακες). Για προγράμματα όμως που έχουν λίγες μόνο προσπελάσεις ανά σελίδα, απλά επιτείνει το φαινόμενο της παλινδρόμησης αφού μεταφέρει πολλά δεδομένα ανά σελίδα που δε χρειάζονται ανάμεσα στη μνήμη και το δίσκο.
- *Χρησιμοποιούνται RAIDs [CLG+94].* Το κέρδος από τη χρήση τους είναι σημαντικό αφού μπορεί να γίνεται εγγραφή πολλών σελίδων παράλληλα. Ομως κατά την ανάγνωση της σελίδας, η καθυστέρηση που οφείλεται στο χρόνο αναζήτησης και περιστροφής παραμένει. Επίσης το κόστος των RAIDs είναι υψηλό.

Οι τεχνικές αυτές δεν αντιμετωπίζουν ριζικά το πρόβλημα, απλά προσπαθούν να μειώσουν τις επιπτώσεις του.

1.3 Η Προτεινόμενη Λύση

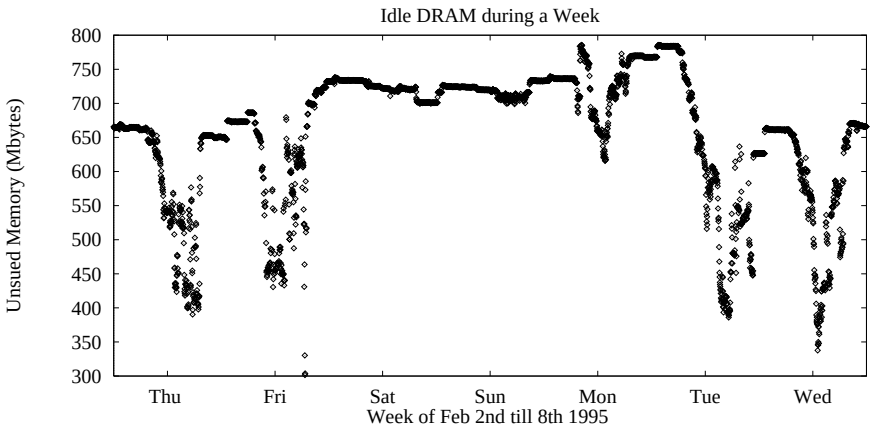
Η μόνη αποτελεσματική λύση για την αποφυγή της παλινδρόμησης είναι η παύση της χρησιμοποίησης του δίσκου για την αποθήκευση σελίδων μνήμης. Η χρήση του δίσκου γι' αυτό το σκοπό στα σημερινά λειτουργικά συστήματα αποτελεί κατάλοιπο της εποχής των πρώτων λειτουργικών συστημάτων που υλοποίησαν σύστημα διαχείρισης ιδεατής μνήμης. Εκείνη την εποχή η χρήση του δίσκου ήταν εφικτή για τους παρακάτω λόγους.

- Οι υπολογιστές είχαν ελάχιστη φυσική μνήμη.
- Τόσο οι επεξεργαστές όσο και η μνήμη ήταν πολύ πιο αργοί και έτσι ο χρόνος προσπέλασης στο δίσκο, μετρημένος σε κύκλους ρολογιού, δεν ήταν τόσο μεγάλος όσο σήμερα.
- Ο δίσκος ήταν μια από τις λίγες συσκευές που μπορούσε να προσφέρει σχετικά μεγάλο ρυθμό μεταγωγής δεδομένων και τυχαία πρόσβαση σ' αυτά.

Σήμερα, η καθυστέρηση της προσπέλασης του δίσκου έχει γίνει τόσο μεγάλη ώστε δεν μπορεί πια να χρησιμοποιηθεί για την αποθήκευση σελίδων μνήμης, γεγονός που φαίνεται έκδηλα και από τις γραφικές παραστάσεις 1.1 και 1.2.

Ο δίσκος λοιπόν πρέπει να αντικατασταθεί από κάποιο άλλο σύστημα. Το σύστημα αυτό, για να είναι επιτυχημένο, θα πρέπει να διαθέτει τα εξής χαρακτηριστικά.

- *Μεγάλη χωρητικότητα σε σχέση με τη μνήμη.* Πρακτικά, αυτό σημαίνει ότι θα πρέπει να παρέχει αποθηκευτικό χώρο μεγέθους αρκετών εκατοντάδων MB.
- *Υψηλό ρυθμό μεταγωγής δεδομένων.*



Σχήμα 1.5: **Η ελεύθερη μνήμη ενός τοπικού δικτύου 16 μηχανών με ολική μνήμη 800 MB κατά τη διάρκεια μιας εβδομάδας.**

- *Μικρή καθυστέρηση προσπέλασης.*

Ενα σύστημα που ανταποκρίνεται στις παραπάνω προδιαγραφές είναι το σύνολο της κύριας μνήμης των σταθμών εργασίας ενός τοπικού δικτύου που στο εξής θα ονομάζεται *απομακρυσμένη μνήμη*. Τα σύγχρονα δίκτυα [MK96, SL91, BJMW95, Del88, BCF+95, Gil95, BLA+94, Dol94] μπορούν να εγγυηθούν μεγάλο ρυθμό μεταγωγής δεδομένων και μικρή καθυστέρηση προσπέλασης. Μάλιστα ο ρυθμός μεταγωγής δεδομένων των δικτύων αυξάνεται κατά 30%−50% το χρόνο. Επίσης το ποσό της ελεύθερης μνήμης στο δίκτυο είναι αρκετό ώστε να μπορεί να καλύψει τις ανάγκες του συστήματος διαχείρισης ιδεατής μνήμης αρκετών σταθμών εργασίας. Αυτή η εκτίμηση επιβεβαιώνεται από το σχήμα 1.5 που δείχνει το συνολικό ποσό της ελεύθερης κύριας μνήμης των μηχανημάτων του τοπικού δικτύου της ομάδας Αρχιτεκτονικής και Συστημάτων VLSI κατά τη διάρκεια μιας εβδομάδας. Οπως αναμενόταν το ποσό της ελεύθερης μνήμης είναι πολύ μεγάλο κατά τις μη εργάσιμες ώρες και το Σαββατοκύριακο. Ακόμη όμως και κατά τις εργάσιμες ώρες των ημερών της εβδομάδας, το ποσό της ελεύθερης μνήμης είναι αρκετά σημαντικό και σπάνια μικρότερο από 400 MB. Ετσι, αφενός μπορούμε να πετύχουμε μεγαλύτερη χρησιμοποίηση της *υπάρχουσας* μνήμης και αφετέρου μπορούμε πάντα να καταφεύγουμε στον τοπικό δίσκο όποτε η ελεύθερη μνήμη δεν επαρκεί.

1.4 Η συνεισφορά αυτής της εργασίας

Η συνεισφορά της εργασίας αυτής σε σχέση με προηγούμενες εργασίες στον ίδιο τομέα συνίσταται στα παρακάτω:

- Παρουσιάζουμε μια νέα τεχνική, την τεχνική της *μεταβλητής ισοτιμίας* που περιγράφεται αναλυτικά στην υποενότητα 4.3.3, με την οποία μπορούμε να επιτύχουμε την αξιόπιστη αποθήκευση δεδομένων στην απομακρυσμένη μνήμη με ελάχιστη επιβάρυνση τόσο στο χρόνο εκτέλεσης όσο και στην επιπλέον χρησιμοποιούμενη μνήμη [MD96].

- Παρουσιάζουμε έναν αλγόριθμο σελιδοδιαχείρισης, ο οποίος είναι σε θέση να προσαρμόζεται στις διαρκώς μεταβαλλόμενες συνθήκες του περιβάλλοντος ενός

τοπικού δικτύου και να χρησιμοποιεί κάθε φορά για την αποθήκευση σελίδων τις συσκευές και τις πολιτικές που δίνουν την καλύτερη απόδοση [MD95].

- Επιτυγχάνουμε αποδοτική αξιόπιστη αποθήκευση δεδομένων σε σύστημα απομακρυσμένης μνήμης χωρίς τη χρήση του τοπικού δίσκου [DM96].
- Υλοποιούμε έναν αποδοτικό σελιδοδιαχειριστή απομακρυσμένης μνήμης έξω από τον πυρήνα του λειτουργικού συστήματος σε μονολιθικό σύστημα, όπως περιγράφεται στο κεφάλαιο 2.

Τα υπόλοιπα κεφάλαια της εργασίας αυτής περιγράφουν τη σχεδίαση, την υλοποίηση και την αξιολόγηση ενός συστήματος που χρησιμοποιεί την ελεύθερη μνήμη του δικτύου για την αποθήκευση σελίδων. Παρουσιάζονται διάφορα προβλήματα που προέκυψαν, όπως η ανάγκη ανθεκτικότητας του συστήματος σε βλάβες, και οι λύσεις που εφαρμόστηκαν. Επίσης παρουσιάζονται διάφορες πολιτικές αξιοπιστίας καθώς και η απόδοσή τους σε δίκτυα διαφόρων ρυθμών μεταγωγής δεδομένων. Τέλος περιγράφονται διάφορα σχετικά συστήματα και παρουσιάζονται τα συμπεράσματα που προέκυψαν από την εργασία αυτή.

Κεφάλαιο 2

Σχεδίαση

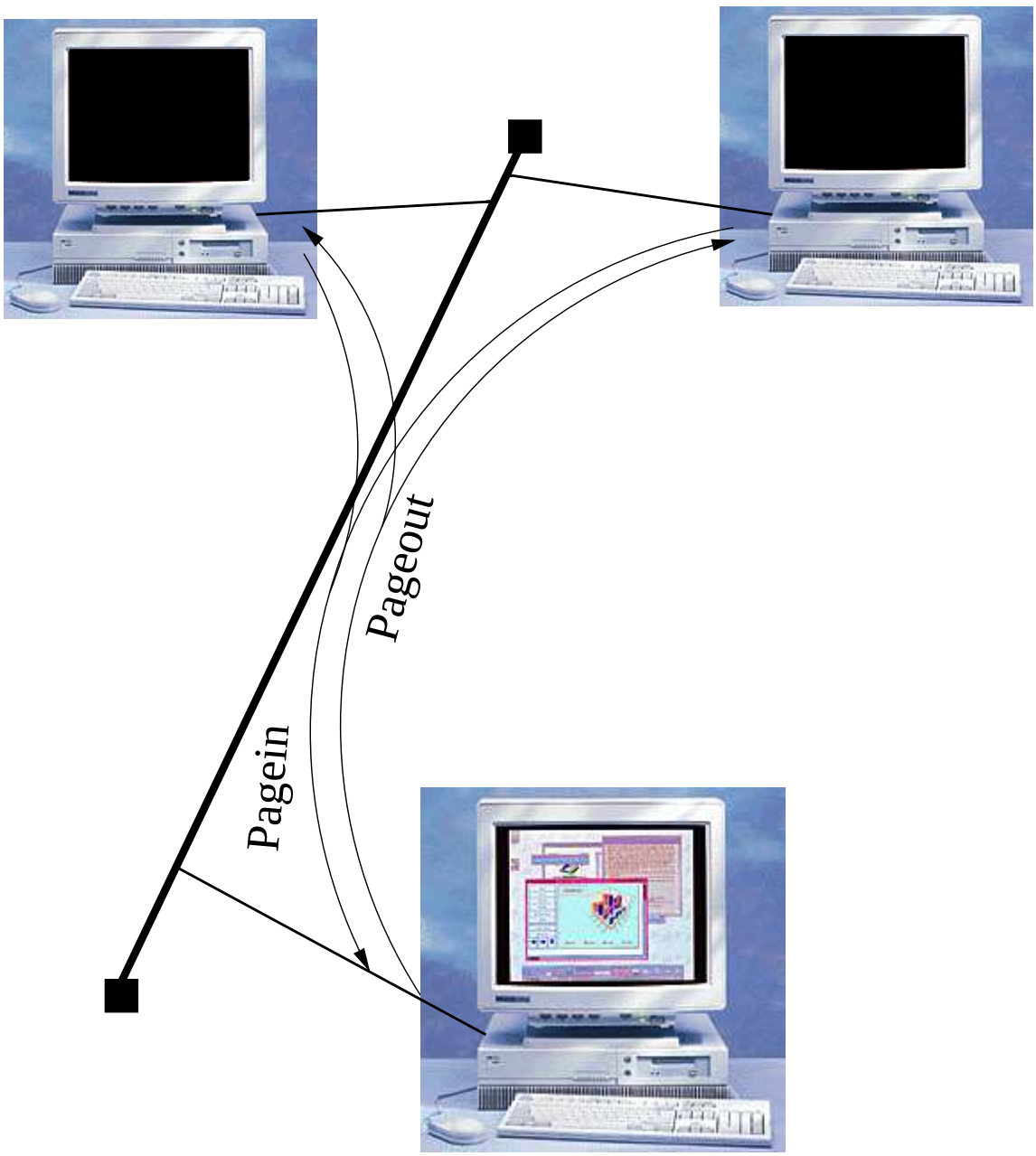
2.1 Στόχοι Σχεδίασης

Το προτεινόμενο σύστημα θα πρέπει να έχει καταρχήν τη δυνατότητα μεταφοράς σελίδων ανάμεσα στην τοπική και την απομακρυσμένη μνήμη, τη μνήμη δηλαδή άλλων μηχανών του δικτύου, με τρόπο διαφανή προς τις εφαρμογές του χρήστη. Δηλαδή οι εφαρμογές δε θα πρέπει να χρειάζονται νέα μεταγλώττιση ή σύνδεση προκειμένου να χρησιμοποιήσουν την απομακρυσμένη μνήμη. Αυτό σημαίνει ότι η μεταφορά των σελίδων πάνω από το δίκτυο θα πρέπει να γίνεται από το λειτουργικό σύστημα. Ετσι, προσθέτοντας κώδικα μέσα στον πυρήνα του λειτουργικού συστήματος, μπορούμε να ελέγξουμε τη μεταφορά των σελίδων που τυπικά γίνεται ανάμεσα στην κύρια μνήμη και στο δίσκο και να χρησιμοποιήσουμε την απομακρυσμένη μνήμη για την εκτέλεση της μεταφοράς αυτής.

Ενα τμήμα λοιπόν του συστήματος αποτελεί ο *σελιδοδιαχειριστής* που ελέγχει την κίνηση δεδομένων ανάμεσα στην κύρια μνήμη και την απομακρυσμένη μνήμη. Ενα άλλο τμήμα του συστήματος αποτελεί το σύστημα αποθήκευσης των σελίδων σε κάποιο απομακρυσμένο κόμβο. Τη λειτουργία αυτή αναλαμβάνει σε κάθε κόμβο του δικτύου ένας δαίμονας, ο *εξυπηρετητής μνήμης*. Επίσης το σύστημα θα πρέπει να δίνει τη δυνατότητα σε όλα τα μηχανήματα του δικτύου να μπορούν να χρησιμοποιηθούν είτε ως εξυπηρετητές μνήμης είτε ως πελάτες που χρησιμοποιούν τους εξυπηρετητές αυτούς, ανάλογα με το φόρτο μνήμης που έχουν κάθε στιγμή, όπως φαίνεται στο σχήμα 2.1.

Επίσης η πρόσθεση κώδικα στο λειτουργικό σύστημα θα πρέπει να γίνει με τρόπο τέτοιο ώστε το σύστημά μας να μπορεί να τρέξει και σε μηχανήματα που δε διαθέτουν τον πηγαίο κώδικα του λειτουργικού συστήματος. Το τμήμα του κώδικα που εξαρτάται από το λειτουργικό σύστημα θα πρέπει να είναι όσο το δυνατό πιο περιορισμένο σε έκταση ώστε να είναι εύκολη η μεταφορά του και σε μηχανήματα διαφορετικής αρχιτεκτονικής με διαφορετικό λειτουργικό σύστημα. Τέλος η ανάπτυξη του συστήματός μας δε θα πρέπει να απαιτεί την ύπαρξη πηγαίου κώδικα του λειτουργικού συστήματος αφού δε διαθέτουμε τον πηγαίο κώδικα του DEC OSF/1, το οποίο χρησιμοποιήθηκε για την ανάπτυξή του.

Οι επόμενες ενότητες περιγράφουν το περιβάλλον ανάπτυξης του συστήματός μας, το σελιδοδιαχειριστή και τον εξυπηρετητή μνήμης.



Σχήμα 2.1: **Σε κάθε χρονική στιγμή, τα μηχανήματα με μεγάλο φόρτο μνήμης ενεργούν σαν πελάτες ενώ τα μηχανήματα με αρκετή ελεύθερη μνήμη σαν εξυπηρετητές.**

2.2 Περιβάλλον Ανάπτυξης

Για την ανάπτυξη του συστήματος χρησιμοποιήθηκε ένα τοπικό δίκτυο από μηχανήματα τύπου DEC 3000 model 300 [Dig93a] με 32 MB φυσικής μνήμης, τα οποία βασίζονται στον επεξεργαστή Alpha της DEC, συνδεμένα με Ethernet [IEE85a]. Το λειτουργικό σύστημα που χρησιμοποιήθηκε είναι το DEC OSF/1 [Dig93b], έκδοση 3.2. Ο δίσκος που χρησιμοποιήθηκε για την αποθήκευση σελίδων μνήμης είναι ο DEC RZ55 με ρυθμό μεταγωγής δεδομένων ίσο με 10 Mbps και μέσο χρόνο αναζήτησης ίσο με 17 ms.

2.3 Ο Σελιδοδιαχειριστής

Ο σελιδοδιαχειριστής, όπως φάνηκε παραπάνω, θα πρέπει να αποτελεί τμήμα του λειτουργικού συστήματος για να μπορεί να έχει πρόσβαση στις σελίδες μνήμης των εφαρμογών με ένα διαφανή τρόπο. Για την υλοποίησή του, μπορούν να ακολουθηθούν δυο κατευθύνσεις. Η πρώτη επιλογή είναι η τροποποίηση του σελιδοδιαχειριστή του λειτουργικού συστήματος. Η δεύτερη είναι η υλοποίηση του σελιδοδιαχειριστή του συστήματός μας σαν οδηγός συσκευής.

Στην πρώτη περίπτωση, ο σελιδοδιαχειριστής θα είχε τη δυνατότητα όχι μόνο μεταφοράς των σελίδων αλλά και λήψης αποφάσεων που σχετίζονται με την πολιτική που εφαρμόζει ο αλγόριθμος αντικατάστασης. Επίσης, αν διαθέταμε δίκτυο που να επιτρέπει αποδοτικές απομακρυσμένες εγγραφές/αναγνώσεις σε επίπεδο λέξης [MK96, Dol94], θα μπορούσαμε να αποφύγουμε τη μεταφορά ολόκληρης της σελίδας όταν γίνεται λάθος σελίδας μεταφέροντας μόνο τη λέξη που προκάλεσε το λάθος αν έχουμε λόγο να πιστεύουμε ότι τα υπόλοιπα δεδομένα της σελίδας πιθανότατα δε θα προσπελαστούν από την εφαρμογή. Επιπλέον, κάθε εφαρμογή θα μπορούσε να καθορίζει ποιες από τις σελίδες της μπορούν να αποθηκευτούν στην απομακρυσμένη μνήμη και ποιες όχι, δίνοντας με αυτό τον τρόπο την ίδια ευελιξία που προσφέρουν και οι εξωτερικοί εξυπηρετητές μνήμης. Το μειονέκτημα της προσέγγισης αυτής είναι ότι απαιτεί την τροποποίηση του πηγαίου κώδικα του λειτουργικού συστήματος και άρα το σύστημα δεν είναι μεταφέρσιμο ούτε καν σε μηχανήματα του ίδιου τύπου που δε διαθέτουν τον πηγαίο κώδικα του λειτουργικού συστήματος.

Στη δεύτερη περίπτωση, το λειτουργικό σύστημα αντιμετωπίζει το σελιδοδιαχειριστή σαν μια συσκευή την οποία χρησιμοποιεί για αποθήκευση σελίδων. Ετσι ο σελιδοδιαχειριστής έχει έλεγχο μόνο στη μεταφορά σελίδων μεταξύ της κύριας μνήμης και της απομακρυσμένης μνήμης. Το πλεονέκτημα της προσέγγισης αυτής είναι το γεγονός ότι είναι πλήρως μεταφέρσιμος και μπορεί να δουλέψει σε κάθε μηχάνημα τύπου DEC 3000 με λειτουργικό σύστημα το DEC OSF/1. Επίσης, επειδή το τμήμα του κώδικα που εξαρτάται από το λειτουργικό σύστημα είναι πολύ μικρό, είναι πολύ εύκολη η μεταφορά του σε μηχανήματα με διαφορετική αρχιτεκτονική και λειτουργικό σύστημα.

Λαμβάνοντας υπόψη όλους τους παραπάνω παράγοντες επιλέχθηκε η δεύτερη λύση για τους εξής λόγους.

- *Εξασφαλίζεται η μεταφερσιμότητα του συστήματος.*
- *Δεν απαιτεί την αλλαγή πηγαίου κώδικα του πυρήνα.*

- *Αποτελεί μια πιο αντικειμενοστρεφή προσέγγιση, αφού ο πυρήνας δε γνωρίζει καν ότι χρησιμοποιεί τη μνήμη άλλων κόμβων του δικτύου για να αποθηκεύσει σελίδες του.* Απλά χρησιμοποιεί τον οδηγό συσκευής με τον ίδιο ακριβώς τρόπο με τον οποίο θα χρησιμοποιούσε και τον οδηγό ενός δίσκου.
- *Η έλλειψη της δυνατότητας μεταβολής της πολιτικής αντικατάστασης σελίδων δεν είναι τόσο σημαντική (όλα τα συστήματα χρησιμοποιούν μια προσέγγιση της πολιτικής LRU).* Επίσης η πολιτική αυτή καθορίζει απλά *ποιες* σελίδες θα αντικατασταθούν κάτι το οποίο δεν έχει άμεση σχέση με το *που* θα αποθηκευτούν οι σελίδες αυτές.
- *Η έλλειψη της δυνατότητας απομακρυσμένων εγγραφών/αναγνώσεων από το δίκτυο στο οποίο έγινε η ανάπτυξη του συστήματος απενεργοποιεί ουσιαστικά το κυριότερο πλεονέκτημα της πρώτης προσέγγισης.*

Ο σελιδοδιαχειριστής λοιπόν υλοποιείται σαν οδηγός συσκευής, τον κώδικα του οποίου εκτελεί ένας δαίμονας που βρίσκεται πάντα σε κατάσταση πυρήνα και ο οποίος αναλαμβάνει τη σύνδεση με τους εξυπηρετητές μνήμης και τη μεταφορά των σελίδων μέσα από τον πυρήνα του λειτουργικού συστήματος. Το κεφάλαιο 3 περιγράφει αναλυτικά την υλοποίησή του και τη διασύνδεσή του με τα διάφορα τμήματα του λειτουργικού συστήματος. Επίσης περιγράφει διάφορα θέματα που ανέκυψαν κατά την υλοποίηση.

2.4 Ο Εξυπηρετητής Μνήμης

Ο εξυπηρετητής μνήμης είναι απλά ένα πρόγραμμα που εκτελείται σε κατάσταση χρήστη, δέχεται αιτήσεις για μεταφορά σελίδων από τους πελάτες που ζητούν να συνδεθούν με αυτόν και περιοδικά ενημερώνει τους πελάτες του για το ποσό της διαθέσιμης μνήμης στο μηχάνημά του. Ετσι οι πελάτες μπορούν να χρησιμοποιήσουν τις υπηρεσίες του αφενός για να αποθηκεύσουν ή να ανακτήσουν σελίδες και αφετέρου για να πληροφορηθούν το ποσό της ελεύθερης μνήμης στον κόμβο του, ώστε να πάρουν, αν χρειαστεί, αποφάσεις για να επιλέξουν τους καταλληλότερους κόμβους όπου θα αποθηκεύσουν τα δεδομένα τους.

Κεφάλαιο 3

Υλοποίηση

3.1 Η Υλοποίηση του Σελιδοδιαχειριστή

3.1.1 Γενική Περιγραφή

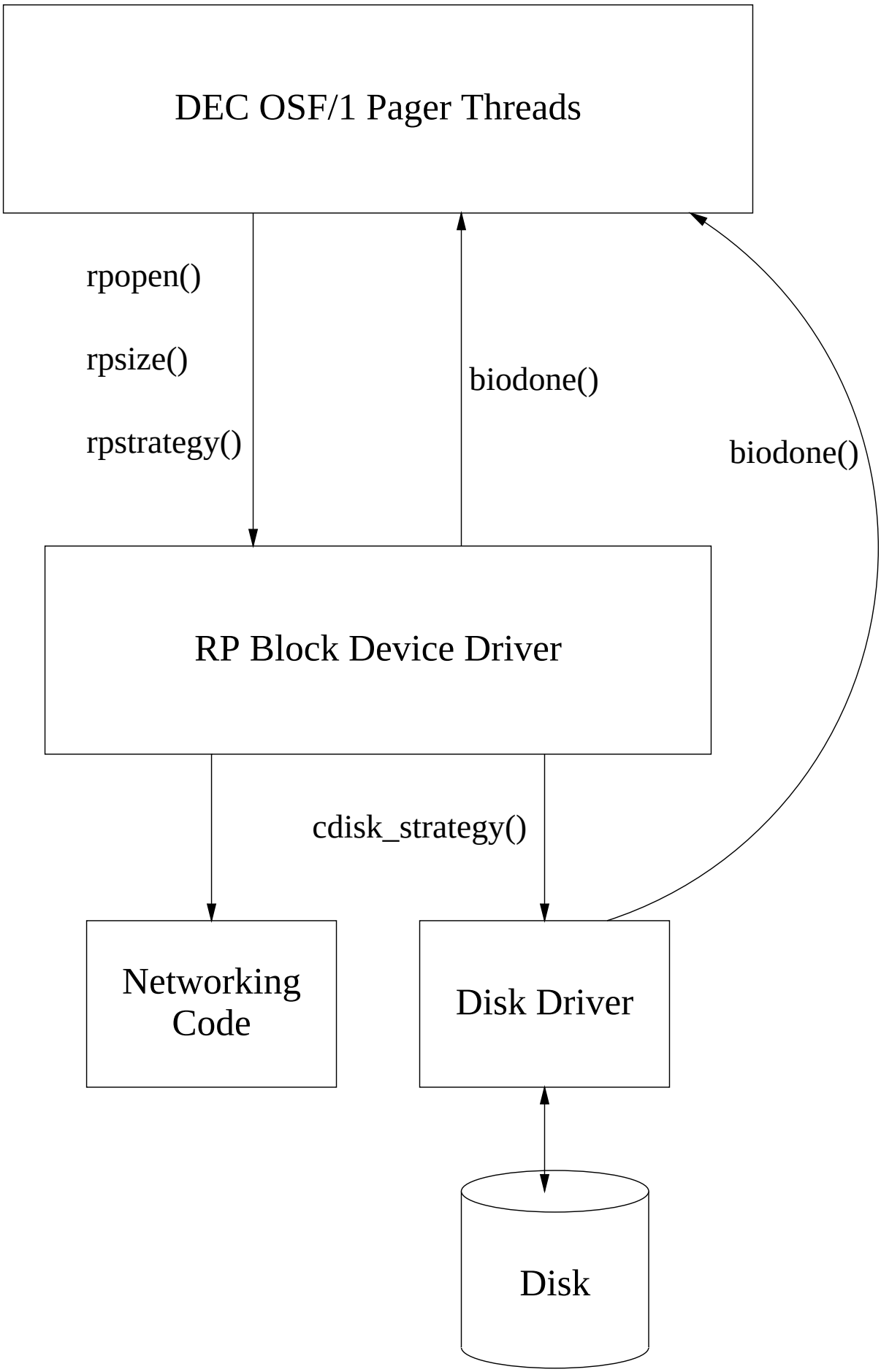
Ο σελιδοδιαχειριστής υλοποιείται σαν οδηγός συσκευής, όπως φαίνεται στο σχήμα 3.1. Υλοποιείται δηλαδή σαν ένα στρώμα λογισμικού. Το αμέσως ανώτερο στρώμα του λειτουργικού συστήματος με το οποίο αλληλεπιδρά είναι ο σελιδοδιαχειριστής του λειτουργικού συστήματος. Το αμέσως κατώτερο στρώμα αποτελείται από τον οδηγό του τοπικού δίσκου και τον κώδικα που υλοποιεί τη διασύνδεση του υπολογιστή με το δίκτυο. Ενας δαίμονας, ο οποίος βρίσκεται συνεχώς σε κατάσταση πυρήνα, εκτελεί τον κώδικα του οδηγού μας, εξασφαλίζοντας τη μεταφορά των σελίδων μεταξύ του σελιδοδιαχειριστή του DEC OSF/1 και των συσκευών αποθήκευσης σελίδων, δηλαδή του δικτύου και του δίσκου.

Στις επόμενες υποενότητες περιγράφεται αναλυτικά ο τρόπος διασύνδεσης του σελιδοδιαχειριστή μας με τα υπόλοιπα τμήματα του λειτουργικού συστήματος.

3.1.2 Η Διασύνδεση με το Σελιδοδιαχειριστή του DEC OSF/1

Ο σελιδοδιαχειριστής του DEC OSF/1 είναι το τμήμα του λειτουργικού συστήματος που διαχειρίζεται την κύρια μνήμη του υπολογιστή. Ενεργοποιείται όποτε το ποσό της ελεύθερης φυσικής μνήμης είναι λιγότερο από ένα κάτω όριο και όποτε κάποια εφαρμογή προσπελάσει κάποια σελίδα της, η οποία δε βρίσκεται στη μνήμη. Στην πρώτη περίπτωση, επιλέγει ένα προκαθορισμένο αριθμό σελίδων, χρησιμοποιώντας έναν αλγόριθμο αντικατάστασης που προσεγγίζει τον LRU, και ζητά από τον οδηγό της συσκευής που χρησιμοποιείται για αποθήκευση σελίδων την εγγραφή των συγκε−κριμένων σελίδων σε κάποια μπλοκ της συσκευής. Στη δεύτερη περίπτωση, αναζητά στις δομές δεδομένων που διατηρεί τα μπλοκ της συσκευής που αντιστοιχούν στη σελίδα που προκάλεσε το λάθος σελίδας και ζητά από τον οδηγό της συσκευής να διαβάσει τα συγκεκριμένα μπλοκ.

Η υλοποίηση του σελιδοδιαχειριστή του λειτουργικού συστήματος στο DEC OSF/1, όπως και στα περισσότερα λειτουργικά συστήματα που βασίζονται στο Mach [ABB+86], είναι πολυνηματική. Συγκεκριμένα, στην έκδοση 3.2, ο σελιδοδιαχειριστής του συστήματος αποτελείται από 4 νήματα. Αυτό σημαίνει ότι ο οδηγός θα πρέπει να είναι σε θέση να χειριστεί πολλές αιτήσεις για μεταφορά σελίδων ταυτόχρονα. Επίσης ο σελιδοδιαχειριστής του DEC OSF/1 χρησιμοποιεί πρώιμη μεταφορά, δηλαδή συχνά



Σχήμα 3.1: **Η σύνδεση του σελιδοδιαχειριστή και η αλληλεπίδρασή του με το λειτουργικό σύστημα.**

Συνάρτηση	Υλοποιείται από	Καλείται από
xxxopen()	οδηγό συσκευής	πυρήνα
xxxsize()	οδηγό συσκευής	πυρήνα
xxxstrategy()	οδηγό συσκευής	πυρήνα
biodone()	πυρήνα	οδηγό συσκευής

Πίνακας 3.1: **Η διασύνδεση μιας συσκευής μπλοκ με το λειτουργικό σύστημα.** Το xxx είναι ένα όνομα που εξαρτάται από τη συσκευή. Στην περίπτωσή μας επιλέχθηκε το rp (από το Remote Pager).

ζητά την ανάγνωση περισσοτέρων της μιας σελίδας σε ένα λάθος σελίδας, με την ελπίδα ότι οι επιπλέον σελίδες που θα διαβαστούν θα αποδειχθούν χρήσιμες, δηλαδή θα προσπελαστούν από την εφαρμογή που προκάλεσε το λάθος σελίδας. Τέλος, για να μειώσει την επίδραση του χρόνου αναζήτησης ανά σελίδα, κατά κανόνα ζητά την εγγραφή πολλών σελίδων ανά αίτηση εγγραφής (τυπικά 4).

Από τα παραπάνω γίνεται φανερό ότι ο πυρήνας του λειτουργικού συστήματος δεν έχει γνώση για τη συσκευή που χρησιμοποιεί προκειμένου να αποθηκεύει σελίδες. Αρκεί να υπάρχει μια συσκευή μπλοκ την οποία να μπορεί να χρησιμοποιήσει. Η διασύνδεση λοιπόν ανάμεσα στον πυρήνα (δηλαδή στο σελιδοδιαχειριστή του λειτουργικού συστήματος) και στη συσκευή (στην προκειμένη περίπτωση το δικό μας σελιδοδιαχειριστή) είναι σαφέστατα ορισμένη και αποτελείται από ένα σύνολο μόλις τεσσάρων ρουτινών, όπως φαίνεται στον πίνακα 3.1.

Η συνάρτηση *rpopen( )* καλείται από τον πυρήνα μόνο μια φορά, όταν δηλαδή ο υπερχρήστης καθορίσει ότι η συγκεκριμένη συσκευή θα χρησιμοποιείται για αποθήκευση σελίδων. Ο σκοπός της είναι η αρχικοποίηση των δομών δεδομένων που διατηρεί ο οδηγός της συσκευής. Η συνάρτηση *rpsize( )* καλείται από το σελιδοδιαχειριστή του λειτουργικού συστήματος μόνο μια φορά. Η συνάρτηση αυτή επιστρέφει το μέγεθος της συσκευής, ένα στοιχείο δηλαδή που είναι απαραίτητο να το γνωρίζει ο πυρήνας.

Οι υπόλοιπες δυο συναρτήσεις είναι αυτές που κάνουν την πραγματική μεταφορά. Η *rpstrategy( )* καλείται από τα νήματα του σελιδοδιαχειριστή του λειτουργικού συστήματος προκειμένου να κάνουν μια αίτηση εξυπηρέτησης στον οδηγό της συσκευής. Η συνάρτηση αυτή έχει σαν όρισμα μια δομή που περιέχει πληροφορίες για το ποια μπλοκ της συσκευής θα μεταφερθούν και ένα σύνολο σελίδων μνήμης όπου θα μεταφερθούν τα προς ανάγνωση δεδομένα ή από όπου θα ληφθούν τα προς εγγραφή δεδομένα. Τυπικά η συνάρτηση αυτή απλά τοποθετεί την αίτηση σε μια ουρά που διατηρεί ο οδηγός και επιστρέφει, προκειμένου το νήμα που την κάλεσε να συνεχίσει την εργασία του.

Ενας δαίμονας, ο οποίος εκτελείται πάντα σε κατάσταση πυρήνα, αναλαμβάνει να απομακρύνει τις αιτήσεις από την ουρά αυτή και να τις εξυπηρετήσει. Οταν μια αίτηση εξυπηρετηθεί, δηλαδή όταν τα δεδομένα έχουν πια μεταφερθεί, καλεί τη συνάρτηση *biodone( )* με όρισμα τη συγκεκριμένη αίτηση. Με αυτόν τον τρόπο ο πυρήνας ενημερώνεται ότι η μεταφορά έχει ολοκληρωθεί. Αυτό, στην περίπτωση εγγραφής, σημαίνει ότι οι σελίδες μνήμης στις οποίες αναφέρεται η αίτηση είναι ελεύθερες για επαναχρησιμοποίηση. Στην περίπτωση ανάγνωσης, αυτό σημαίνει ότι τα περιεχόμενα των σελίδων στις οποίες αναφέρεται η αίτηση είναι έγκυρα, άρα η εφαρμογή που έχει προκαλέσει λάθος σελίδας προσπελαύνοντας κάποια από τις σελίδες αυτές μπορεί να συνεχίσει την εκτέλεσή της.

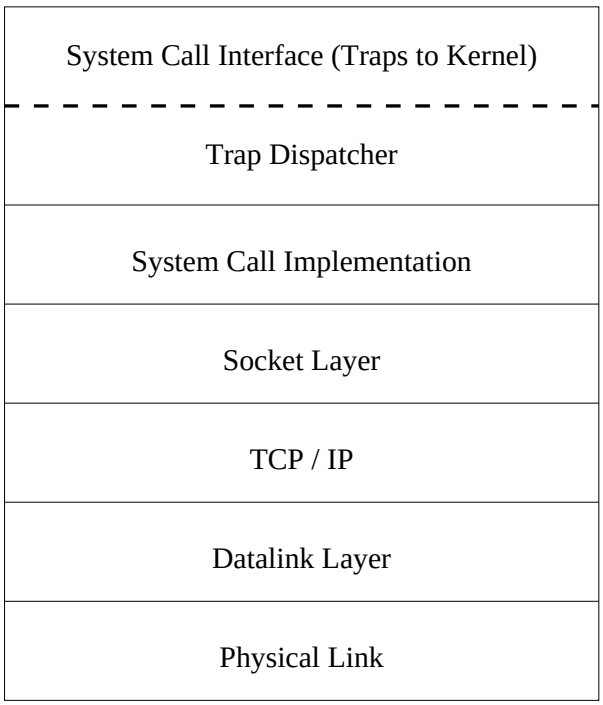
3.1.3 Η Διασύνδεση με το Δίκτυο

Ο δαίμονας που αναλαμβάνει την ικανοποίηση των αιτήσεων του σελιδοδιαχειριστή του λειτουργικού συστήματος θα πρέπει να έχει τη δυνατότητα να επικοινωνεί με τους εξυπηρετητές μνήμης προκειμένου να μεταφέρει τις σελίδες. Γι' αυτό θα πρέπει με κάποιο τρόπο να χρησιμοποιεί τον κώδικα του λειτουργικού συστήματος που υλοποιεί τη σύνδεση με το δίκτυο. Για την υλοποίηση πάνω από το Ethernet επιλέχθηκε το πρωτόκολλο TCP/IP [Pos81b, Pos81a], αφού είναι το μόνο πρωτόκολλο αξιόπιστης μεταφοράς δεδομένων που διαθέτουμε.

Η στοίβα των στρωμάτων λογισμικού που επιτρέπουν την επικοινωνία πάνω από το δίκτυο σε ένα τυπικό σύστημα Unix [Bac86, LMKQ89] φαίνεται στο σχήμα 3.2. Το ανώτερο στρώμα εκτελείται σε κατάσταση χρήστη και αποτελείται από τις συναρτήσεις που παρέχει η βιβλιοθήκη του συστήματος, οι οποίες τελικά προκαλούν μια κλήση συστήματος στον πυρήνα (*open( ), read( ), write( )* κ.λ.π.). Στη συνέχεια, μέσα στο λειτουργικό σύστημα υλοποιούνται τα υπόλοιπα στρώματα. Ο διαχειριστής των κλήσεων συστήματος καλεί την αντίστοιχη συνάρτηση που υλοποιεί τη συγκεκριμένη κλήση (*socket( ), connect( ), read( ), write( )* κ.λ.π.). Οι ρουτίνες αυτού του στρώματος τυπικά αντιγράφουν δεδομένα ανάμεσα στο χώρο διευθύνσεων του πυρήνα και στο χώρο διευθύνσεων του χρήστη και χρησιμοποιούν τις υπηρεσίες του επόμενου στρώματος, που ονομάζεται στρώμα πρίζας. Οι συναρτήσεις του στρώματος αυτού (*socreate( ), soconnect( ), sosend( ), soreceive( )* κ.λ.π.) αναλαμβάνουν τη διαχείρηση των ενταμιευτών που χρησιμοποιούνται για τη μεταφορά δεδομένων ανάμεσα στο χώρο διευθύνσεων του πυρήνα και στην πλακέτα διασύνδεσης του δικτύου και την υλοποίηση στοιχειωδών υπηρεσιών (π.χ. έναρξη σύνδεσης, αποστολή και λήψη δεδομένων). Το αμέσως επόμενο στρώμα αποτελούν τα πρωτόκολλα TCP και IP. Στη συνέχεια υπάρχει το στρώμα διασύνδεσης δεδομένων και ο οδηγός συσκευής της πλακέτας διασύνδεσης του δικτύου.

Ο δαίμονας που έχει το ρόλο του σελιδοδιαχειριστή για τη συσκευή μας προφανώς δεν μπορεί να χρησιμοποιήσει το ανώτερο στρώμα λογισμικού, αφού εκτελείται σε κατάσταση πυρήνα. Παράλληλα δεν μπορεί να χρησιμοποιήσει το υψηλού επιπέδου στρώμα υλοποίησης των κλήσεων συστήματος, αφού οι αντιγραφές των δεδομένων ανάμεσα στους ενταμιευτές των πριζών και στις σελίδες μνήμης των αιτήσεων που εξυπηρετούνται είναι αντιγραφές από το χώρο διευθύνσεων του πυρήνα στο χώρο διευθύνσεων του πυρήνα και όχι μεταξύ των χώρων διευθύνσεων του πυρήνα και κάποιας διεργασίας του χρήστη. Ετσι χρησιμοποιούνται οι συναρτήσεις του στρώματος πρίζας ενώ η διαχείριση των ενταμιευτών γίνεται απευθείας από το σελιδοδιαχειριστή μας.

Το πρωτόκολλο TCP και ο τρόπος με τον οποίο χρησιμοποιείται από τα σημερινά λειτουργικά συστήματα έχουν σαν αποτέλεσμα την ύπαρξη σημαντικής επιβάρυνσης, η οποία οφείλεται κυρίως στις τέσσερις αντιγραφές δεδομένων που πραγματοποιούνται ανά μεταφορά, όπως φαίνεται στο σχήμα 3.3. Η επιβάρυνση αυτή, που μετρήθηκε να είναι ίση με 1.6 ms ανά σελίδα (8 KB), μπορεί να μην αποτελεί μεγάλο πρόβλημα για το έτσι κι αλλιώς αργό Ethernet, όμως για δίκτυα μεγαλύτερου ρυθμού μεταγωγής δεδομένων είναι απαγορευτική, όπως φαίνεται στο σχήμα 3.4. Αυτό συμβαίνει γιατί όσο αυξάνεται ο ρυθμός μεταγωγής δεδομένων του δικτύου, τόσο μειώνεται ο χρόνος που απαιτείται για τη μεταφορά μιας σελίδας πάνω από το φυσικό μέσο μετάδοσης και τελικά ο χρόνος αυτός γίνεται αμελητέος σε σχέση με το χρόνο που ξοδεύεται από το λογισμικό και που αφιερώνεται κυρίως στην αντιγραφή δεδομένων. Παρόλα αυτά, χρησιμοποιήθηκε στην υλοποίησή μας αφού δεν είχαμε στη διάθεσή μας κάποιο

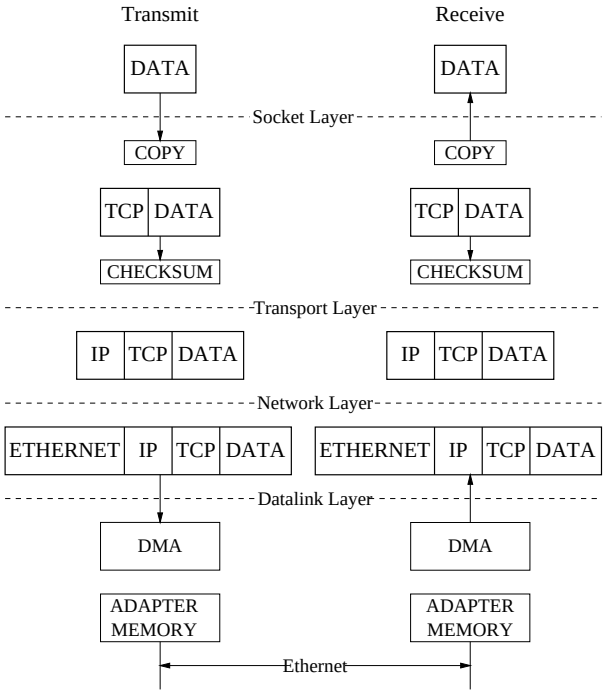


Σχήμα 3.2: **Τα στρώματα συναρτήσεων του κώδικα δικτύωσης.**

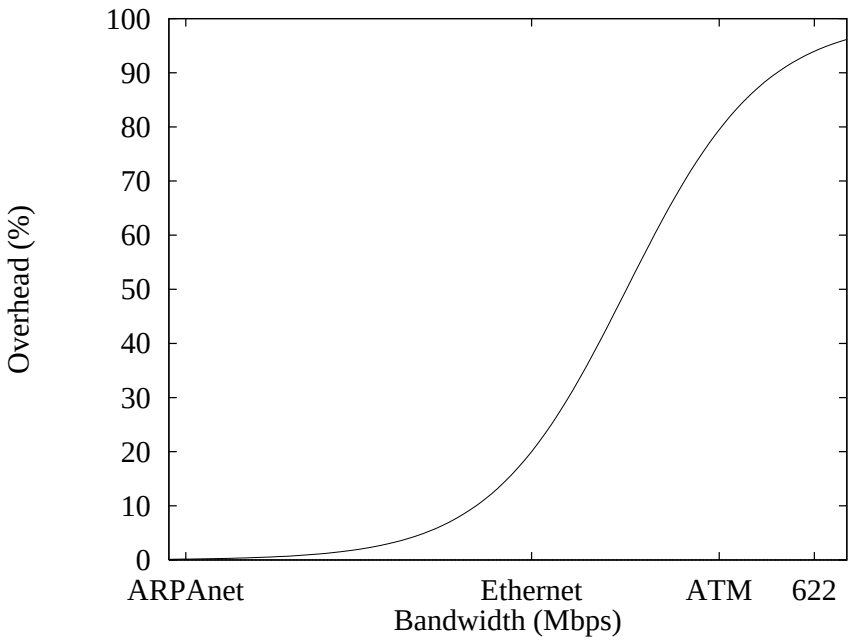
καλύτερο πρωτόκολλο.

Ενα άλλο χαρακτηριστικό του TCP αποτελεί η μεγάλη καθυστέρηση που επιβάλει σε μικρά μηνύματα, αφού κατά κανόνα περιμένει μέχρι να συγκεντρωθούν αρκετά δεδομένα πριν τα προωθήσει στα επόμενα στρώματα. Η συμπεριφορά αυτή όμως μπορεί να αλλάξει, αν ζητηθεί κάτι τέτοιο από το χρήστη. Με αυτόν τον τρόπο όλα τα μηνύματα προωθούνται άμεσα στα επόμενα στρώματα.

Επίσης το TCP δεν επιβεβαιώνει άμεσα τη λήψη των πακέτων από το δίκτυο, αλλά στέλνει τις επιβεβαιώσεις αυτές σαν τμήμα των πακέτων δεδομένων. Αυτό σημαίνει ότι αν ένας κόμβος Α στέλνει για μεγάλο χρονικό διάστημα στον κόμβο Β δεδομένα (π.χ. επειδή στέλνει σελίδες για αποθήκευση), ο κόμβος Β δε θα στείλει επιβεβαίωση αν δεν έχει και ο ίδιος δεδομένα να στείλει στον κόμβο Α, παρά μόνο αν έχει περάσει ένα συγκεκριμένο χρονικό όριο. Η συμπεριφορά αυτή μπορεί να οδηγήσει σε μεγάλη καθυστέρηση αν το χρονικό όριο αυτό είναι μεγαλύτερο από το χρόνο που χρειάζεται ο κόμβος Α για να γεμίσει τους ενταμιευτές της σύνδεσης με τον κόμβο Β. Σε τέτοια περίπτωση μπορεί να συμβεί η εξής κατάσταση. Ο κόμβος Α έχει στείλει τόσα δεδομένα στο Β όσα χρειάζονται για να γεμίσει ο ενταμιευτής της μεταξύ τους σύνδεσης. Στη συνέχεια δε στέλνει επιπλέον δεδομένα, αφού γνωρίζει ότι ο κόμβος Β δεν έχει χώρο να τα αποθηκεύσει, αλλά περιμένει για την επιβεβαίωση των δεδομένων που έχει ήδη στείλει. Ο κόμβος Β δε στέλνει επιβεβαίωση, αφού δεν έχει δεδομένα να στείλει στον Α, μέχρι να περάσει το χρονικό όριο. Ετσι από τη στιγμή που έχουν μεταδοθεί αρκετά δεδομένα ώστε να γεμίσει ο ενταμιευτής της σύνδεσης μέχρι το χρονικό όριο δε μεταφέρονται δεδομένα από τον Α στο Β, με αποτέλεσμα να καθυστερεί η μεταφορά επιπλέον σελίδων από τον Α στο Β. Ο μόνος τρόπος για να λυθεί αυτό το πρόβλημα, το οποίο είναι εγγενές στο TCP και είναι γνωστό σαν το πρόβλημα της καθυστερημένης επιβεβαίωσης, είναι η χρήση αρκετά μεγάλων ενταμιευτών, ώστε ο χρόνος που απαιτείται για να γεμίσουν ακόμη και με συνεχή ροή



Σχήμα 3.3: **Οι διαδοχικές αντιγραφές δεδομένων χρησιμοποιώντας το TCP/IP.**



Σχήμα 3.4: **Η επιβάρυνση του λογισμικού κατά τη μεταφορά δεδομένων μεγέθους 8KB με τη χρήση του TCP/IP.**

Συνάρτηση	Υλοποιείται από	Καλείται από
xxxopen()	οδηγό συσκευής	πυρήνα
xxxread()	οδηγό συσκευής	πυρήνα
xxxwrite()	οδηγό συσκευής	πυρήνα
physio()	πυρήνα	οδηγό συσκευής

Πίνακας 3.2: **Η διασύνδεση μιας συσκευής τυχαίας προσπέλασης σαν συσκευή χαρακτήρων με το λειτουργικό σύστημα.**

δεδομένων να είναι μεγαλύτερος από το χρονικό όριο στο οποίο ένας κόμβος στέλνει επιβεβαίωση για τα δεδομένα που έχει λάβει.

3.1.4 Η Διασύνδεση με το Δίσκο

Το ποσό της ελεύθερης μνήμης στο δίκτυο, όπως φαίνεται και στο σχήμα 1.5, είναι σημαντικό, όχι όμως αρκετό για να αποθηκεύσει όλες τις σελίδες όλων των μηχανών του δικτύου. Ο σελιδοδιαχειριστής θα πρέπει λοιπόν να έχει τη δυνατότητα να χρησιμοποιεί και τον τοπικό δίσκο για την αποθήκευση σελίδων σαν εναλλακτική λύση. Ο σελιδοδιαχειριστής υποστηρίζει την αποθήκευση σελίδων τόσο σε τμήματα του δίσκου, όσο και σε αρχεία. Η δεύτερη δυνατότητα μάλιστα δεν υποστηρίζεται από το DEC OSF/1, αλλά υλοποιήθηκε για λόγους πληρότητας και για να συγκριθεί η απόδοσή της με τη χρήση τμημάτων του δίσκου.

Για την αποθήκευση σελίδων σε τμήμα του δίσκου, ο σελιδοδιαχειριστής απλά δίνει την αίτηση που πρόκειται να εξυπηρετηθεί σαν όρισμα στην αντίστοιχη συνάρτηση *cdisk_strategy()* του δίσκου, η οποία φροντίζει να ξεκινήσει μια μεταφορά DMA και να ενημερώσει τον πυρήνα καλώντας την *biodone()* όταν αυτή ολοκληρωθεί. Αντιμετωπίζει δηλαδή το δίσκο σαν συσκευή μπλοκ, όπως ακριβώς θα έκανε και ο σελιδοδιαχειριστής του λειτουργικού συστήματος. Για την αποθήκευση σελίδων σε αρχείο, ο σελιδοδιαχειριστής αντιμετωπίζει το δίσκο σαν συσκευή χαρακτήρων, αφού τα μπλοκ στα οποία θα αποθηκευτούν ή από τα οποία θα ανακτηθούν οι ζητούμενες σελίδες δεν είναι απαραίτητα συνεχόμενα στο δίσκο. Ετσι καλεί τις αντίστοιχες συναρτήσεις που φαίνονται στον πίνακα 3.2 και συγκεκριμένα τις *cdisk_write()* και *cdisk_read()*.

Επειδή κατά την αντιμετώπιση του δίσκου σαν συσκευή χαρακτήρων ο σελιδο–διαχειριστής μπλοκάρει μέχρι τα δεδομένα να μεταφερθούν ανάμεσα στη μνήμη και το δίσκο, η χρήση αρχείου για την αποθήκευση σελίδων είναι αρκετά πιο αργή από τη χρήση τμήματος δίσκου. Δηλαδή ένα αρχείο θα πρέπει να χρησιμοποιείται για αποθήκευση σελίδων μόνο αν δεν υπάρχει διαθέσιμο τμήμα του δίσκου για τέτοια χρήση.

3.2 Η Υλοποίηση του Εξυπηρετητή Μνήμης

Ο εξυπηρετητής μνήμης, όπως αναφέρθηκε και στην ενότητα 2.4, δεν είναι παρά ένα πρόγραμμα που εκτελείται σε κατάσταση χρήστη. Δέχεται συνδέσεις μέσω πριζών TCP/IP από τα υπόλοιπα μηχανήματα του δικτύου και διαχειρίζεται ένα τμήμα της ελεύθερης μνήμης του μηχανήματος και εξυπηρετεί αιτήσεις για μεταφορά σελίδων. Επίσης περιοδικά ενημερώνει τον πελάτη του για το ποσό της ελεύθερης μνήμης.

Ο κώδικας του εξυπηρετητή είναι πλήρως μεταφέρσιμος και μπορεί να εκτελεστεί σε οποιοδήποτε περιβάλλον Unix. Αυτό συμβαίνει γιατί χρησιμοποιεί πρίζες εξασφαλίζοντας την ανεξαρτησία του από το δίκτυο και επειδή η μονάδα μεταφοράς δεδομένων δεν είναι η σελίδα αλλά το μπλοκ μεγέθους 0.5 KB, ίσου δηλαδή με το μέγεθος ενός τομέα δίσκου. Το γεγονός αυτό κάνει δυνατή τη λειτουργία του συστήματος και σε ετερογενή δίκτυα τα οποία μπορεί να αποτελούνται από μηχανήματα διαφορετικής αρχιτεκτονικής με διαφορετικά λειτουργικά συστήματα και διαφορετικό μέγεθος σελίδας.

Ο εξυπηρετητής εκτελείται σε κάθε μηχάνημα του δικτύου και αποδέχεται αιτήσεις σύνδεσης μόνο αν υπάρχει αρκετή ελεύθερη μνήμη στο μηχάνημά του. Με αυτόν τον τρόπο κάθε μηχάνημα μπορεί να παίζει το ρόλο πελάτη ή εξυπηρετητή ανάλογα με το ποσό της ελεύθερης φυσικής του μνήμης σε κάθε χρονική στιγμή. Η ασφάλεια του συστήματος εξασφαλίζεται με τη χρήση προνομιούχων πορτών για τη σύνδεση πελάτη−εξυπηρετητή και με την παραχώρηση άδειας προσπέλασης στη συσκευή μας μόνο στον υπερχρήστη του συστήματος.

3.3 Θέματα Πολιτικής

3.3.1 Επιλογή Εξυπηρετητών

Καθένα από τα μηχανήματα του συστήματος γνωρίζει όλα τα άλλα μηχανήματα που συμμετέχουν σ' αυτό. Τα μηχανήματα που προσφέρουν τμήμα της φυσικής τους μνήμης για την αποθήκευση σελίδων ονομάζονται εξυπηρετητές ενώ τα μηχανήματα που χρησιμοποιούν τη μνήμη άλλων για την αποθήκευση σελίδων τους ονομάζονται πελάτες. Κάθε μηχάνημα, ανάλογα με το φόρτο μνήμης του, μπορεί να λειτουργεί είτε σαν πελάτης είτε σαν εξυπηρετητής σε κάθε χρονική στιγμή.

Ο στόχος του εξυπηρετητή είναι διπλός. Καταρχήν θα πρέπει να αποθηκεύει σελίδες μνήμης με τρόπο ώστε το σύνολο των εξυπηρετητών να λειτουργεί σαν μια ολική κρυφή μνήμη για όλα τα τμήματα δίσκων που χρησιμοποιούνται για αποθήκευση σελίδων στο δίκτυο. Επίσης δε θα πρέπει η λειτουργία του να έχει σαν αποτέλεσμα την επιβάρυνση των τοπικών εφαρμογών. Προφανώς είναι απαράδεκτο το φαινόμενο να υποφέρει μια τοπική εφαρμογή από παλινδρόμηση ενώ τμήμα της μνήμης του σταθμού εργασίας της χρησιμοποιείται για την αποθήκευση σελίδων μνήμης άλλων μηχανών του δικτύου. Γι' αυτό το ποσό της μνήμης που χρησιμοποιείται από τον εξυπηρετητή αναπροσαρμόζεται περιοδικά με τρόπο τέτοιο ώστε να υπάρχει πάντα αρκετή ελεύθερη μνήμη για τις τοπικές εφαρμογές. Οταν οι τοπικές απαιτήσεις για μνήμη αυξάνονται, μερικές από τις λιγότερο πρόσφατα χρησιμοποιημένες σελίδες του εξυπηρετητή ελευθερώνονται και τα δεδομένα τους αποθηκεύονται στον τοπικό του δίσκο. Οταν οι τοπικές απαιτήσεις για μνήμη είναι μικρές, ο εξυπηρετητής δεσμεύει μερικές από τις ελεύθερες σελίδες.

Ο εξυπηρετητής υλοποιείται σαν ένα πολυνηματικό πρόγραμμα που εκτελείται σε κατάσταση χρήστη και δέχεται αιτήσεις μεταφοράς σελίδων από τους πελάτες του. Η μνήμη του είναι οργανωμένη σε δυο λίστες σελίδων (λίστα ελεύθερων και λίστα ενεργών σελίδων). Επίσης υπάρχει ένα ευρετήριο των σελίδων που έχουν αποθηκευτεί στον τοπικό δίσκο λόγω έλλειψης μνήμης. Το πρωτόκολλο επικοινωνίας ανάμεσα στον πελάτη και τον εξυπηρετητή αποτελείται από αιτήσεις τεσσάρων τύπων:

- Αιτήσεις μεταφοράς σελίδων από τον πελάτη στον εξυπηρετητή.

- Αιτήσεις μεταφοράς σελίδων από τον εξυπηρετητή στον πελάτη.

- Αιτήσεις από τον πελάτη για την ελευθέρωση σελίδων που πληροφορούν τον εξυπηρετητή ότι η δοσμένη σελίδα δε χρησιμοποιείται πια και μπορεί να ελευθερωθεί.

- Αιτήσεις πληροφόρησης του ελεύθερου αποθηκευτικού χώρου του εξυπηρετητή. Το ποσό αυτό της μνήμης είναι το άθροισμα των σελίδων που ανήκουν στη λίστα ελεύθερων σελίδων και των ενεργών σελίδων που δεν έχουν προσπελαστεί κατά τα τελευταία `THRES` δευτερόλεπτα.

Ο πελάτης διατηρεί μια λίστα από εξυπηρετητές με τους οποίους έχει συνδεθεί, ταξινομημένη με βάση το ποσό της ελεύθερης μνήμης τους. Με αυτό τον τρόπο επιλέγει το λιγότερο φορτωμένο εξυπηρετητή προκειμένου να μεταφέρει την επόμενη σελίδα (η οποία μπορεί να περιέχει είτε δεδομένα είτε πληροφορία ισοτιμίας όπως περιγράφεται στο κεφάλαιο 4). Περιοδικά ενημερώνεται για το ποσό της ελεύθερης μνήμης των εξυπηρετητών κάνοντας αιτήσεις πληροφόρησης του ελεύθερου αποθηκευτικού τους χώρου και στην περίπτωση που ο ελεύθερος χώρος κάποιου εξυπηρετητή πέσει στο μηδέν σταματά να του στέλνει δεδομένα. Με αυτό τον τρόπο επιτυγχάνεται εύκολα η εξισορρόπιση του φόρτου μνήμης στο δίκτυο αφού οι αποφάσεις επιλογής εξυπηρετητή παίρνονται σε επίπεδο σελίδας, η πληροφορία ισοτιμίας κατανέμεται σε όλους τους εξυπηρετητές και το σύνολο των εξυπηρετητών δρα σαν μια κρυφή μνήμη για τα τμήματα των δίσκων που χρησιμοποιούνται για την αποθήκευση σελίδων.

Η παράμετρος `THRES` σχετίζεται με το ρυθμό μεταγωγής δεδομένων του δικτύου. Αν η τιμή της γίνει μηδέν, τότε οι εξυπηρετητές συμπεριφέρονται σαν μια κρυφή μνήμη για τα τμήματα των δίσκων που χρησιμοποιεί πολιτική LRU. Αυτό το γεγονός είναι επιθυμητό για τοπικά δίκτυα μεγάλου ρυθμού μεταγωγής δεδομένων στα οποία το κόστος μεταφοράς μιας σελίδας πάνω από το δίκτυο είναι αμελητέο σε σχέση με το κόστος αποθήκευσης της σελίδας στο δίσκο. Ομως σε δίκτυα χαμηλού ρυθμού μεταγωγής δεδομένων (όπως το Ethernet), θέτοντας το `THRES` σε μια τιμή κοντά στο μηδέν μπορεί να προκαλέσει χειροτέρευση της απόδοσης, αφού πολλές ενεργές σελίδες στους εξυπηρετητές θα αντικατασταθούν από νέες σελίδες και θα αποθηκευτούν στο δίσκο του εξυπηρετητή. Αφού το κόστος μεταφοράς μιας σελίδας πάνω από τέτοια δίκτυα είναι συγκρίσιμο με το κόστος αποθήκευσης στο δίσκο, θα ήταν προτιμότερη η απευθείας αποθήκευση της σελίδας στον τοπικό δίσκο του πελάτη από τη μεταφορά της σε ένα εξυπηρετητή κατά την οποία μια άλλη ενεργή σελίδα θα μεταφερόταν στο δίσκο του.

Ο εξυπηρετητής θα μπορούσε να υλοποιηθεί μέσα στον πυρήνα προκειμένου να αποφευχθεί το επιπλέον κόστος αλλαγής χώρου διευθύνσεων κατά την αποστολή και λήψη μηνυμάτων. Προτιμήθηκε όμως η υλοποίηση σε κατάσταση χρήστη για λόγους ευελιξίας, ευκολίας στην ανάπτυξη και μεταφερσιμότητας. Αφού οι εξυπηρετητές εκτελούνται σε κατάσταση χρήστη θα πρέπει να αποκλειστεί το ενδεχόμενο σχηματισμού αλυσίδων από μηχανήματα όπου το καθένα θα αποθηκεύει σελίδες του προηγούμενού του στην αλυσίδα οδηγώντας σε άσκοπη χρήση του δικτύου και σε τεράστια καθυστέρηση. Αυτό επιτυγχάνεται κλειδώνοντας το χώρο διευθύνσεων του εξυπηρετητή στη μνήμη.

3.3.2 Συμπίεση

Οι απαιτήσεις σε ρυθμό μεταγωγής δεδομένων που έχει ένα σύστημα αποθήκευσης σελίδων σε απομακρυσμένους κόμβους θα μπορούσαν να μειωθούν με τη χρήση συμπίεσης. Γι' αυτό το λόγο δοκιμάστηκε η χρήση μερικών από τις πιο δημοφιλείς υλοποιήσεις αλγορίθμων συμπίεσης, όπως τα *gzip* [ZL77], *compress* [Wel84] και *compact* [Gal78]. Η χρήση όμως των αλγορίθμων αυτών όχι μόνο δεν ωφέλησε την απόδοση του συστήματος συμπιέζοντας τις μεταφερόμενες σελίδες αλλά αντίθετα επιδείνωσε δραματικά το μέσο χρόνο μεταφοράς σελίδων. Αυτό συμβαίνει για τους εξής λόγους

- *Οι αλγόριθμοι αυτοί είναι σχεδιασμένοι για τη συμπίεση μεγάλων αρχείων.* Ο βαθμός συμπίεσης που επιτυγχάνουν για δυαδικά δεδομένα μεγέθους μιας σελίδας είναι πολύ κοντά στη μονάδα, ενώ πολλές φορές τα συμπιεσμένα δεδομένα έχουν μεγαλύτερο μέγεθος από μια σελίδα, από το μέγεθος δηλαδή των ασυμπίεστων δεδομένων. Ετσι η μείωση στον απαιτούμενο ρυθμό μεταγωγής δεδομένων είναι ελάχιστη.
- *Ο χρόνος που απαιτείται για τη συμπίεση είναι αρκετά μεγάλος και μάλιστα πολλαπλάσιος από το χρόνο που απαιτείται για τη μετάδοση των δεδομένων μέσω του δικτύου.*

3.4 Πειραματικά Αποτελέσματα

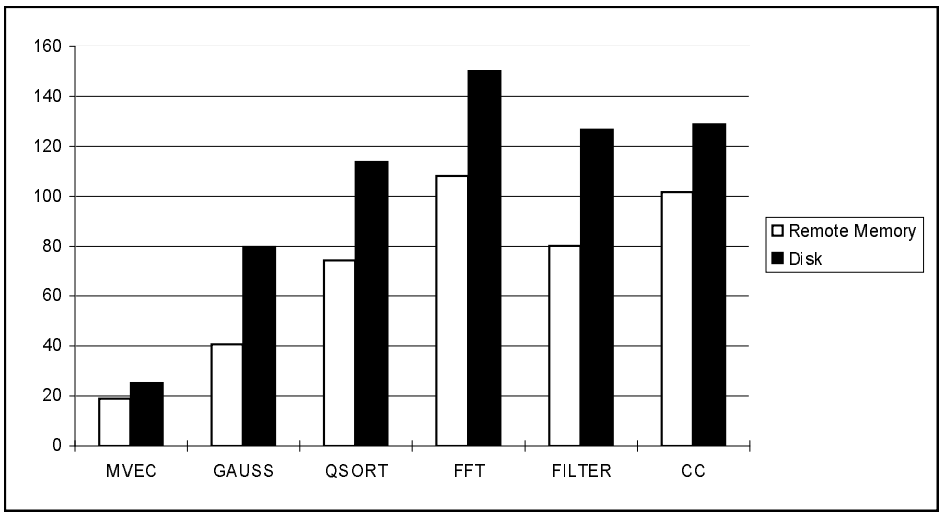
Στην ενότητα αυτή παρουσιάζονται πειραματικά αποτελέσματα από τη χρήση του συστήματός μας. Οι εφαρμογές που χρησιμοποιήθηκαν για την αξιολόγηση της απόδοσής του είναι οι εξής

- `MVEC`. Υλοποίηση πολλαπλασιασμού δισδιάστατου πίνακα με διάνυσμα.
- `GAUSS`. Απαλοιφή Gauss.
- `QSORT`. Ταξινόμηση πίνακα στοιχείων μεγέθους μιας σελίδας.
- `FFT`. Γρήγορος μετασχηματισμός Fourier και αντίστροφος μετασχηματισμός Fourier σε μια ακολουθία σημείων.
- `FILTER`. Εφαρμογή ενός διαχωρίσιμου φίλτρου τονισμού ακμών σε εικόνες.
- `CC`. Μεταγλώττιση του κώδικα που υλοποιεί τον οδηγό της συσκευής μας και σύνδεσή του με τα υπόλοιπα αρχεία για την παραγωγή του πυρήνα του λειτουργικού συστήματος.

Για τις μετρήσεις που εμφανίζονται στις επόμενες υποενότητες χρησιμοποιήθηκαν σταθμοί εργασίας DEC 3000 model 300 με 32 MB φυσικής μνήμης συνδεμένα με Ethernet. Το ένα από τα μηχανήματα είχε το ρόλο του πελάτη και τα υπόλοιπα το ρόλο των εξυπηρετητών.

3.4.1 Απόδοση πάνω από το Ethernet

Για την αξιολόγηση του συστήματός μας μετρήσαμε το χρόνο εκτέλεσης των προγραμμάτων που αναφέρονται στην προηγούμενη υποενότητα χρησιμοποιώντας σαν αποθηκευτικό χώρο σελίδων είτε τον τοπικό δίσκο είτε τη μνήμη δυο άλλων σταθμών



Σχήμα 3.5: **Η απόδοση εφαρμογών χρησιμοποιώντας είτε το δίσκο είτε την απομακρυσμένη μνήμη.** Για όλες τις εφαρμογές, η χρήση της απομακρυσμένης μνήμης δίνει πολύ καλύτερα αποτελέσματα από τη χρήση του δίσκου. Ολες οι εφαρμογές εκτελέστηκαν σε DEC Alpha 3000 model 300. Τα μεγέθη εισόδου ήταν 3000 στοιχεία για την `QSORT`, ένας πίνακας 1700 × 1700 για το `GAUSS`, ένας πίνακας 2100 × 2100 για τον `MVEC`, ένας πίνακας με 700 K στοιχεία για τον `FFT` και μια εικόνα 12 MB για το `FILTER`.

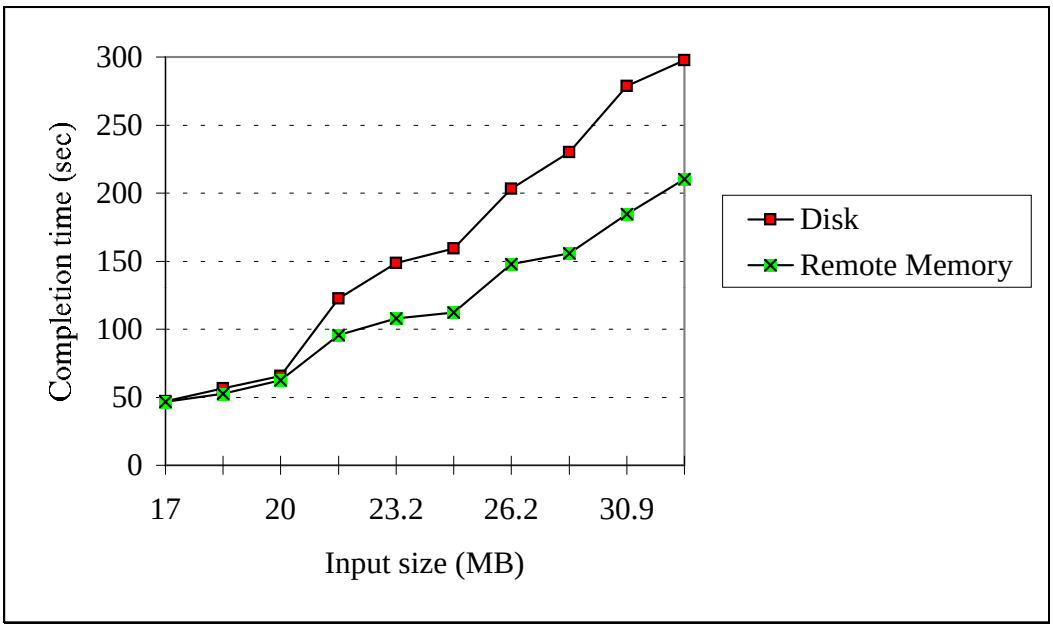
εργασίας. Στην πρώτη περίπτωση χρησιμοποιήθηκε ο δίσκος απευθείας από το λειτουργικό σύστημα χωρίς την ανάμιξη του δικού μας σελιδοδιαχειριστή. Στη δεύτερη οι μετρήσεις έγιναν πάνω από ένα σχεδόν ανενεργό Ethernet ώστε να εξασφαλιστεί η επαναληπτικότητα των αποτελεσμάτων.

Ο χρόνος εκτέλεσης των εφαρμογών χρησιμοποιώντας είτε τον τοπικό δίσκο είτε τη μνήμη των άλλων μηχανών του δικτύου φαίνεται στο σχήμα 3.5. Παρατηρούμε ότι σε όλες τις περιπτώσεις η χρήση της απομακρυσμένης μνήμης βελτιώνει σημαντικά την απόδοση. Για παράδειγμα, στην εφαρμογή `GAUSS` η απομακρυσμένη μνήμη δίνει 96% γρηγορότερο χρόνο εκτέλεσης από το δίσκο. Ακόμη και για την εφαρμογή `MVEC` που προκάλεσε μεταφορά λίγων μόνο σελίδων, η απομακρυσμένη μνήμη δίνει γρηγορότερο χρόνο εκτέλεσης κατά 32%. Ακόμη και σε εφαρμογές που χρησιμοποιούν πολύ το σύστημα αρχείων, όπως οι `FILTER` και `CC`, η βελτίωση που προκύπτει είναι σημαντική.

Τα παραπάνω αποτελέσματα φανερώνουν ότι η μεταφορά σελίδων πάνω από το δίκτυο είναι απλά ταχύτερη από τη μεταφορά σελίδων στο δίσκο. Παρόλο που ο ρυθμός μεταγωγής δεδομένων που παρέχουν τα δυο αυτά μέσα αποθήκευσης είναι παραπλήσιοι, το δίκτυο δεν έχει καθυστέρηση αναζήτησης και περιστροφής όπως ο δίσκος. Αφού τα σύγχρονα δίκτυα προσφέρουν όλο και μεγαλύτερο ρυθμό μεταγωγής δεδομένων σε σχέση με το δίσκο, η βελτίωση στην απόδοση δεν μπορεί παρά να αυξηθεί ακόμη περισσότερο στο μέλλον.

3.4.2 Απόδοση σε Σχέση με το Μέγεθος Εισόδου

Για να κατανοήσουμε τη συμπεριφορά των εφαρμογών σαν συνάρτηση του μεγέθους του συνόλου εργασίας τους, μετρήσαμε το χρόνο εκτέλεσης της εφαρμογής `FFT` σαν συνάρτηση του μεγέθους της εισόδου του, δηλαδή του αριθμού των σημείων της ακολουθίας, χρησιμοποιώντας είτε το δίσκο είτε την απομακρυσμένη μνήμη. Τα



Σχήμα 3.6: **Απόδοση της εφαρμογής** `FFT` **χρησιμοποιώντας είτε την απομακρυσμένη μνήμη είτε το δίσκο, σαν συνάρτηση του μεγέθους των δεδομένων εισόδου.**

αποτελέσματα φαίνονται στο σχήμα 3.6. Παρατηρούμε ότι μόλις το μέγεθος του συνόλου εργασίας υπερβεί τα 18 MB, αρχίζει η μεταφορά σελίδων ανάμεσα στη μνήμη και τη συσκευή αποθήκευσης σελίδων με αποτέλεσμα τη μεγάλη αύξηση του χρόνου εκτέλεσης. Οι περισσότεροι χρήστες σίγουρα δε θα ήθελαν να ανεχθούν μια τόσο μεγάλη επιβάρυνση στο χρόνο εκτέλεσης για προγράμματα που δε χωρούν στην κύρια μνήμη. Ευτυχώς η χρήση της απομακρυσμένης μνήμης μειώνει την επιβάρυνση αυτή σημαντικά.

3.4.3 Απόδοση πάνω από Δίκτυα Μεγαλύτερου Ρυθμού Μεταγωγής Δεδομένων

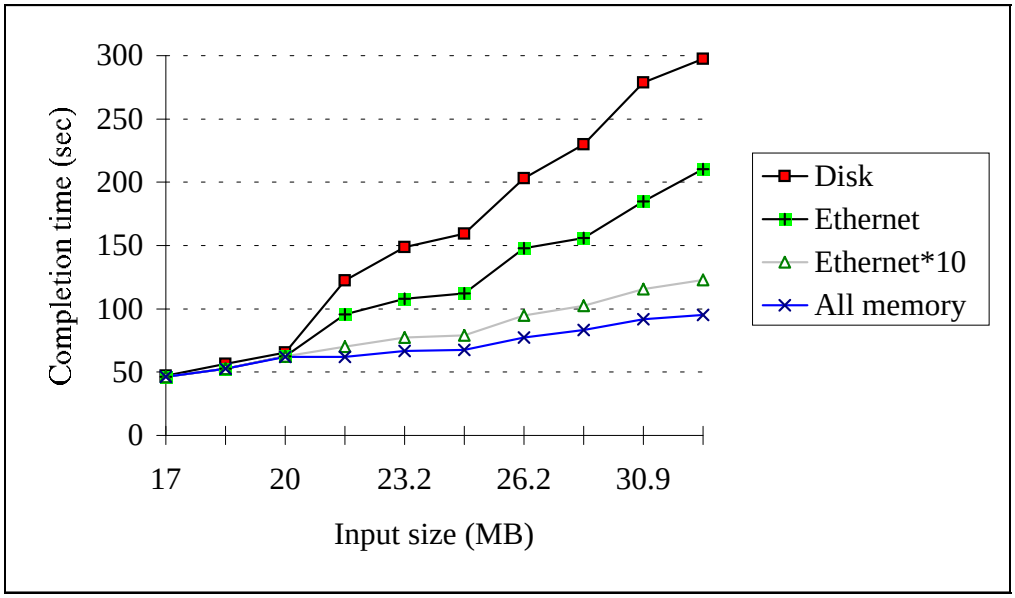
Παρόλο που, σύμφωνα με το σχήμα 3.6, η απόδοση της απομακρυσμένης μνήμης σε σχέση με το δίσκο είναι σημαντικά καλύτερη, ο χρόνος εκτέλεσης εφαρμογών που δε χωρούν στην κύρια μνήμη, ακόμη κι αν χρησιμοποιείται η απομακρυσμένη μνήμη, εξακολουθεί να είναι πολύ μεγάλος. Βέβαια αυτό θα πάψει να ισχύει μόλις το Ethernet αντικατασταθεί από ένα δίκτυο που παρέχει μεγαλύτερο ρυθμό μεταγωγής δεδομένων (π.χ. FDDI [Ame87], ATM [New94]).

Για να εκτιμήσουμε την απόδοση των εφαρμογών μας πάνω από τέτοια δίκτυα, έγιναν λεπτομερείς μετρήσεις οι οποίες μας επιτρέπουν το διαχωρισμό του χρόνου εκτέλεσης στους παρακάτω παράγοντες

1. *Χρόνος χρήστη* ($utime$). Ο χρόνος αυτός είναι ο χρόνος κατά τον οποίο η εφαρμογή εκτελείται σε κατάσταση χρήστη.

2. *Χρόνος συστήματος* ($systime$). Ο χρόνος αυτός είναι ο χρόνος κατά τον οποίο η εφαρμογή εκτελείται σε κατάσταση πυρήνα.

3. *Χρόνος αρχικοποίησης* ($inittime$). Ο χρόνος αυτός είναι ο χρόνος κατά τον οποίο η εφαρμογή αρχικοποιείται από το λειτουργικό σύστημα.



Σχήμα 3.7: **Απόδοση του `FFT` για διάφορες επιλογές τοποθέτησης σελίδων.** Παρουσιάζονται οι επιδόσεις της εφαρμογής πάνω από τον τοπικό δίσκο και το Ethernet. Επίσης παρουσιάζονται οι εκτιμούμενες επιδόσεις της εφαρμογής σε ένα δίκτυο με 10 φορές μεγαλύτερο ρυθμό μεταγωγής δεδομένων από το Ethernet και σε ένα σύστημα με αρκετή φυσική μνήμη ώστε να χωρά όλα τα δεδομένα της εφαρμογής.

4. *Χρόνος μεταφοράς σελίδων* ($ptime$). Ο χρόνος αυτός είναι ο χρόνος τον οποίο η εφαρμογή ξοδεύει κατά τη μεταφορά σελίδων.

Χρησιμοποιώντας την εντολή *time*, μετράμε τα $utime, systime$ και τον ολικό χρόνο ($etime$) για κάθε εφαρμογή. Αφαιρώντας τα $utime$ και $systime$ από το $etime$ για μεγέθη εισόδου που δεν προκαλούν μεταφορά σελίδων υπολογίζουμε το $inittime$. Ο χρόνος $ptime$ αποτελείται από το χρόνο που οφείλεται στο λογισμικό για αντιγραφή δεδομένων και στην επεξεργασία από τα πρωτόκολλα ($pptime$) και από το χρόνο κατά τον οποίο η εφαρμογή μπλοκάρει περιμένοντας τη μεταφορά δεδομένων πάνω από το δίκτυο και ο οποίος εξαρτάται από το ρυθμό μεταγωγής δεδομένων του δικτύου ($btime$). Μετρήσαμε το $pptime$ και το υπολογίσαμε ίσο με 1.6 ms ανά σελίδα για το TCP/IP. Υπολογίζουμε το $btime$ χρησιμοποιώντας τον τύπο $btime = (etime - utime - systime - inittime - no_of_page_transfers * pptime)$. Υποθέτοντας ότι ένα δίκτυο με X φορές μεγαλύτερο ρυθμό μεταγωγής δεδομένων θα μειώσει το $btime$ κατά X φορές, μπορούμε να προβλέψουμε το χρόνο εκτέλεσης ($etime$) πάνω από αυτό το δίκτυο. Δηλαδή ο τύπος που χρησιμοποιείται γι' αυτή την εκτίμηση είναι ο εξής $Expected_elapsed_time = utime + systime + inittime + no_of_page_transfers * pptime + btime/X$.

Κάναμε όλους αυτούς τους υπολογισμούς για την εφαρμογή `FFT` και προβλέψαμε το χρόνο εκτέλεσής της σε ένα δίκτυο με 10 φορές μεγαλύτερο ρυθμό μεταγωγής δεδομένων από το Ethernet. Επίσης προβλέψαμε το χρόνο εκτέλεσης της εφαρμογής σε ένα μηχάνημα με αρκετή φυσική μνήμη ώστε να χωρά ολόκληρο το σύνολο εργασίας της εφαρμογής, προσθέτοντας τους χρόνους $utime, systime$ και $inittime$. Οι υπολογιζόμενοι χρόνοι εκτέλεσης, μαζί με τους χρόνους που μετρήθηκαν με τη χρήση του δίσκου και της απομακρυσμένης μνήμης πάνω από το Ethernet παρουσιάζονται στο σχήμα 3.7. Παρατηρούμε ότι το δίκτυο δεκαπλάσιου ρυθμού μεταγωγής δεδομένων δίνει παραπλήσια αποτελέσματα με τη χρήση αρκετής κύριας μνήμης ώστε να μη γίνεται μεταφορά σελίδων και πολύ καλύτερα και από το δίσκο και από το Ethernet.

Για την καλύτερη κατανόηση των αποτελεσμάτων που παρουσιάζονται στο σχήμα

3.7, αναλύουμε το χρόνο εκτέλεσης της εφαρμογής `FFT` για μέγεθος εισόδου ίσο με 30.9 MB. Ο συνολικός χρόνος εκτέλεσης μετρήθηκε ίσος με 184.77 sec, αποτελούμενος από 86.045 sec χρόνου χρήστη, 5.53 sec χρόνου συστήματος, 0.21 sec χρόνου αρχικοποίησης και 92.985 sec χρόνου μεταφοράς σελίδων. Κατά τη διάρκεια της εκτέλεσης αυτής έγιναν 5905 εγγραφές και 4247 αναγνώσεις σελίδων. Αρα, ο χρόνος $pptime$ ήταν ίσος με $10152 * 0.0016$, ή περίπου 16.2432 sec. Ο χρόνος $btime$ λοιπόν είναι ίσος με $92.985 - 16.2432$, δηλαδή 76.7418 sec. Χρησιμοποιώντας ένα δίκτυο δεκαπλάσιου ρυθμού μεταγωγής δεδομένων ο χρόνος αυτός θα μειωθεί στα 7.67418 sec. Ετσι, ο ολικός χρόνος εκτέλεσης της εφαρμογής θα είναι $86.045 + 5.53 + 0.21 + 16.2432 + 7.67418$ sec ή 115.70238 sec, μοιρασμένος ως εξής 74.368% χρόνος χρήστη, 4.78% χρόνος συστήματος, 0.182% χρόνος αρχικοποίησης και 20.67% χρόνος μεταφοράς σελίδων. Παρατηρούμε λοιπόν ότι σε ένα δίκτυο με ρυθμό μεταγωγής δεδομένων ίσο με 100 Mbps ο χρόνος μεταφοράς σελίδων μειώνεται στο 20.67% του χρόνου εκτέλεσης της εφαρμογής πάνω από το δίκτυο αυτό. Πιστεύουμε ότι οι περισσότεροι χρήστες μπορούν να ανεχθούν μια επιβάρυνση της τάξης αυτής προκειμένου να μπορούν να εκτελούν προγράμματα που το σύνολο εργασίας τους δε χωρά στην κύρια μνήμη. Εξάλλου η μόνη άλλη επιλογή που έχουν είναι η χρήση του τοπικού δίσκου.

3.4.4 Η Καθυστέρηση ανά Σελίδα

Οπως αναφέρθηκε στην υποενότητα 3.4.3, ο χρόνος μεταφοράς σελίδων για την εφαρμογή `FFT` ήταν ίσος με 45.666 sec ή 9.2 msec για κάθε μεταφορά σελίδας. Προηγούμενες μετρήσεις αναφέρουν ότι χρειάζονται περίπου 45 ms για τη μεταφορά μιας σελίδας μεγέθους 4 KB πάνω από το Ethernet [SD91]. Από αυτά τα 19 ms οφείλονται στο TCP, 4ms στο μηχανισμό επικοινωνίας διεργασιών του Mach, 7.2 ms στο Ethernet και τα υπόλοιπα στο δίαυλο εισόδου/εξόδου του συστήματος. Στη δική μας υλοποίηση η ολική επιβάρυνση του λογισμικού και των πρωτοκόλλων για μια σελίδα μεγέθους 8 KB είναι μόλις 1.6 ms. Η τεράστια αυτή διαφορά οφείλεται στους εξής λόγους

- *Ο δίαυλος εισόδου/εξόδου στα DEC 3000 model 300 είναι πολύ ταχύτερος και δεν προκαλεί προβλήματα στην απόδοση.*

- *Χρησιμοποιούμε μηχανήματα βασισμένα στον επεξεργαστή Alpha, που είναι πάνω από 4 φορές γρηγορότερος από τον i386[Int86], ο οποίος χρησιμοποιήθηκε στο [SD91].*

- *Ο δικός μας σελιδοδιαχειριστής υλοποιήθηκε σαν οδηγός συσκευής, ενώ στο [SD91] υλοποιήθηκε σαν εξωτερικός διαχειριστής μνήμης πάνω από το Mach.* Παρόλο που οι εξωτερικοί διαχειριστές μνήμης προσφέρουν μεγαλύτερη ευελιξία, το επιπλέον κόστος που εισάγουν είναι τεράστιο, γι' αυτό και δε χρησιμοποιούνται στα περισσότερα εμπορικά λειτουργικά συστήματα, ακόμη κι αν αυτά έχουν τη δυνατότητα να τους υποστηρίξουν.

Γενικά ο σελιδοδιαχειριστής μας, αν και δεν έχει την ευελιξία ενός εξωτερικού διαχειριστή μνήμης, έχει πολύ καλύτερη απόδοση. Επιπλέον, η υλοποίηση σαν οδηγό συσκευής παρέχει καλύτερη απόδοση από τον τοπικό δίσκο, κάτι που δεν μπορούν να επιτύχουν υλοποιήσεις σε επίπεδο χρήστη.

3.4.5 Η Χρήση Φορτωμένων Εξυπηρετητών

Σε όλα τα πειράματά μας μέχρι τώρα, οι εξυπηρετητές μνήμης εκτελούνταν σε μηχανές που δεν εκτελούσαν καμία άλλη εφαρμογή χρήστη, προκειμένου τα αποτελέσματα να είναι επαναλήψιμα. Η κατάσταση αυτή όμως δεν είναι πάντα ρεαλιστική, αφού οι κόμβοι που διαθέτουν τη μνήμη τους για αποθήκευση σελίδων μπορεί να μην είναι εντελώς ανενεργοί. Θα ήταν σκόπιμο λοιπόν να μετρήσουμε τη μεταβολή της απόδοσης όταν χρησιμοποιούνται ενεργοί εξυπηρετητές. Γι' αυτό το λόγο κάναμε το εξής πείραμα

> Σε κάθε εξυπηρετητή ξεκινήσαμε το περιβάλλον X και ένα κειμενογράφο που χρησιμοποιούνταν για συγγραφή κειμένου. Μετά εκτελέσαμε ξανά τις εφαρμογές μας χρησιμοποιώντας την ίδια είσοδο και τους ίδιους εξυπηρετητές. Η μόνη διαφορά ήταν ότι οι εξυπηρετητές τώρα δεν ήταν ανενεργοί πλέον.[1]

Με έκπληξη διαπιστώσαμε ότι ο χρόνος εκτέλεσης για τις περισσότερες εφαρμογές διέφερε λιγότερο από 1 sec από το χρόνο εκτέλεσης με ανενεργούς εξυπηρετητές. Μόνο η `QSORT` παρουσίασε μια αύξηση του χρόνου εκτέλεσής της κατά 7%. Πιθανότατα ο εξυπηρετητής μετέφερε κάποιες από τις αποθηκευμένες σελίδες στον τοπικό του δίσκο.

Για να βρούμε το πόσο επηρεάζει ο φόρτος του επεξεργαστή ενός εξυπηρετητή το χρόνο εκτέλεσης των εφαρμογών μας, τρέξαμε σε έναν από τους εξυπηρετητές ένα πρόγραμμα που εκτελούσε την εντολή "while(1);". Ακόμη και τότε, ο χρόνος εκτέλεσης των εφαρμογών μας δεν αυξήθηκε πάνω από 7% σε σχέση με το χρόνο της εκτέλεσής τους χρησιμοποιώντας ανενεργούς εξυπηρετητές. Τα αποτελέσματα αυτά δείχνουν ότι τις περισσότερες φορές οι εξυπηρετητές ήταν σε θέση να ικανοποιήσουν άμεσα τις αιτήσεις του πελάτη, ακόμη κι αν ήταν φορτωμένοι.

Επίσης θα θέλαμε να μετρήσουμε το ποσοστό του χρόνου κατά τον οποίο ο εξυπηρετητής μνήμης χρησιμοποιεί τον επεξεργαστή του μηχανήματός του, καθυστερώντας έτσι τις διεργασίες που εκτελούνται σ' αυτό. Μετρήσαμε λοιπόν τη χρήση του επεξεργαστή από τον εξυπηρετητή μνήμης και βρήκαμε ότι ήταν πάντα μικρότερη του 15%. Αρα η επιβάρυνση του επεξεργαστή του εξυπηρετητή είναι τόσο μικρή ώστε ο ιδιοκτήτης του μηχανήματος δε θα αντιμετωπίζει προβλήματα αν το μηχάνημά του χρησιμοποιείται για αποθήκευση σελίδων.

3.4.6 Η Απόδοση πάνω από Φορτωμένο Ethernet

Ολες οι μετρήσεις που παρουσιάστηκαν μέχρι τώρα έγιναν πάνω από ένα σχεδόν ανενεργό Ethernet. Θα ήταν ενδιαφέρον λοιπόν να δούμε πώς επηρεάζεται η απόδοση του συστήματος σαν συνάρτηση του φόρτου του δικτύου, δηλαδή του ρυθμού μεταγωγής δεδομένων που χρησιμοποιείται για μεταφορά σελίδων. Ετσι επαναλάβαμε τις μετρήσεις πάνω από ένα ήδη φορτωμένο Ethernet. Τα αποτελέσματα έδειξαν μια ραγδαία πτώση της απόδοσης του συστήματος. Το γεγονός αυτό είναι αναμενόμενο αφού η μεταφορά σελίδων απαιτεί όσο το δυνατό μεγαλύτερο ρυθμό μεταγωγής δεδομένων υπερφορτώνοντας το δίκτυο. Η πρόσθεση περισσότερων πηγών κυκλοφορίας

[1] Θα μπορούσε κάποιος να υποστηρίξει ότι το περιβάλλον X και ένας κειμενογράφος δεν αποτελούν σημαντικό φόρτο για ένα μηχάνημα. Αλλά αυτό ακριβώς είναι το θέμα. Ενας τυπικός σταθμός εργασίας, ακόμη κι όταν χρησιμοποιείται, είναι ελαφρά φορτωμένος, αφού τον περισσότερο καιρό οι χρήστες εκτελούν προγράμματα που δεν έχουν μεγάλες υπολογιστικές απαιτήσεις. Τα λίγα μηχανήματα που είναι πραγματικά φορτωμένα σε μια δεδομένη χρονική στιγμή συνήθως δεν παραχωρούν τη μνήμη τους για αποθήκευση σελίδων.

οδηγεί σε τεράστια αύξηση της ζήτησης για μεταφορά, η οποία δημιουργεί συνεχείς συγκρούσεις και οδηγεί σε κατάρρευση του ρυθμού μεταγωγής δεδομένων.

Ευτυχώς, το φαινόμενο αυτό δεν οφείλεται εγγενώς στη χρήση του δικτύου για τη μεταφορά σελίδων, αλλά στο πρωτόκολλο CSMA/CD που χρησιμοποιεί το Ethernet [Tan89]. Αυτό σημαίνει ότι εξακολουθεί να είναι σημαντική η χρήση της απομακρυσμένης μνήμης για αποθήκευση σελίδων πάνω από δίκτυα άλλων τεχνολογιών (π.χ. token ring [IEE85b]), τα οποία είναι σε θέση να προσφέρουν ένα ρυθμό μεταγωγής δεδομένων 10 ή περισσοτέρων Mbps για το σκοπό αυτό.

Κεφάλαιο 4

Αξιοπιστία

4.1 Εισαγωγή

Σε ένα κατανεμημένο σύστημα, όπως είναι ένα τοπικό δίκτυο, ένα μηχάνημα μπορεί να κρεμάσει ανά πάσα στιγμή. Αν το μηχάνημα αυτό λειτουργεί σαν εξυπηρετητής, αυτό σημαίνει ότι θα χαθούν όλες οι αποθηκευμένες σελίδες σ' αυτό. Είναι προφανές ότι οι εφαρμογές που εκτελούνται σε μια μηχανή του δικτύου δεν είναι δυνατό να αποτυγχάνουν εξαιτίας της βλάβης κάποιου άλλου μηχανήματος που τυχαίνει να αποθηκεύει κάποιες από τις σελίδες τους. Επιπλέον, η αποτυχία προγραμμάτων όπως ο *init* ή οι δαίμονες του συστήματος θα προκαλέσει κρέμασμα και των μηχανών–πελατών του εξυπηρετητή που παρουσίασε βλάβη. Αυτό σημαίνει ότι το σύστημά μας θα πρέπει να είναι σε θέση να εγγυηθεί ότι η πιθανότητα να χαθεί κάποια σελίδα ενός μηχανήματος θα είναι μικρότερη ή ίση της πιθανότητας που έχει το ίδιο το μηχάνημα να υποστεί βλάβη.

Υπάρχουν πολλά είδη βλαβών. Καταρχήν, μπορεί να υπάρξει βλάβη λόγω πτώσης της τάσης του ηλεκτρικού ρεύματος. Μια τέτοια βλάβη δεν αντιμετωπίζεται από την εργασία αυτή, αφού τα περισσότερα κτίρια που στεγάζουν υπολογιστές διαθέτουν UPS. Μια άλλη αιτία βλάβης είναι η βλάβη του δικτύου (π.χ. διάσπαση του τοπικού δικτύου σε τμήματα που δεν μπορούν να επικοινωνήσουν λόγω βλάβης μιας γέφυρας). Σ' αυτή την περίπτωση, ο πελάτης γενικά δεν μπορεί να ανακτήσει τις αποθηκευμένες σελίδες του, αφού μπορεί να έχει αποκοπεί από όλα τα υπόλοιπα μηχανήματα. Η λύση λοιπόν που ακολουθείται είναι ότι ο πελάτης μπλοκάρει μέχρι να επανέλθει το δίκτυο. Αυτή τη λύση άλλωστε υιοθετούν και τα περισσότερα κατανεμημένα συστήματα, όπως το NFS [SGK+85]. Η πιο πιθανή αιτία βλάβης είναι η βλάβη ενός εξυπηρετητή, η οποία οφείλεται είτε σε πρόβλημα του υλικού είτε σε πρόβλημα του λογισμικού και έχει σαν αποτέλεσμα την απώλεια των περιεχομένων της μνήμης. Μια τέτοια περίπτωση λοιπόν θα πρέπει να αντιμετωπίζεται από το σύστημά μας.

Ο μέσος χρόνος μεταξύ βλαβών του συστήματός μας είναι περίπου ίσο με $\frac{MTTFw}{N}$, όπου N ο αριθμός των μηχανών μας και $MTTFw$ ο μέσος χρόνος μεταξύ βλαβών ενός μηχανήματος, θεωρώντας ότι οι βλάβες είναι ανεξάρτητες και ακολουθούν εκθετική κατανομή. Ο μέσος χρόνος αυτός είναι πολύ μικρός γεγονός που σημαίνει ότι θα πρέπει να αντιμετωπιστεί. Μια βλάβη μπορεί να αντιμετωπιστεί αν αποθηκεύουμε πλεονάζουσα πληροφορία ώστε να μπορούμε να ανακατασκευάσουμε τις σελίδες που έχουν αποθηκευτεί στο μηχάνημα που έπαθε τη βλάβη. Εστω ότι η πλεονάζουσα πληροφορία που διατηρούμε μας επιτρέπει την ανακατασκευή των σελίδων που είναι αποθηκευμένες σε ένα υπολογιστή αν διαθέτουμε τις σελίδες που είναι αποθηκευμέ–

νες σε άλλους υπολογιστές του δικτύου, στις οποίες συμπεριλαμβάνονται και σελίδες πλεονάζουσας πληροφορίας. Εστω $MTTRw$ ο μέσος χρόνος επισκευής ενός μηχανήματος. Αν G είναι ο αριθμός των μηχανών από τις οποίες αν κάποια πάθει βλάβη μπορούμε να ανακατασκευάσουμε τις σελίδες της χρησιμοποιώντας την πληροφορία που είναι αποθηκευμένη στις υπόλοιπες $G-1$ μηχανές, τότε ο μέσος χρόνος μεταξύ βλαβών του συστήματός μας, όπως αποδεικνύεται στο [CLG+94] είναι ίσος με $\frac{MTTFw^2}{(N(G-1)MTTRw)}$. Για τυπικές τιμές των παραμέτρων, η τιμή της παράστασης αυτής είναι πολύ μεγάλη, π.χ. για $MTTFw$ ίσο με τρεις μήνες, $MTTRw$ δέκα λεπτά, N ίσο με 20 και G ίσο με 4, η τιμή της παράστασης είναι ίση περίπου με 54 χρόνια. Για τις ίδιες παραμέτρους με G ίσο με 16, η τιμή της είναι περίπου ίση με 10 χρόνια. Αρα η ικανότητα του συστήματος να αναρρώνει από βλάβες ενός μηχανήματος προσφέρει ικανοποιητική αξιοπιστία μια που το ενδεχόμενο ταυτόχρονης βλάβης περισσοτέρων του ενός μηχανήματος είναι σπάνιο.

4.2 Στόχοι των Πολιτικών Αξιοπιστίας

Για την αντιμετώπιση λοιπόν των βλαβών υλοποιήθηκαν και εφαρμόστηκαν διάφορες τεχνικές οι οποίες είναι σε θέση να ανακατασκευάσουν τις σελίδες που χάθηκαν. Οι στόχοι που θα πρέπει να εκπληρώνει μια τέτοια τεχνική είναι οι παρακάτω.

- *Η επιβάρυνση του χρόνου εκτέλεσης θα πρέπει να είναι όσο το δυνατό μικρότερη, αφού η επιβάρυνση αυτή αποτελεί ένα κόστος το οποίο το πληρώνουμε σε κάθε μεταφορά σελίδας ανεξάρτητα αν κάποιο μηχάνημα έχει υποστεί βλάβη ή όχι.*
- *Η επιπλέον μνήμη που απαιτεί για να λειτουργήσει θα πρέπει επίσης να είναι όσο το δυνατό μικρότερη, αφού η μνήμη που είναι δεσμευμένη για την εξασφάλιση αξιοπιστίας θα μπορούσε να χρησιμοποιηθεί για την αποθήκευση σελίδων άλλων μηχανών του δικτύου.*
- *Ο χρόνος που απαιτείται για την ανακατασκευή των σελίδων σε περίπτωση βλάβης θα πρέπει να είναι μικρός.* Αυτό το κριτήριο δεν είναι τοσο σημαντικό όσο τα δυο προηγούμενα, αφού οι βλάβες συμβαίνουν σχετικά σπάνια, άρα υπάρχει η δυνατότητα διάθεσης λίγων επιπλέον δευτερολέπτων για την ανακατασκευή των σελίδων που χάθηκαν.

Στην ενότητα 4.3 περιγράφουμε τρεις διαφορετικές πολιτικές αξιοπιστίας και ελέγχουμε το κατά πόσο ικανοποιούν τα κριτήρια αυτά.

4.3 Συγκριτική Μελέτη Πολιτικών Αξιοπιστίας

4.3.1 Η Τεχνική των Πολλαπλών Αντιγράφων

Η απλούστερη τεχνική αξιοπιστίας είναι η *τεχνική διπλών (ή στη γενική περίπτωση πολλαπλών) αντιγράφων*. Για κάθε σελίδα δηλαδή που βρίσκεται αποθηκευμένη στην απομακρυσμένη μνήμη, υπάρχουν δυο αντίγραφα, αποθηκευμένα σε διαφορετικούς κόμβους του δικτύου. Ετσι, αν κάποιος κόμβος υποστεί βλάβη, όλες οι σελίδες που αποθήκευε μπορούν να ανακτηθούν από τους υπόλοιπους κόμβους στους οποίους είναι αποθηκευμένες.

Η μέθοδος αυτή δεν επιφέρει επιπλέον κόστος στην ανάγνωση σελίδων και η ανακατασκευή των κατεστραμένων σελίδων δε χρειάζεται, αφού αυτές υπάρχουν ήδη

αποθηκευμένες και σε άλλους κόμβους. Κατά την εγγραφή όμως, πρέπει να γίνουν δυο μεταφορές της σελίδας, μια σε καθένα από τους κόμβους όπου θα αποθηκευτεί. Ετσι, η επιβάρυνση στο χρόνο εκτέλεσης είναι τεράστια, αφού ο χρόνος εγγραφής σελίδων διπλασιάζεται. Επιπλέον, το κόστος σε μνήμη είναι επίσης απαγορευτικό, αφού η μισή από τη χρησιμοποιούμενη μνήμη ξοδεύεται για την εξασφάλιση της αξιοπιστίας.

4.3.2 Η Τεχνική της Ισοτιμίας

Για να μειωθεί το ποσό της μνήμης που απαιτείται για αξιοπιστία, μπορεί να χρησιμοποιηθεί η τεχνική της *ισοτιμίας*, την οποία δανειστήκαμε από τα RAIDs [CLG+94]. Ας υποθέσουμε, για παράδειγμα, ότι έχουμε S εξυπηρετητές, καθένας από τους οποίους αποθηκεύει P σελίδες. Εστω $P(i, j)$ η $j-$στή σελίδα που είναι αποθηκευμένη στον εξυπηρετητή i. Υποθέτουμε ότι έχουμε P σελίδες ισοτιμίας, όπου η $j-$στή σελίδα σχηματίζεται σαν το αποκλειστικό ή από τις $j-$στές σελίδες όλων των εξυπηρετητών. Λέμε ότι όλες αυτές οι $j-$στές σελίδες ανήκουν στο ίδιο σύνολο ισοτιμίας. Αν ένας εξυπηρετητής υποστεί βλάβη, κάθε σελίδα του μπορεί να ανασκευαστεί υπολογίζοντας το αποκλειστικό ή όλων των υπολοίπων σελίδων του συνόλου ισοτιμίας της. Η τεχνική αυτή αποτελεί υπερσύνολο της τεχνικής διπλών αντιγράφων, αφού ταυτίζεται με αυτή για $S = 1$.

Οταν ο πελάτης μεταφέρει μια σελίδα για εγγραφή, πρέπει να ανανεώσει την αντίστοιχη σελίδα ισοτιμίας. Αυτή η ανανέωση γίνεται σε δυο στάδια.

1. *Ο πελάτης στέλνει τη σελίδα στον κατάλληλο εξυπηρετητή, ο οποίος υπολογίζει το ''αποκλειστικό ή'' των παλιών και των νέων περιεχομένων της σελίδας.*
2. *Ο εξυπηρετητής στέλνει το αποτέλεσμα του υπολογισμού αυτού στον εξυπηρετητή ισοτιμίας, ο οποίος υπολογίζει το ''αποκλειστικό ή'' των δεδομένων που του στέλνονται με την παλιά σελίδα ισοτιμίας.* Το αποτέλεσμα είναι η νέα σελίδα ισοτιμίας.

Η μέθοδος αυτή αυξάνει ελάχιστα την απαιτούμενη φυσική μνήμη (κατά ένα παράγοντα $1/S$). Ομως η επιβάρυνση στο χρόνο εκτέλεσης εξακολουθεί να είναι τεράστια, αφού κάθε εγγραφή σελίδας απαιτεί δυο μεταφορές σελίδων, μια από τον πελάτη στον εξυπηρετητή μνήμης και μια από τον εξυπηρετητή μνήμης στον εξυπηρετητή ισοτιμίας. Ο πελάτης δεν μπορεί να θεωρήσει ότι η εγγραφή ολοκληρώθηκε, και άρα να ειδοποιήσει τον πυρήνα, παρά μόνο όταν και η σελίδα ισοτιμίας έχει ανανεωθεί. Σε αντίθετη περίπτωση, αν δηλαδή θεωρήσει ότι η εγγραφή ολοκληρώθηκε στέλνοντας τα δεδομένα στον εξυπηρετητή μνήμης, μια ενδεχόμενη βλάβη του θα είχε σαν αποτέλεσμα την καταστροφή των δεδομένων αυτών. Αρα η επιβάρυνση στο χρόνο εκτέλεσης εξακολουθεί να είναι η ίδια όπως και στην τεχνική των διπλών αντιγράφων. Επιπλέον απαιτείται και υλοποίηση νέας εφαρμογής, αυτής του εξυπηρετητή ισοτιμίας.

4.3.3 Η Τεχνική της Μεταβλητής Ισοτιμίας

Οπως είδαμε στην υποενότητα 4.3.2, η τεχνική της ισοτιμίας απαιτεί ελάχιστη επιπλέον φυσική μνήμη, εξακολουθεί όμως να εισάγει τεράστια επιβάρυνση στο χρόνο εκτέλεσης. Για να απαλλαγούμε και από την επιβάρυνση στο χρόνο εκτέλεσης που εισάγουν οι προηγούμενες τεχνικές αξιοπιστίας, αναπτύξαμε μια νέα τεχνική την οποία ονομάσαμε *τεχνική μεταβλητής ισοτιμίας*. Η κύρια ιδέα πίσω από την τεχνική αυτή είναι ότι μια συγκεκριμένη σελίδα δε χρειάζεται να είναι συνδεδεμένη με ένα συγκεκριμένο

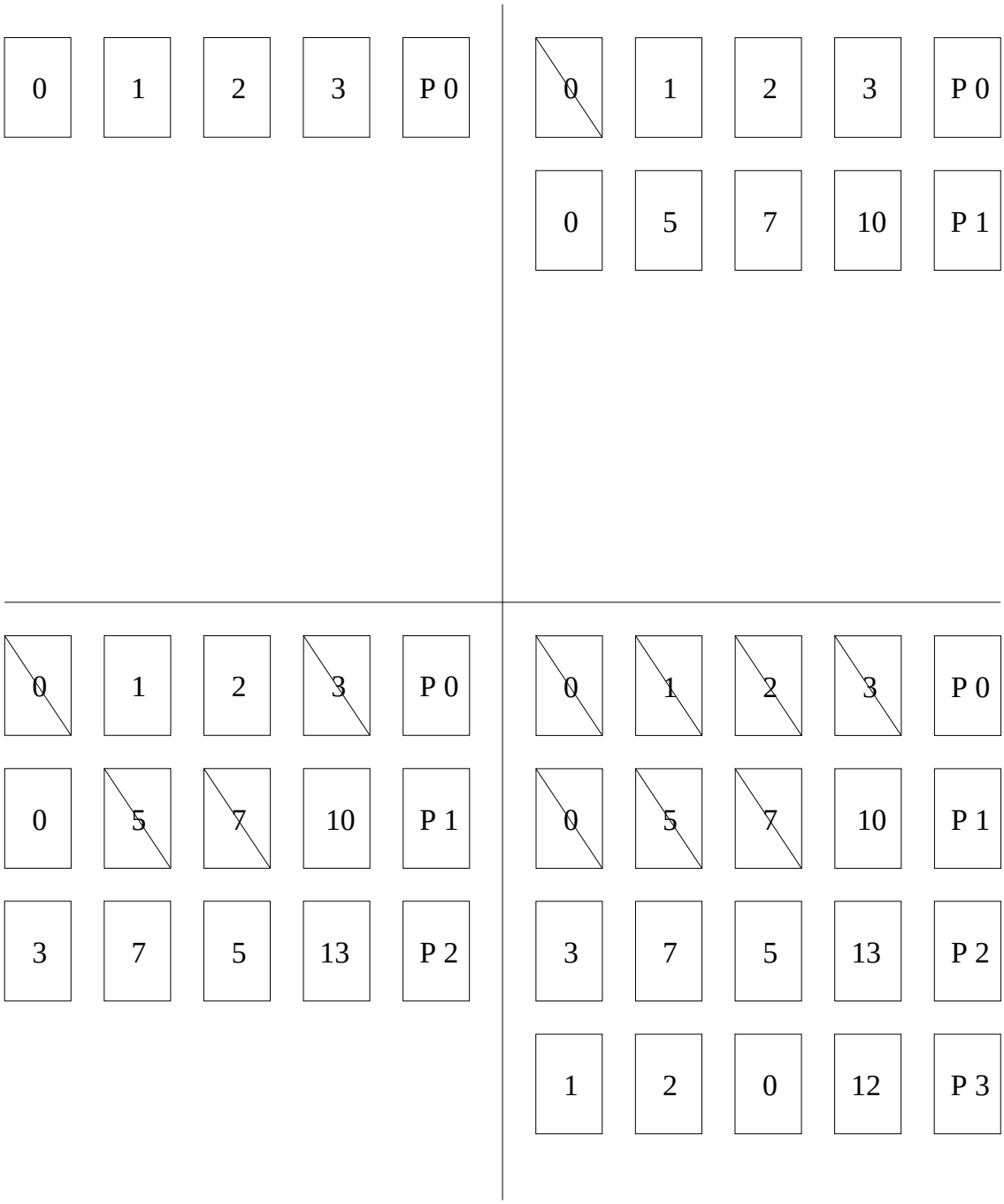
εξυπηρετητή μνήμης ούτε με ένα συγκεκριμένο σύνολο ισοτιμίας. Αντίθετα κάθε φορά που επανεγγράφεται, ένας άλλος εξυπηρετητής μνήμης και ένα νέο σύνολο ισοτιμίας μπορεί να χρησιμοποιηθεί.

Ας υποθέσουμε ότι ο πελάτης χρησιμοποιεί S εξυπηρετητές μνήμης. Για κάθε σελίδα που προωθείται για εγγραφή υπολογίζεται το "αποκλειστικό ή" των δεδομένων της με τα δεδομένα ενός ενταμιευτή μεγέθους σελίδας (που αρχικά είναι γεμάτος μηδενικά) και το αποτέλεσμα αποθηκεύεται στον ενταμιευτή. Στη συνέχεια η σελίδα μεταφέρεται στον επόμενο στη σειρά εξυπηρετητή μνήμης. Οταν έχουν μεταφερθεί S σελίδες, τότε μεταφέρεται και ο ενταμιευτής στον εξυπηρετητή ισοτιμίας. Χρησιμοποιώντας αυτήν την τεχνική, η επιβάρυνση στο χρόνο εκτέλεσης είναι πολύ μικρή, αφού για κάθε εγγραφή σελίδας μεταφέρονται $1 + 1/S$ σελίδες. Αν ένας εξυπηρετητής υποστεί βλάβη, όλες οι σελίδες του μπορούν να ανασκευαστούν υπολογίζοντας το αποκλειστικό ή όλων των σελίδων που μετέχουν στο σύνολο ισοτιμίας τους και της αντίστοιχης σελίδας ισοτιμίας. Η επιβάρυνση στο χρόνο εκτέλεσης είναι μικρή επειδή ο πελάτης δεν είναι υποχρεωμένος να περιμένει επιβεβαίωση από τους εξυπηρετητές που χρησιμοποιεί πριν μεταφερθεί η σελίδα ισοτιμίας, αφού την υπολογίζει ο ίδιος.

Οταν μια σελίδα επανεγγράφεται, σημειώνεται στο παλιό σύνολο ισοτιμίας που την περιέχει σαν *ανενεργή*. Η παλιά όμως έκδοσή της δε σβήνεται από τον εξυπηρετητή μνήμης, αφού σε τέτοια περίπτωση η παλιά σελίδα ισοτιμίας θα έπρεπε να ενημερωθεί, οδηγώντας σε μια επιπλέον μεταφορά σελίδας. Οταν όλες οι σελίδες ενός συνόλου ισοτιμίας έχουν σημειωθεί ως ανενεργές, τόσο οι σελίδες αυτές όσο και η αντίστοιχη σελίδα ισοτιμίας μπορούν να επαναχρησιμοποιηθούν.

Στο σχήμα 4.1 φαίνεται η διαδικασία που ακολουθείται κατά την εγγραφή σελίδων για S ίσο με 4. Αρχικά το λειτουργικό σύστημα δίνει στον οδηγό της συσκευής μας για εγγραφή τις σελίδες 0, 1, 2 και 3, οι οποίες συμπληρώνουν ένα σύνολο ισοτιμίας και αποθηκεύονται στους 4 εξυπηρετητές μνήμης ενώ παράλληλα μεταφέρεται και η αντίστοιχη σελίδα ισοτιμίας στον εξυπηρετητή ισοτιμίας. Στη συνέχεια δίνονται για εγγραφή οι σελίδες 0, 5, 7 και 10. Αυτές αποτελούν ένα νέο σύνολο ισοτιμίας. Ταυτόχρονα με την εγγραφή τους, η παλιά έκδοση της σελίδας 0 σημειώνεται ως ανενεργή. Ομως ο χώρος στον οποίο έχει αποθηκευτεί η παλιά της έκδοση δεν ελευθερώνεται, αφού αν γινόταν κάτι τέτοιο, σε περίπτωση βλάβης π.χ. του δεύτερου εξυπηρετητή δε θα υπήρχε δυνατότητα ανασκευής της σελίδας 1 χρησιμοποιώντας μόνο την σελίδα ισοτιμίας και τις σελίδες 2 και 3. Στη συνέχεια αποθηκεύονται οι σελίδες 3, 7, 5 και 13 που σχηματίζουν ένα νέο σύνολο ισοτιμίας ενώ οι προηγούμενες εκδόσεις των σελίδων 3, 5 και 7 σημειώνονται ως ανενεργές. Τέλος αποθηκεύονται σε ένα νέο σύνολο ισοτιμίας οι σελίδες 1, 2, 0 και 12. Ταυτόχρονα οι προηγούμενες εκδόσεις των σελίδων 0, 1 και 2 σημειώνονται ως ανενεργές. Σημειώνοντας όμως τις σελίδες 1 και 2 ως ανενεργές παρατηρούμε ότι το πρώτο σύνολο ισοτιμίας αποτελείται εξ ολοκλήρου από ανενεργές σελίδες, από παλιές εκδόσεις δηλαδή σελίδων που έχουν ξανααποθηκευτεί. Αρα το πρώτο σύνολο δεν έχει πια καμιά χρησιμότητα και ο χώρος που καταλαμβάνουν τόσο οι σελίδες του όσο και η αντίστοιχη σελίδα ισοτιμίας μπορούν να ελευθερωθούν.

Είναι προφανές ότι κάθε εξυπηρετητής θα πρέπει να διαθέτει ένα μικρό ποσό επιπλέον μνήμης για να υποστηριχθεί αποτελεσματικά η τεχνική της μεταβλητής ισοτιμίας, αφού πολλές εκδόσεις μιας δοσμένης σελίδας μπορεί να βρίσκονται ταυτόχρονα αποθηκευμένες στη μνήμη των εξυπηρετητών. Επίσης, λόγω του φαινομένου αυτού είναι πιθανό κάποιος εξυπηρετητής να εξαντλήσει τη διαθέσιμη μνήμη του.



Σχήμα 4.1: **Η λειτουργία της μεθόδου της μεταβλητής ισοτιμίας.**

Σ' αυτή την περίπτωση, η λύση που ακολουθείται είναι η απελευθέρωση συνόλων ισοτιμίας συνδυάζοντας τις ενεργές σελίδες τους σε νέα σύνολα ισοτιμίας. Στα πειράματά μας χρησιμοποιήσαμε 4 εξυπηρετητές μνήμης και έναν εξυπηρετητή ισοτιμίας, καθένας από τους οποίους διέθετε 10% περισσότερη μνήμη για την αποτελεσματική υποστήριξη της μεταβλητής ισοτιμίας και πολύ σπάνια χρειάστηκε να απελευθερωθούν σύνολα ισοτιμίας συνδυάζοντας τις ενεργές τους σελίδες.

Επιπλέον ο εξυπηρετητής ισοτιμίας δε διαφέρει σε τίποτα από τους εξυπηρετητές μνήμης, αφού απλά ικανοποιεί αιτήσεις ανάγνωσης και εγγραφής σελίδων. Ετσι, το ίδιο πρόγραμμα μπορεί να χρησιμοποιηθεί είτε σαν διεργασία εξυπηρετητή μνήμης είτε σαν διεργασία εξυπηρετητή ισοτιμίας. Στην πράξη οι ίδιοι οι εξυπηρετητές χρησιμοποιούνται για να αποθηκεύσουν τόσο σελίδες δεδομένων όσο και σελίδες ισοτιμίας.

4.3.4 Πειραματικά Αποτελέσματα

Για την πειραματική αξιολόγηση των διαφόρων τεχνικών αξιοπιστίας επαναλάβαμε τις μετρήσεις που παρουσιάστηκαν στο σχήμα 3.5. Οι μέθοδοι που παρουσιάζονται είναι οι εξής

- *Η χρήση της απομακρυσμένης μνήμης χωρίς την εφαρμογή κάποιας μεθόδου αξιοπιστίας.* Η μέθοδος αυτή θα χρησιμοποιηθεί για να μετρήσουμε την επιβάρυνση που προκαλούν στο χρόνο εκτέλεσης οι διάφορες πολιτικές αξιοπιστίας.

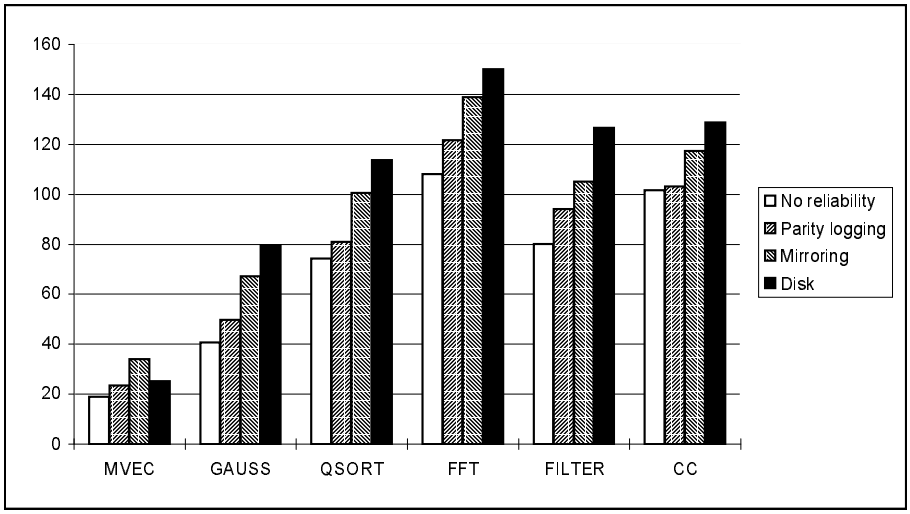
- *Η μεταβλητή ισοτιμία που χρησιμοποιεί 4 εξυπηρετητές μνήμης και έναν εξυπηρετητή ισοτιμίας με 10% επιπλέον μνήμη.*

- *Η τεχνική των διπλών αντιγράφων που χρησιμοποιεί έναν κύριο και ένα δευτερεύοντα εξυπηρετητή μνήμης.*

- *Ο τοπικός δίσκος που χρησιμοποιείται σαν η μέθοδος αναφοράς.* Στην περίπτωση αυτή ο δίσκος χρησιμοποιείται απευθείας από το σελιδοδιαχειριστή του λειτουργικού συστήματος χωρίς να παρεμβαίνει ο σελιδοδιαχειριστής του συστήματός μας.

Οπως φαίνεται στο σχήμα 4.2, η τεχνική της *μεταβλητής ισοτιμίας* δίνει σε όλες τις περιπτώσεις πολύ καλύτερα αποτελέσματα από το *δίσκο*, ενώ η επιβάρυνση που προκαλεί η εξασφάλιση αξιοπιστίας είναι πολύ μικρή. Συγκεκριμένα, η βελτίωση σε σχέση με το *δίσκο* κυμαίνεται από 7.6% ως 59.86%, ενώ η επιβάρυνση που εισάγει είναι μικρότερη του 18.5% και για τις περισσότερες εφαρμογές μικρότερη του 15%. Η τεχνική των *διπλών αντιγάφων* δίνει ελαφρά καλύτερα αποτελέσματα από το *δίσκο* σε όλες τις περιπτώσεις εκτός από μια, ενώ η επιβάρυνση που εισάγει είναι σημαντική και κυμαίνεται από 13.3% ως το 44.14% για την εφαρμογή `MVEC`. Αποτέλεσμα της επιβάρυνσης αυτής είναι το γεγονός ότι για την εφαρμογή `MVEC` η οποία προκαλεί πολλές εγγραφές και ελάχιστες αναγνώσεις σελίδων δίνει χειρότερο χρόνο εκτέλεσης από το *δίσκο*.

Τα παραπάνω αποτελέσματα δείχνουν ότι η επιβάρυνση που προσθέτει η τεχνική της *μεταβλητής ισοτιμίας* είναι εκπληκτικά μικρή. Οσο ο ρυθμός μεταγωγής δεδομένων των χρησιμοποιούμενων δικτύων αυξάνει τόσο η επιβάρυνση αυτή θα γίνεται μικρότερη.



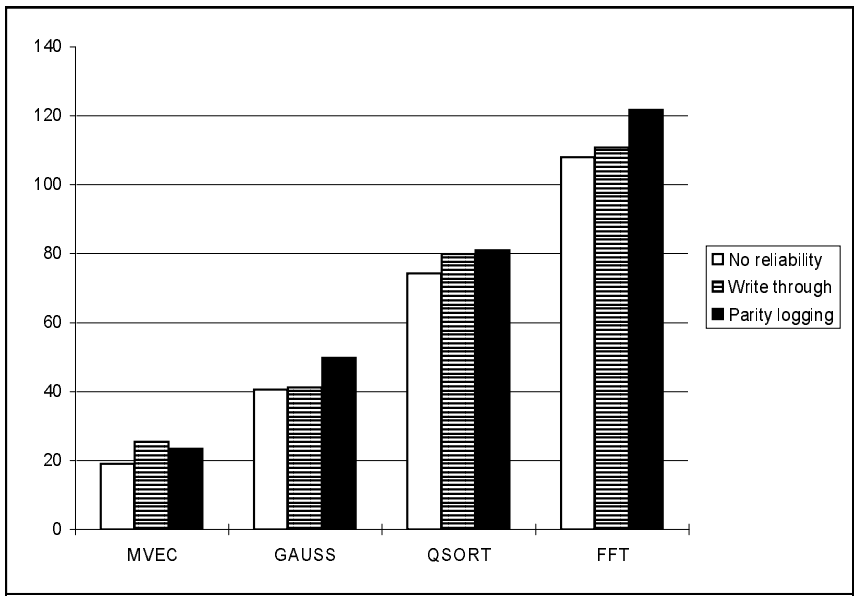
Σχήμα 4.2: **Η απόδοση των εφαρμογών με τη χρήση διαφόρων τεχνικών αξιοπιστίας.** Για όλες τις εφαρμογές, η τεχνική της *μεταβλητής ισοτιμίας* δίνει πολύ καλύτερες επιδόσεις από το δίσκο προσθέτοντας μια πολύ μικρή επιβάρυνση σε σχέση με τη χρήση της απομακρυσμένης μνήμης χωρίς αξιοπιστία.

4.4 Η Χρήση του Τοπικού Δίσκου για την Αύξηση της Αξιοπιστίας

Στην εργασία αυτή χρησιμοποιούμε την απομακρυσμένη μνήμη για να αποθηκεύσουμε τόσο σελίδες που περιέχουν δεδομένα όσο και σελίδες που περιέχουν πλεονάζουσα πληροφορία που χρησιμοποιείται για την ανακατασκευή των σελίδων σε περίπτωση που κάποιος κόμβος υποστεί βλάβη. Μια άλλη προσέγγιση είναι η αποθήκευση των σελίδων τόσο στην απομακρυσμένη μνήμη όσο και στο δίσκο [FMP+95], χρησιμοποιώντας με αυτόν τον τρόπο την απομακρυσμένη μνήμη ουσιαστικά σαν κρυφή μνήμη διεγγραφής για το δίσκο. Θα ήταν χρήσιμη λοιπόν η σύγκριση των δυο αυτών μεθόδων ώστε να μάθουμε κάτω από ποιες συνθήκες είναι προτιμότερη η χρήση καθεμιάς από αυτές.

Καταρχήν και οι δυο μέθοδοι χρησιμοποιούν την απομακρυσμένη μνήμη προκειμένου να ικανοποιήσουν αιτήσεις ανάγνωσης. Αυτό σημαίνει ότι δίνουν την ίδια απόδοση για τις αναγνώσεις σελίδων και επιπλέον ότι αποφεύγουν την καθυστέρηση αναζήτησης και περιστροφής του δίσκου για τις αναγνώσεις, δίνοντας έτσι καλύτερη απόδοση από το δίσκο. Οσον αφορά στις εγγραφές, η μέθοδος της *μεταβλητής ισοτιμίας* μεταφέρει $1 + 1/S$ σελίδες για κάθε εγγραφή λόγω του υπολογισμού και μεταφοράς της σελίδας ισοτιμίας (στα πειράματά μας το S ήταν ίσο με 4). Αντίθετα η μέθοδος της *διεγγραφής* μεταφέρει κάθε σελίδα και στην απομακρυσμένη μνήμη και στο δίσκο. Οι δυο συτές μεταφορές όμως μπορούν να γίνονται παράλληλα. Αυτό σημαίνει ότι η επιλογή της καταλληλότερης μεθόδου εξαρτάται από το ρυθμό μεταγωγής δεδομένων που δίνει ο δίσκος και το δίκτυο. Αν ο ρυθμός μεταγωγής δεδομένων του δικτύου είναι πολύ μεγαλύτερος από το ρυθμό μεταγωγής δεδομένων του δίσκου, τότε ο δίσκος θα αποτελεί το αργότερο κομμάτι του συστήματος, οπότε η μέθοδος της *διεγγραφής* δε θα αποτελεί καλή επιλογή. Αν όμως ο ρυθμός μεταγωγής δεδομένων του δίσκου είναι συγκρίσιμος με το ρυθμό μεταγωγής δεδομένων του δικτύου τότε οι μεταφορές στο δίσκο και το δίκτυο μπορούν να επικαλύπτονται χρονικά σε ένα βαθμό, οπότε δεν είναι προφανές ποια μέθοδος είναι προτιμότερη.

Στο δικό μας περιβάλλον τόσο ο δίσκος όσο και το δίκτυο έχουν ρυθμό μεταγωγής δεδομένων ίσο με 10 Mbps. Οταν χρησιμοποιείται η μέθοδος της *διεγγραφής*, ο



Σχήμα 4.3: **Απόδοση της τεχνικής της** *μεταβλητής ισοτιμίας* **και της τεχνικής της** *διεγγραφής* **πάνω από Ethernet για διάφορες εφαρμογές.** Τα μεγέθη εισόδου είναι 3000 στοιχεία για την `QSORT`, ένας πίνακας 3000 στοιχείων για το `GAUSS`, ένας πίνακας 1700×1700 για τον `MVEC` και ένας πίνακας 700 K σημείων για τον `FFT`.

ρυθμός μεταγωγής δεδομένων που δίνει ο δίσκος είναι κονα στα 10 Mbps αφού δεν έχουμε κίνηση των κεφαλών του δίσκου για τις αναγνώσεις και κάθε εγγραφή αφορά μεγάλο όγκο δεδομένων (τυπικά 4 σελίδες). Σ' αυτό το περιβάλλον, η μέθοδος της *διεγγραφής* δίνει καλύτερα αποτελέσματα από τη μέθοδο της *μεταβλητής ισοτιμίας* για τις περισσότερες εφαρμογές, όπως φαίνεται στο σχήμα 4.3. Ομως πάνω από δίκτυα μεγάλου ρυθμού μεταγωγής δεδομένων, η μέθοδος της *μεταβλητής ισοτιμίας* είναι σαφώς προτιμότερη αφού η μέθοδος της *διεγγραφής* περιορίζεται από το ρυθμό μεταγωγής δεδομένων του δισκου.

4.5 Προσαρμοζόμενη Σελιδοδιαχείριση

Τα αποτελέσματα της ενότητας 4.4 δείχνουν ότι η ανωτερότητα της μιας πολιτικής αξιοπιστίας έναντι της άλλης εξαρτάται από τον *πραγματικό ρυθμό* μεταγωγής δεδομένων για μεταφορά σελίδων του δικτύου σε σχέση με τον *πραγματικό ρυθμό* μεταγωγής δεδομένων για μεταφορά σελίδων του δίσκου. Ο πραγματικός ρυθμός μεταγωγής δεδομένων του δικτύου εξαρτάται από το φόρτο του τοπικού δικτύου, ο οποίος μεταβάλλεται σημαντικά κατά τη διάρκεια μιας ημέρας. Ο πραγματικός ρυθμός μεταγωγής δεδομένων του δίσκου εξαρτάται από το χρόνο αναζήτησης και περιστροφής, ο οποίος εξαρτάται από την ακολουθία των προσπελάσεων στη μνήμη των εκτελούμενων εφαρμογών. Επίσης, συνήθως τα τμήματα του δίσκου που χρησιμοποιούνται για αποθήκευση σελίδων συνυπάρχουν στον ίδιο δίσκο με τμήματα τα οποία αποθηκεύουν συστήματα αρχείων. Αυτό σημαίνει ότι η κίνηση σ' αυτά τα τμήματα μπορεί να αυξήσει το χρόνο αναζήτησης δραματικά.

Θα ήταν λοιπόν χρήσιμο για το σύστημά μας να είναι ικανό να εκτιμά τον πραγματικό ρυθμό μεταγωγής δεδομένων που παρέχει το δίκτυο και ο δίσκος και να προσαρμόζει ανάλογα την πολιτική του. Η προτιμότερη πολιτική θα ήταν να χρησιμοποιεί

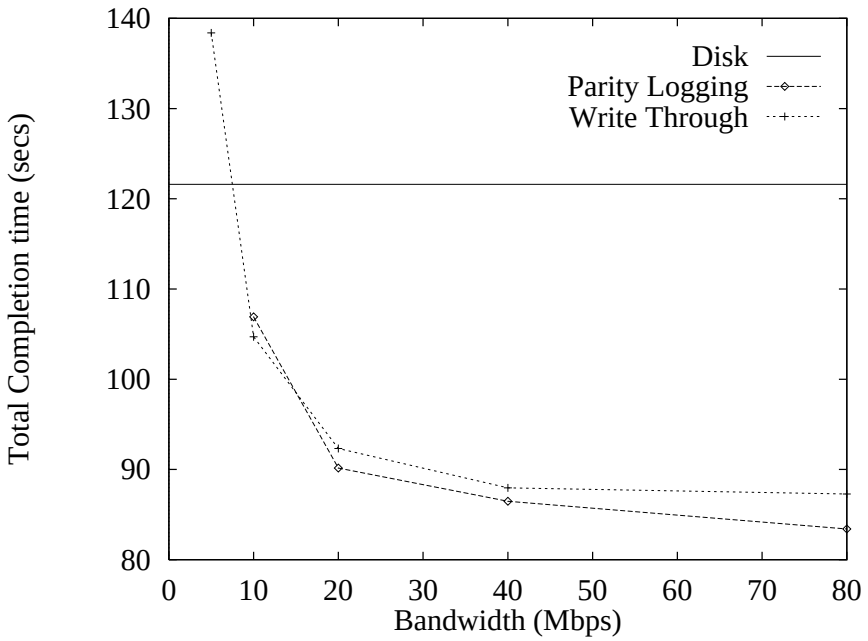
- τον τοπικό *δίσκο*, όποτε ο πραγματικός ρυθμός μεταγωγής του (για εγγραφές και αναγνώσεις) είναι μεγαλύτερος του πραγματικού ρυθμού μεταγωγής δεδομένων του δικτύου.
- τη μέθοδο της *διεγγραφής*, όποτε το δίκτυο προσφέρει μεγαλύτερο ρυθμό μεταγωγής δεδομένων από το δίσκο και ο ρυθμός μεταγωγής δεδομένων του δίσκου για εγγραφές είναι τουλάχιστο τόσος όσος ο ρυθμός μεταγωγής δεδομένων του δικτύου.
- τη μέθοδο της *μεταβλητής ισοτιμίας*, όποτε το δίκτυο προσφέρει μεγαλύτερο ρυθμό μεταγωγής δεδομένων από το δίσκο και ο ρυθμός μεταγωγής δεδομένων του δίσκου για εγγραφές είναι πολύ μικρότερος από το ρυθμό μεταγωγής δεδομένων του δικτύου.

4.5.1 Πειραματικά Αποτελέσματα

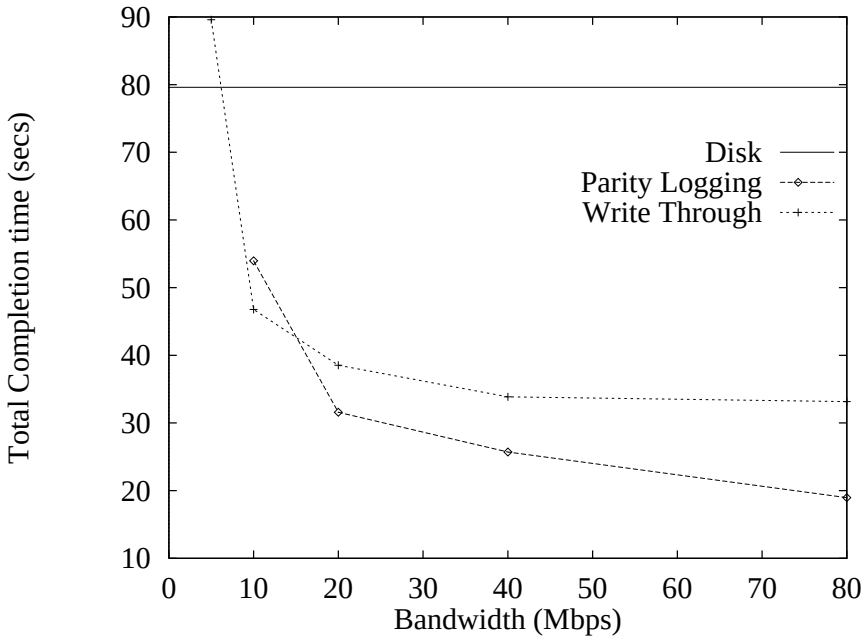
Για να ποσοτικοποιήσουμε την επίδραση του ρυθμού μεταγωγής δεδομένων του δικτύου και του δίσκου σε καθεμιά από τις πολιτικές που αναφέρθηκαν παραπάνω, προσομοιώσαμε την εκτέλεση των εφαρμογών μας πάνω από δίκτυα διαφορετικού ρυθμού μεταγωγής δεδομένων. Για να μετρήσουμε το χρόνο εκτέλεσης των εφαρμογών μας πάνω από ένα δίκτυο με X φορές μικρότερο ρυθμό μεταγωγής δεδομένων από το Ethernet, τροποποιήσαμε το σελιδοδιαχειριστή του συστήματός μας ώστε να μεταφέρει X φορές περισσότερα δεδομένα ανά σελίδα. Με αυτό τον τρόπο, η καθυστέρηση του προσομοιούμενου δικτύου παραμένει η ίδια όπως και στο Ethernet ενώ ο ρυθμός μεταγωγής δεδομένων που παρουσιάζεται στο σύστημά μας είναι X φορές μικρότερος. Για να μετρήσουμε το χρόνο εκτέλεσης πάνω από ένα δίκτυο με X φορές μεγαλύτερο ρυθμό μεταγωγής δεδομένων, τροποποιήσαμε το σελιδοδιαχειριστή του συστήματός μας ώστε να μεταφέρει μόνο το $1/X$ των δεδομένων κάθε σελίδας. Επιπλέον, για να μη χαθούν τα δεδομένα των μεταφερομένων σελίδων στη δεύτερη περίπτωση, αυξήσαμε τη φυσική μνήμη του μηχανήματος–πελάτη και χρησιμοποιήσαμε την επιπλέον αυτή μνήμη για να αποθηκεύσουμε τις σελίδες αυτές. Η επιπλέον αυτή μνήμη δεσμεύτηκε από τον οδηγό της συσκευής μας αποκλειστικά γι' αυτό το σκοπό και δεν αποτελούσε τμήμα της μνήμης που διαχειρίζεται το υποσύστημα ιδεατής μνήμης. Δηλαδή η επιπλέον αυτή μνήμη ήταν αόρατη στις εφαρμογές του χρήστη.

Το σχήμα 4.4 παρουσιάζει το χρόνο εκτέλεσης για την εφαρμογή `FFT` για καθεμιά από τις πολιτικές αξιοπιστίας σαν συνάρτηση του ρυθμού μεταγωγής δεδομένων του δικτύου. Κατα τη διάρκεια αυτών των πειραμάτων δεν υπήρχε μεταφορά δεδομένων προς άλλα τμήματα του δίσκου που χρησιμοποιήθηκε για αποθήκευση σελίδων, γεγονός που σημαίνει ότι ο πραγματικός ρυθμός δεδομένων του δίσκου εξαρτάται αποκλειστικά από την ακολουθία προσπελάσεων στη μνήμη κάθε εφαρμογής.

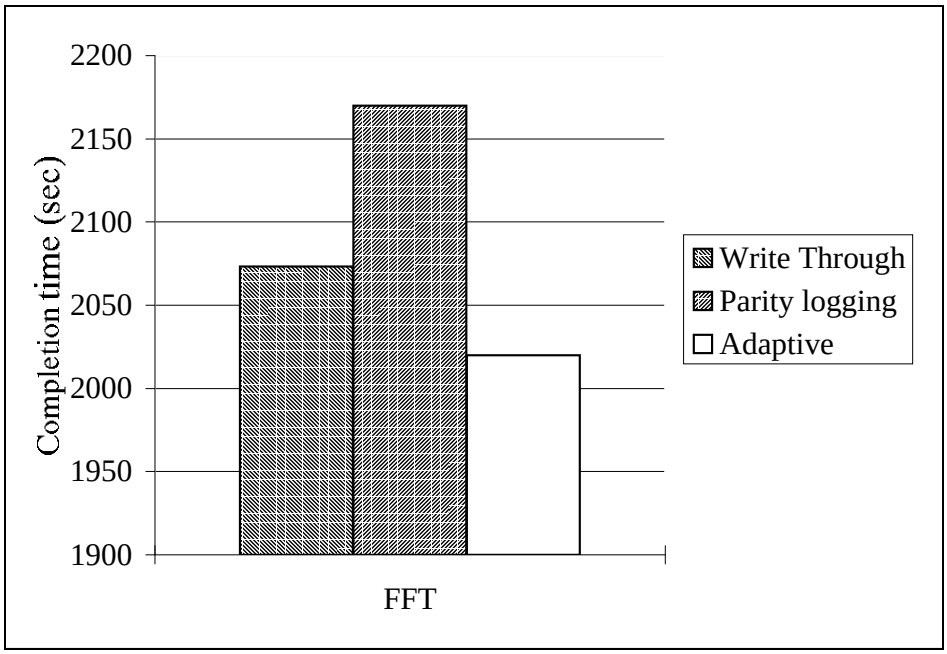
Ο πραγματικός ρυθμός μεταγωγής δεδομένων του δίσκου για την εφαρμογή `FFT` μετρήθηκε ίσος με 7.5 Mbps. Γι' αυτό και ο *δίσκος* δίνει καλύτερη απόδοση από όλες τις άλλες πολιτικές όταν ο ρυθμός μεταγωγής δεδομένων του δικτύου είναι μικρότερος των 7.5 Mbps. Μόλις ο ρυθμός μεταγωγής δεδομένων του δικτύου υπερβεί τα 7.5 Mbps, η τεχνική της *διεγγραφής* δίνει την καλύτερη απόδοση. Σ' αυτή την περίπτωση, οι



Σχήμα 4.4: **Απόδοση του** `FFT` **χρησιμοποιώντας διάφορες τεχνικές αξιοπιστίας.** Το μέγεθος εισόδου είναι μια ακολουθία με 650K στοιχεία.



Σχήμα 4.5: **Απόδοση του** `GAUSS` **χρησιμοποιώντας διάφορες τεχνικές αξιοπιστίας.** Το μέγεθος εισόδου είναι ένας πίνακας με 1700 στοιχεία.

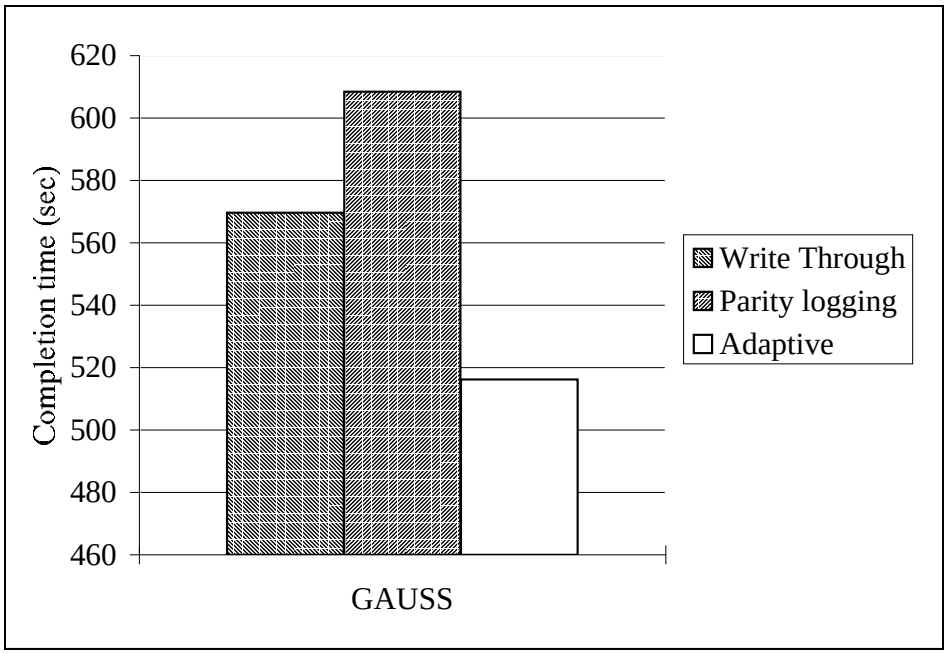


Σχήμα 4.6: **Τα οφέλη της ικανότητας προσαρμογής για την εφαρμογή FFT.** Το μέγεθος της εισόδου είναι 20 ακολουθίες των 650K στοιχείων.

εγγραφές γίνονται και στο δίκτυο και στο δίσκο, ενώ οι αναγνώσεις ικανοποιούνται από το δίκτυο. Αφού ο ρυθμός μεταγωγής δεδομένων του δίσκου για εγγραφή είναι τουλάχιστο τόσος όσος και ο ρυθμός μεταγωγής του δικτύου, οι εγγραφές στο δίσκο και το δίκτυο εκτελούνται παράλληλα. Μόλις ο ρυθμός μεταγωγής δεδομένων του δικτύου γίνει $(S+1)/S$ φορές μεγαλύτερος από το ρυθμό μεταγωγής δεδομένων του δίσκου για εγγραφές, η μέθοδος της *μεταβλητής ισοτιμίας* γίνεται η καλύτερη. Σ' αυτή την περίπτωση το S είναι ίσο με τον αριθμό των εξυπηρετητών που συμμετέχουν σε ένα σύνολο ισοτιμίας. Οταν χρησιμοποιείται η μέθοδος της *μεταβλητής ισοτιμίας*, μια εγγραφή σελίδας απαιτεί $(1+1/S)$ μεταφορές σελίδων σε σύγκριση με τη μεταφορά μιας μόνο σελίδας που απαιτεί η μέθοδος της *διεγγραφής* και στο δίσκο και στο δίκτυο. Στην περίπτωσή μας το S ήταν ίσο με 4, οπότε η μέθοδος της *μεταβλητής ισοτιμίας* δίνει τα καλύτερα αποτελέσματα μόλις ο ρυθμός μεταγωγής δεδομένων του δικτύου υπερβεί τα $(5/4) * 10$ ή τα 12.5 Mbps, αφού ο ρυθμός μεταγωγής δεδομένων του δίσκου για εγγραφές είναι κοντά στα 10 Mbps. Παρόμοια είναι και η συμπεριφορά της εφαρμογής `GAUSS`, όπως φαίνεται στο σχήμα 4.5.

Τα αποτελέσματα αυτά δείχνουν καθαρά τη σημασία της προσαρμοστικότητας του συστήματος. Υλοποιήσαμε τους μηχανισμούς εκτίμησης του ρυθμού μεταγωγής δεδομένων που προσφέρει τόσο ο δίσκος όσο και το δίκτυο μετρώντας το χρόνο που απαιτείται προκειμένου να εξυπηρετηθεί ένα σύνολο μεταφορών σελίδων στα δυο αυτά μέσα. Κάθε λεπτό επανεκτιμάται ο ρυθμός μεταγωγής δεδομένων και ο σελιδοδιαχειριστής του συστήματός μας προσαρμόζεται στην καταλληλότερη πολιτική. Η επιβάρυνση που προκαλεί στο χρόνο εκτέλεσης αυτή η εκτίμηση μετρήθηκε να είναι μικρότερη από 0.5 sec, δηλαδή μικρότερη από 1%.

Για να επιβεβαιώσουμε τα ευεργετικά αποτελέσματα της ικανότητας προσαρμογής του συστήματος, μετρήσαμε το χρόνο εκτέλεσης των εφαρμογών μας μεταβάλλοντας



Σχήμα 4.7: **Τα οφέλη της ικανότητας προσαρμογής για την εφαρμογή GAUSS.** Το μέγεθος της εισόδου είναι 20 πίνακες των 1600 στοιχείων.

το ρυθμό μεταγωγής δεδομένων του δικτύου από τα 10 Mbps στα 80 Mbps σε συγκεκριμένο σημείο εκτέλεσης του πηγαίου κώδικα των εφαρμογών μας. Αυτό σημαίνει ότι η καλύτερη πολιτική ήταν στο πρώτο μέρος η μέθοδος της *διεγγραφής* και στο δεύτερο μέρος, όταν ο ρυθμός μεταγωγής δεδομένων γίνεται ίσος με 80 Mbps, η χρήση της *μεταβλητής ισοτιμίας*. Μετρήσαμε το χρόνο εκτέλεσης των εφαρμογών μας όταν χρησιμοποιείται μια στατική πολιτική (*διεγγραφή* ή *μεταβλητή ισοτιμία*) και όταν χρησιμοποιείται η προσαρμοζόμενη έκδοση του σελιδοδιαχειριστή μας. Στη δεύτερη περίπτωση, ο σελιδοδιαχειριστής μας κατάφερε να ανιχνεύσει την αύξηση στο ρυθμό μεταγωγής δεδομένων του δικτύου και να μεταβάλλει την πολιτική του από *διεγγραφή* σε *μεταβλητή ισοτιμία* δίνοντας μέχρι και 18% καλύτερη απόδοση από τις στατικά ορισμένες πολιτικές, όπως φαίνεται στα σχήματα 4.6 και 4.7. Τα οφέλη θα μπορούσαν να είναι ακόμη μεγαλύτερα αν η επανεκτίμηση του ρυθμού μεταγωγής δεδομένων γινόταν συχνότερα από μια φορά το λεπτό. Τα καλύτερα αποτελέσματα που δίνει σε σχέση με τη σταθερή χρήση μιας πολιτικής δείχνουν ότι η ικανότητα προσαρμογής είναι ένα χαρακτηριστικό επιβεβλημένο για την αποδοτική του λειτουργία, ειδικά σε ένα περιβάλλον τοπικού δικτύου όπου οι συνθήκες του περιβάλλοντος μπορούν να μεταβάλλονται συνεχώς.

Κεφάλαιο 5

Σχετική Εργασία

Στο κεφάλαιο αυτό παρουσιάζονται συστήματα τα οποία έχουν κάποια σχέση με την εργασία αυτή. Τα συστήματα αυτά παρουσιάζονται στις επόμενες ενότητες και εξετάζονται σε σχέση με το σύστημά μας.

5.1 Συστήματα που Βασίζονται σε Ειδικό Υλικό

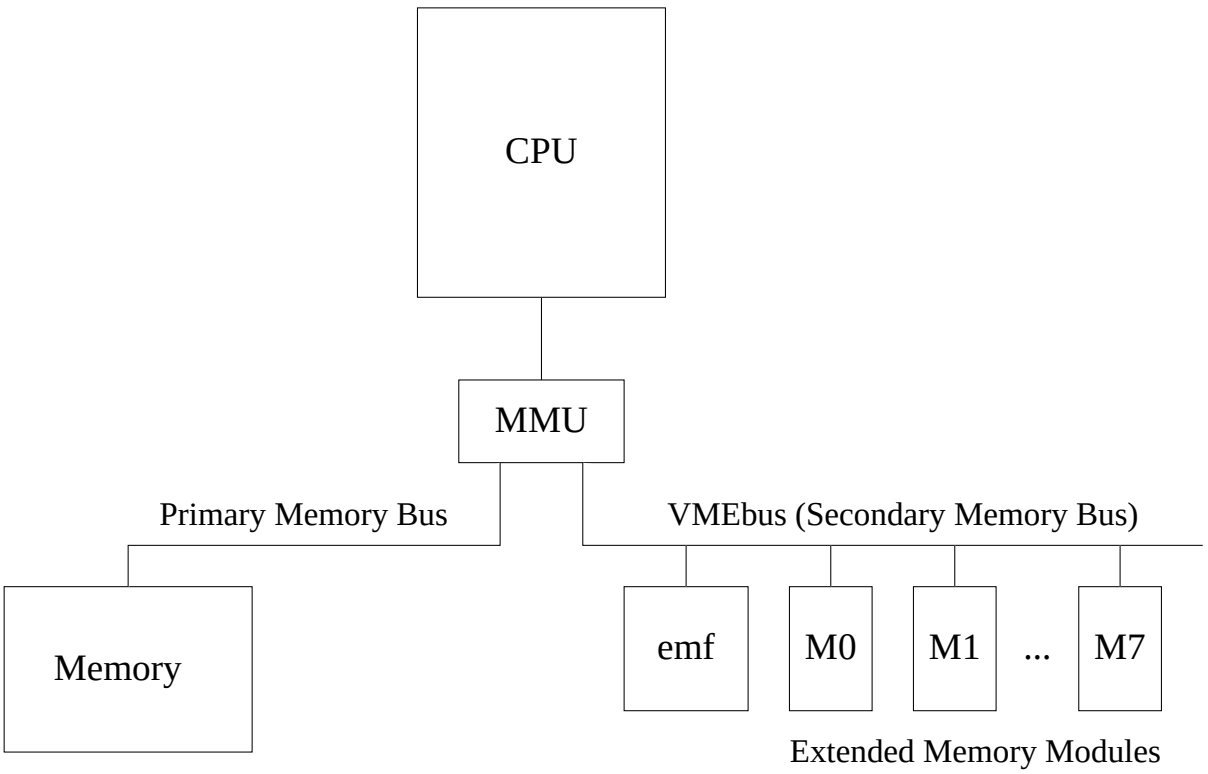
5.1.1 Επεκτάσεις Μνήμης με τη Χρήση του Διαύλου Εισόδου/Εξόδου

Οι Li και Petersen έχουν υλοποιήσει ένα σύστημα επέκτασης μνήμης, προσθέτοντας κάρτες μνήμης που περιέχουν επιπλέον και ένα μικροεπεξεργαστή για τη διαχείρισή τους στο διάδρομο εισόδου/εξόδου [LP91], όπως φαίνεται στο σχήμα 5.1. Ο σελιδο–διαχειριστής του λειτουργικού συστήματος έχει τροποποιηθεί ώστε να αναγνωρίζει την ύπαρξη της πρόσθετης μνήμης η οποία μπορεί να χρησιμοποιηθεί και για την αποθήκευση σελίδων όπως φαίνεται στο σχήμα 5.2 αλλά και με απλές εντολές ανά–γνωσης/εγγραφής σαν κανονική (αλλά πιο αργή) μνήμη, όπως φαίνεται στο σχήμα 5.3. Η πρόσβαση στην επεκταμένη μνήμη είναι προφανώς ακριβότερη από την πρόσβαση στην κύρια μνήμη, αφού βρίσκεται πάνω στο διάδρομο εισόδου/εξόδου, που στην περίπτωση αυτή είναι ο διάδρομος VME [IEE87].

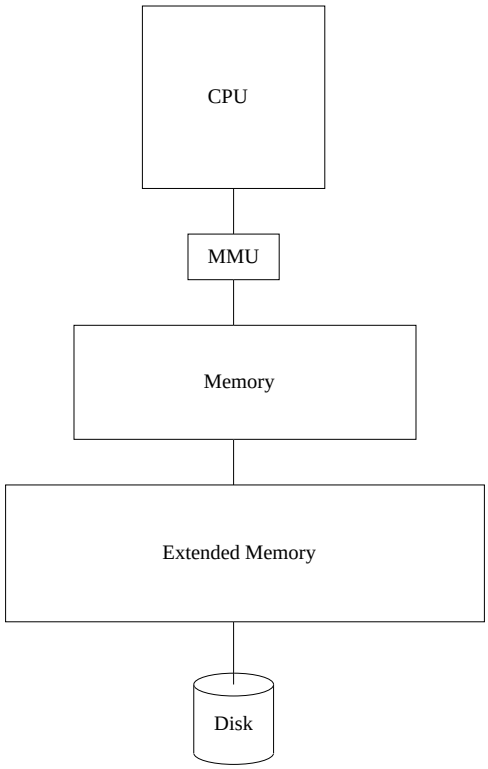
Η χρήση της επεκταμένης μνήμης για αποθήκευση σελίδων δημιουργεί ένα επι–πλέον επίπεδο στην ιεραρχία μνήμης, όπως ακριβώς συμβαίνει και με τη χρήση της απομακρυσμένης μνήμης. Τα πειραματικά αποτελέσματα που παρουσιάζονται δεί–χνουν σημαντική βελτίωση στο χρόνο εκτέλεσης των εφαρμογών, αφού μειώνονται οι προσπελάσεις στο δίσκο. Επίσης η χρήση της επεκταμένης μνήμης σαν κανο–νική αλλά πιο αργή κύρια μνήμη φαίνεται να δίνει καλύτερα αποτελέσματα για τις εφαρμογές που εξετάστηκαν από τη χρήση της για αποθήκευση σελίδων.

5.1.2 eNVy

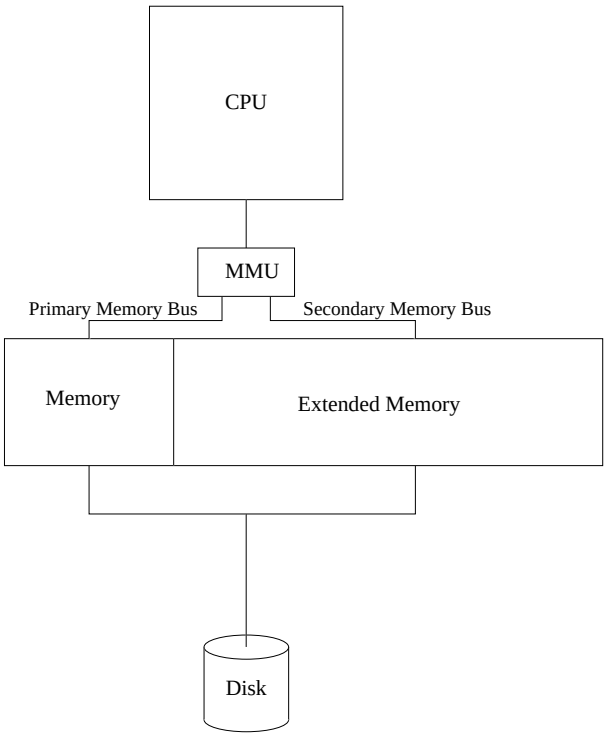
Το eNVy είναι ένα σύστημα που αναπτύχθηκε στο πανεπιστήμιο Rice με στόχο την αποφυγή της χρήσης του δίσκου για την αποθήκευση δεδομένων [WZ95]. Χρησιμοποιεί μνήμες ειδικού τύπου, οι οποίες διατηρούν τα περιεχόμενά τους χωρίς την ανάγκη ύπαρξης οποιασδήποτε πηγής ενέργειας, είναι λίγο ακριβότερες από την κύρια μνήμη, και έχουν τον ίδιο χρόνο ανάγνωσης με αυτή. Το μειονέκτημά τους είναι ότι δεν μπορεί να γίνει εγγραφή μιας λέξης μόνο, αλλά αντίθετα ένα ολόκληρο μπλοκ μνήμης πρέπει να σβηστεί και να επαναπρογραμματιστεί κάθε φορά, διαδικασία που διαρκεί αρκετά



Σχήμα 5.1: **Η επεκταμένη μνήμη στο δίαυλο VME.**



Σχήμα 5.2: **Η χρήση της επεκταμένης μνήμης αντί του δίσκου για την αποθήκευση σελίδων που δε χωρούν στην κύρια μνήμη.**



Σχήμα 5.3: **Η χρήση της επεκταμένης μνήμης σαν πιο αργή κύρια μνήμη.**

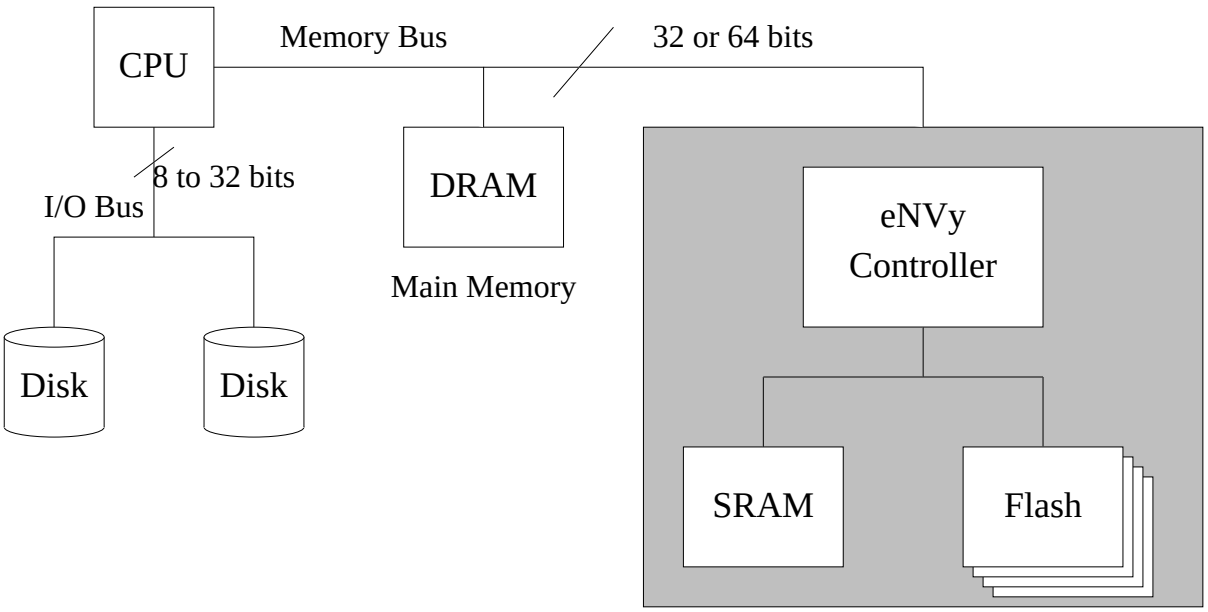
μs. Μια μικρή στατική μνήμη χρησιμοποιείται σαν ενταμιευτής εγγραφής. Το όλο σύστημα φαίνεται στο σχήμα 5.4.

Το eNVy μπορεί να χρησιμοποιηθεί και για την αποθήκευση σελίδων μνήμης αντί του δίσκου, αλλά και σαν κύρια μνήμη. Στη δεύτερη περίπτωση, η μνήμη του χωρίζεται σε σελίδες και κάθε φορά που γίνεται μια εγγραφή σε σελίδα του, δημιουργείται ένα νέο αντίγραφό της που περιέχει τα νέα δεδομένα.

5.1.3 Σύγκριση

Τα προαναφερθέντα συστήματα αυξάνουν την κύρια μνήμη που είναι διαθέσιμη σε ένα μηχάνημα προσθέτοντας πιο αργή μνήμη. Με αυτόν τον τρόπο μειώνουν τις προσπελάσεις στο δίσκο. Ομως απαιτούν ειδικό υλικό, το οποίο αυξάνει το κόστος του υπολογιστή. Επιπλέον τα υπόλοιπα μηχανήματα του ίδιου τοπικού δικτύου δεν μπορούν να επωφεληθούν από την ύπαρξη της επιπλέον αυτής μνήμης. Βέβαια τα συστήματα αυτά παρέχουν τη δυνατότητα είτε δημιουργίας ενός νέου επιπέδου στην ιεραρχία μνήμης είτε αύξησης της κύριας μνήμης. Ομως και η απομακρυσμένη μνήμη θα μπορούσε να προσπελαστεί με απλές εντολές ανάγνωσης/εγγραφής αν χρησιμοποιούνταν το κατάλληλο δίκτυο διασύνδεσης ([Dol94, MK96, SL91, BJMW95, Gil95, BLA+94]).

Γενικά εμείς δεν αυξάνουμε το κόστος των υπολογιστών προσθέτοντας επιπλέον μνήμη και ειδικό υλικό αλλά χρησιμοποιούμε την ελεύθερη μνήμη των μηχανών του τοπικού δικτύου. Επίσης η διαθέσιμη μνήμη αυξάνεται αναλογικά με τον αριθμό των μηχανών του δικτύου μας.



Σχήμα 5.4: **Διάγραμμα του συστήματος eNVy.**

5.2 Συστήματα που Χρησιμοποιούν Εξωτερικούς Εξυπηρετητές Μνήμης

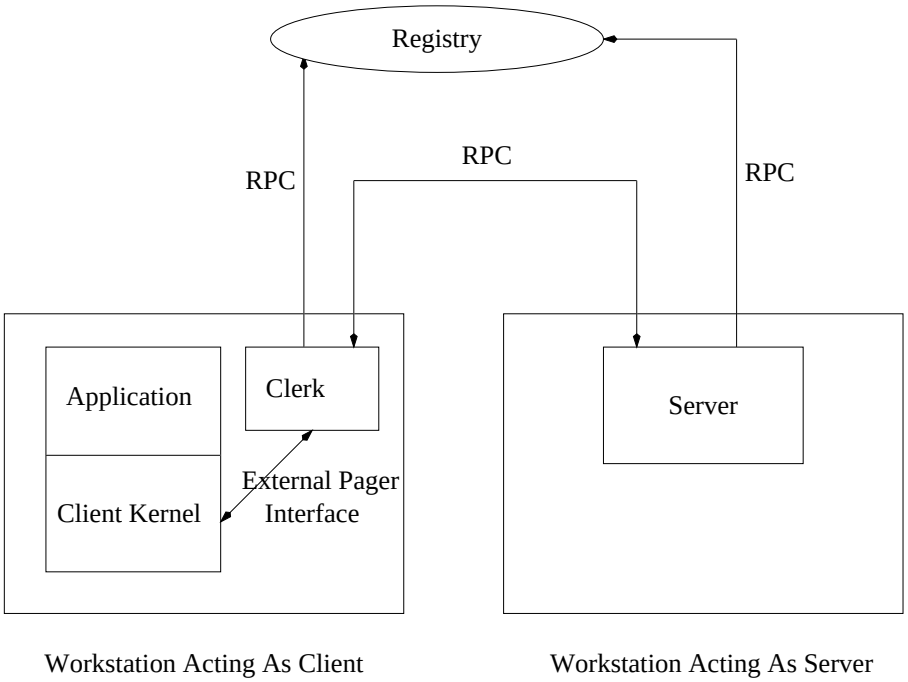
5.2.1 Το Σύστημα του Πανεπιστημίου της Washington

Στο πανεπιστήμιο της Washington έχει υλοποιηθεί ένα σύστημα χρήσης της απομακρυσμένης μνήμης για αποθήκευση σελίδων σε ένα δίκτυο από σταθμούς εργασίας [FZ91]. Οι σταθμοί εργασίας είναι DEC Firefly [TSJ88] και το λειτουργικό σύστημα που χρησιμοποιείται το Topaz [SB90]. Το σύστημα αυτό χρησιμοποιεί εξυπηρετητές μνήμης σε κάθε κόμβο του δικτύου. Ο πυρήνας του λειτουργικού συστήματος έχει τροποποιηθεί ώστε κάθε διεργασία να είναι σε θέση να καθορίσει ποιες από τις σελίδες της μπορούν να αποθηκευτούν στην απομακρυσμένη μνήμη και να δημιουργηθεί ένα περιβάλλον υποστήριξης εξωτερικών εξυπηρετητών μνήμης. Η επικοινωνία γίνεται χρησιμοποιώντας απομακρυσμένες κλήσεις διαδικασιών όπως φαίνεται στο σχήμα 5.5.

Τα αποτελέσματά τους πάνω από Ethernet και FDDI είναι ενθαρρυντικά. Παράλληλα παρουσιάζουν και ένα αναλυτικό μοντέλο πρόβλεψης της επίδοσης του συστήματός τους. Ο χρόνος μεταφοράς μιας σελίδας είναι ίσος με 13 ms, αρκετά από τα οποία οφείλονται στην επιβάρυνση του λογισμικού (πρωτόκολλα και χρήση εξωτερικού εξυπηρετητή μνήμης).

5.2.2 Η Χρήση Απομακρυσμένης Μνήμης σε Φορητούς Υπολογιστές

Στο πανεπιστήμιο Columbia έχει υλοποιηθεί ένα σύστημα απομακρυσμένης μνήμης για φορητούς υπολογιστές χρησιμοποιώντας εξωτερικούς εξυπηρετητές μνήμης πάνω από το Mach 2.5 [SD91]. Η απόδοση του συστήματος αυτού είναι παραπλήσια με την απόδοση του τοπικού δίσκου. Ο χρόνος που απαιτείται για τη μεταφορά μιας σελίδας μεγάθους 4 KB είναι ίσος με 45 ms, από τα οποία τα περισσότερα οφείλονται στην επιβάρυνση του λογισμικού λόγω της χρήσης εξωτερικού εξυπηρετητή μνήμης.



Σχήμα 5.5: **Το σύστημα του πανεπιστημίου της Washington.**

5.2.3 Σύγκριση

Τα συστήματα που χρησιμοποιούν εξωτερικούς εξυπηρετητές μνήμης έχουν σαν αποτέλεσμα τη μεγάλη επιβάρυνση στο χρόνο μεταφοράς σελίδων λόγω της μεταφοράς δεδομένων ανάμεσα σε πολλούς χώρους διευθύνσεων. Για παράδειγμα, στο σύστημα του Columbia η επιβάρυνση που οφείλεται στο TCP και τον κώδικα διεργασιακής επικοινωνίας του Mach είναι μεγαλύτερη από 16 ms, ενώ στο δικό μας σύστημα η αντίστοιχη επιβάρυνση είναι 10 φορές μικρότερη. Επιπλέον το σύστημά μας δεν απαιτεί την αλλαγή του κώδικα του λειτουργικού συστήματος, όπως συμβαίνει με το σύστημα της Washington, ενώ εξασφαλίζει την ομαλή λειτουργία του ακόμη και μετά από βλάβη κάποιου μηχανήματος του δικτύου.

5.3 Η Χρήση της Απομακρυσμένης Μνήμης Τροποποιώντας την Εφαρμογή

Στο πανεπιστήμιο του Berkeley, έχει υλοποιηθεί ένα σύστημα χρήσης της απομακρυσμένης μνήμης για αποθήκευση σελίδων, το οποίο βασίζεται στην τροποποίηση του πηγαίου κώδικα των εφαρμογών [AN95]. Συγκεκριμένα, για να χρησιμοποιήσει μια εφαρμογή την απομακρυσμένη μνήμη πρέπει να αντικατασταθούν οι κλήσεις δέσμευσης μνήμης από μια νέα συνάρτηση η οποία δεσμεύει απομακρυσμένη μνήμη.

Οταν μια διεργασία προσπελάσει μια σελίδα η οποία δεν υπάρχει στη μνήμη, το λειτουργικό σύστημα προκαλεί ένα σήμα, με αποτέλεσμα την κλήση μιας ρουτίνας διαχείρισης του σήματος αυτού, η οποία φέρνει τη σελίδα από την απομακρυσμένη μνήμη. Αυτό επιτυγχάνεται με τη χρήση ενός τοπικού αρχείου τμήματα του οποίου απεικονίζονται στον ιδεατό χώρο διευθύνσεων της διεργασίας.

Το σύστημα αυτό δίνει καλά αποτελέσματα πάνω από δίκτυα ATM και συγκεκριμένα

2 ως 5 φορές καλύτερη απόδοση από τον τοπικό δίσκο. Τα κυριότερα χαρακτηριστικά του είναι τα παρακάτω:

- *Απαιτεί την τροποποίηση του πηγαίου κώδικα της εφαρμογής.* Το γεγονός αυτό έχει σαν αποτέλεσμα οι εφαρμογές για τις οποίες δε διαθέτουμε πηγαίο κώδικα δεν μπορούν να επωφεληθούν από τη χρήση της απομακρυσμένης μνήμης. Ομως με αυτό τον τρόπο η ανάγκη για αξιόπιστη μεταφορά δεδομένων δεν είναι τόσο επιτακτική, αφού τα απαραίτητα προγράμματα για τη σωστή λειτουργία του συστήματος (π.χ. ο init) δε χρησιμοποιούν την απομακρυσμένη μνήμη.

- *Επιβαρύνει την εφαρμογή με επιπλέον κόστος λόγω του χειρισμού σημάτων και της συχνής αλλαγής χώρων διευθύνσεων.* Το κόστος αυτό είναι της τάξης των αρκετών δεκάδων usec, το οποίο είναι αμελητέο για δίκτυα μικρού ρυθμού μεταγωγής δεδομένων, όμως πάνω από σύγχρονα δίκτυα με υψηλό ρυθμό μεταγωγής δεδομένων και μικρή καθυστέρηση γίνεται σημαντικό.

- *Εχει ανεπιθύμητες παρενέργειες λόγω της χρήσης αρχείων που απεικονίζονται στη μνήμη, αφού τμήματά τους μπορεί να μεταφερθούν στο δίσκο.* Αυτή η συμπεριφορά οφείλεται στο γεγονός ότι η μέθοδος αυτή είναι τελείως αποκομένη από το λειτουργικό σύστημα, το οποίο δε γνωρίζει αν μια σελίδα έχει αποθηκευτεί στη μακρινή μνήμη ή όχι.

5.4 Η Χρήση Απομακρυσμένης Μνήμης σε Μηχανήματα Χωρίς Τοπικό Δίσκο

Στο πανεπιστήμιο Purdue, έχει αναπτυχθεί ένα σύστημα χρήσης της απομακρυσμένης μνήμης για αποθήκευση σελίδων [CG90]. Στο σύστημα αυτό υπάρχουν μηχανήματα που χρησιμοποιούνται αποκλειστικά ως εξυπηρετητές μνήμης από τα υπόλοιπα μηχανήματα του δικτύου. Η απόδοση του συστήματος συγκρίνεται με την απόδοση της μεταφοράς σελίδων σε απομακρυσμένους δίσκους μέσω NFS [SGK+85] και τα αποτελέσματα που προκύπτουν είναι ενθαρρυντικά. Το δίκτυο που χρησιμοποιείται είναι το Ethernet, ενώ για τη μεταφορά δεδομένων αναπτύχθηκε ένα νέο πρωτόκολλο επικοινωνίας πάνω από το λειτουργικό σύστημα Xinu [Com91].

Οι σημαντικότερες διαφορές του συστήματός μας από το [CG90] είναι οι παρακάτω.

- Αποδεικνύουμε ότι η μεταφορά σελίδων στην απομακρυσμένη μνήμη είναι προτιμότερη από τη μεταφορά στον *τοπικό δίσκο*, ενώ το [CG90] δείχνει ότι η μεταφορά σελίδων στην απομακρυσμένη μνήμη είναι προτιμότερη από τη μεταφορά στον *απομακρυσμένο* δίσκο.

- Δε χρησιμοποιούμε μηχανήματα αποκλειστικά ως εξυπηρετητές μνήμης αλλά αντίθετα κάθε μηχάνημα μπορεί να λειτουργεί είτε ως πελάτης είτε ως εξυπηρετητής, ανάλογα με το φόρτο μνήμης που έχει σε κάθε χρονική στιγμή.

- Δεν αλλάζουμε τον κώδικα του λειτουργικού συστήματος και χρησιμοποιούμε πρωτόκολλα διαθέσιμα σε κάθε σύστημα Unix.

5.5 NOW

Στο [ACP95] οι Anderson, Culler και Patterson προτείνουν τη χρήση της απομακρυσμένης μνήμης για αποθήκευση σελίδων. Οι προσομειώσεις τους δείχνουν ότι η χρήση της απομακρυσμένης μνήμης πάνω από το Ethernet δίνει ελαφρά καλύτερα αποτελέσματα από τη χρήση του τοπικού δίσκου σε ένα δίκτυο από DEC 3000/400 [Dig93a]. Επίσης, σύμφωνα με τα αποτελέσματά τους, η χρήση της απομακρυσμένης μνήμης δίνει 5 ως 10 φορές καλύτερο χρόνο εκτέλεσης από τον τοπικό δίσκο αν χρησιμοποιηθεί δίκτυο ATM [New94].

Στην επόμενη εργασία της ίδιας ομάδας [MYW94], περιγράφεται η υλοποίηση ενός συστήματος απομακρυσμένης μνήμης πάνω από ένα δίκτυο ATM με SuperSparc 10/20 και το λειτουργικό σύστημα Solaris [Sun92]. Η μεταφορά των σελίδων γίνεται από ένα δαίμονα που εκτελείται σε κατάσταση χρήστη και ο οποίος επικοινωνεί με τον πυρήνα μέσω ενός οδηγού συσκευής που έχει αναπτυχθεί.

Οι κυριότερες διαφορές του συστήματός μας σε σχέση με το παραπάνω που αναπτύχθηκε στο πανεπιστήμιο του Berkeley είναι οι εξής

- *Οι σελίδες μεταφέρονται κατευθείαν από τον οδηγό συσκευής, δηλαδή από το χώρο διευθύνσεων του πυρήνα αποφεύγοντας την επιβάρυνση που προκαλούν οι αλλαγές χώρου διευθύνσεων και οι άσκοπες αντιγραφές των δεδομένων.*

- *Παρουσιάζουμε πειραματικά αποτελέσματα από πραγματικές εφαρμογές που δείχνουν πόσο χρήσιμη είναι η απομακρυσμένη μνήμη ακόμη και πάνω από δίκτυα χαμηλού ρυθμού μεταγωγής δεδομένων, όπως το Ethernet.*

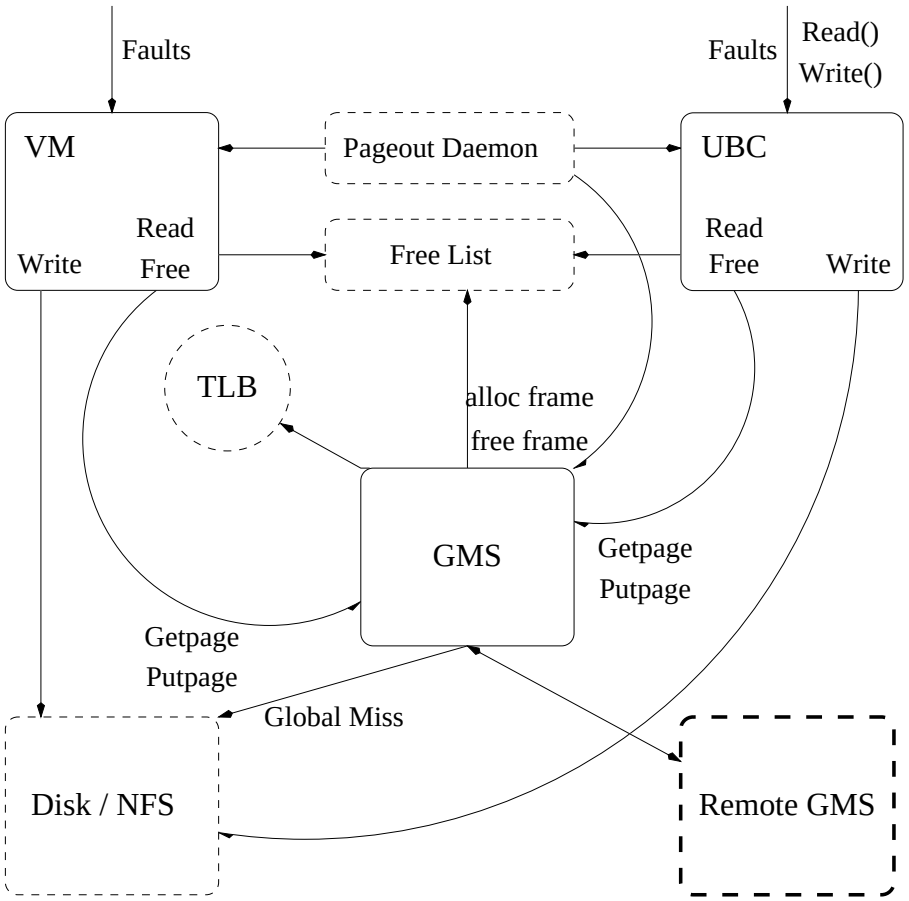
- *Το σύστημά μας έχει τη δυνατότητα προσαρμογής στις μεταβαλλόμενες συνθήκες ενός τοπικού δικτύου.* Ετσι έχει τη δυνατότητα χρήσης διαφόρων πολιτικών αξιοπιστίας αλλά και συσκευών με στόχο τη μεγιστοποίηση της απόδοσης.

- *Περιγράφουμε και αξιολογούμε διάφορες τεχνικές αξιόπιστης αποθήκευσης δεδομένων ώστε να είναι δυνατή η συνέχιση της λειτουργίας του συστήματός μας ακόμη και μετά από βλάβη κάποιας μηχανής του τοπικού δικτύου.*

5.6 Υλοποίηση Κοινής Μνήμης σε ένα Δίκτυο Υπολογιστών

Στο [FMP+95] περιγράφεται ένα σύστημα που υλοποιήθηκε στο πανεπιστήμιο της Washington, το οποίο επανασχεδιάζει το σύστημα διαχείρισης μνήμης στα μηχανήματα ενός τοπικού δικτύου ώστε η μνήμη όλων των μηχανών να αντιμετωπίζεται σαν ενιαία κοινή κατανεμημένη μνήμη. Το σύστημα αυτό έχει υλοποιηθεί σε δίκτυο ATM με μηχανήματα DEC και λειτουργικό σύστημα το DEC OSF/1. Το λειτουργικό σύστημα και συγκεκριμένα το τμήμα διαχείρισης της μνήμης έχει τροποποιηθεί όπως φαίνεται στο σχήμα 5.6 και η μνήμη κάθε κόμβου διακρίνεται σε *τοπική* που περιέχει δεδομένα ιδιωτικά του συγκεκριμένου κόμβου και *ολική* που περιέχει *καθαρές* σελίδες των υπόλοιπων κόμβων. Το ποσοστό της μνήμης που δίνεται σαν ολική καθορίζεται δυναμικά ανάλογα με τις ανάγκες τόσο του τοπικού όσο και των υπολοίπων κόμβων. Τα αποτελέσματά τους δείχνουν ότι μνημοβόρες εφαρμογές μπορούν να επιταχυνθούν ως 3.5 φορές.

Το σύστημα αυτό, που ονομάζεται GMS, έχει κάποιες ομοιότητες με το σύστημά μας με την έννοια ότι χρησιμοποιεί την απομακρυσμένη μνήμη για να αποφύγει



Σχήμα 5.6: **Η τροποποίηση στο σύστημα διαχείρισης μνήμης του DEC OSF/1.**

την προσπέλαση στο δίσκο. Η απαίτηση αποθήκευσης στην απομακρυσμένη μνήμη μόνο καθαρών σελίδων απλοποιεί τη σχεδίαση και δε θέτει προβλήματα αξιοπιστίας, όμως η χρήση του για αυτό το λόγο κατά την εκτέλεση μνημοβόρων εφαρμογών περιορίζεται από την ταχύτητα του τοπικού δίσκου, όπου αναγκαστικά αποθηκεύονται οι σελίδες που περιέχουν τα δεδομένα που τροποποιούνται από την εφαρμογή. Οπως αποδεικνύεται και στην ενότητα 4.5, η σχεδίαση αυτή δίνει καλά αποτελέσματα όταν ο ρυθμός μεταγωγής δεδομένων του δίσκου για εγγραφές είναι παραπλήσιος του ρυθμού μεταγωγής δεδομένων του δικτύου. Ομως, όσο αυξάνεται ο ρυθμός μεταγωγής δεδομένων του δικτύου που χρησιμοποιείται τόσο η απόδοση του συστήματος αυτού περιορίζεται από τον τοπικό δίσκο.

5.7 Κατανεμημένα Συστήματα Αρχείων

5.7.1 xFS

Το xFS είναι ένα κατανεμημένο σύστημα αρχείων στο οποίο δεν υπάρχει κάποιος κεντρικός εξυπηρετητής [ADN+95]. Αντίθετα, όλα τα μηχανήματα του δικτύου χρησιμοποιούνται για να αποθηκεύσουν τμήματα του συστήματος αρχείων. Η ολική μνήμη όλων των μηχανών χρησιμοποιείται όποτε είναι δυνατό ώστε να αποφεύγεται η πρόσβαση στο δίσκο. Ετσι μια αίτηση ανάγνωσης δε θα ικανοποιηθεί από το

δίσκο παρά μόνο αν τα ζητούμενα δεδομένα δεν υπάρχουν στην κύρια μνήμη κανενός μηχανήματος του δικτύου. Επίσης το τοπικό δίκτυο χρησιμοποιείται προκειμένου να μεταφέρονται δεδομένα προς/από τους δίσκους παράλληλα χρησιμοποιώντας τεχνικές ισοτιμίας για να εξασφαλιστεί η ομαλή λειτουργία του συστήματος σε περίπτωση βλάβης κάποιου δίσκου.

5.7.2 PACA

Το PACA είναι ένα άλλο σύστημα αρχείων το οποίο έχει υλοποιηθεί για χρήση από πολυεπεξεργαστές [CGL95]. Η μνήμη όλων των επεξεργαστών αντιμετωπίζεται σαν κρυφή μνήμη για το δίσκο, προκειμένου οι λειτουργίες εισόδου/εξόδου να μην αποτελούν τον κρίσιμο παράγοντα ο οποίος καθυστερεί την εκτέλεση παράλληλων προγραμμάτων.

5.7.3 Σύγκριση

Τα κατανεμημένα συστήματα αρχείων έχουν κάποια σχέση με το σύστημά μας, αφού χρησιμοποιούν τη μνήμη όλων των κόμβων του δικτύου ή όλων των επεξεργαστών της πολυεπεξεργαστικής μηχανής προκειμένου να αποφύγουν την προπέλαση στο δίσκο. Οι κύριοι στόχοι των συστημάτων αυτών είναι αφενός να βελτιώσουν την απόδοση των λειτουργιών εισόδου/εξόδου χρησιμοποιώντας δίκτυα μεγάλου ρυθμού μεταγωγής δεδομένων για να μεταφέρουν τα δεδομένα και αφετέρου να πάψουν να βασίζονται σε ένα κεντρικό εξυπηρετητή ο οποίος μπορεί να αποδειχθεί καταλυτικός για την απόδοση του συστήματος όσο ο αριθμός των πελατών αυξάνει.

Παρόλο που οι βελτιώσεις στο σύστημα αρχείων μπορούν να οδηγήσουν και σε βελτίωση της απόδοσης του συστήματος μεταφοράς σελίδων, οι λύσεις που εφαρμόζονται στα συστήματα αρχείων είναι πολύ γενικές για να εφαρμοστούν ως έχουν σε συστήματα χρήσης της απομακρυσμένης μνήμης για αποθήκευση σελίδων. Στα συστήματα αρχείων, οι διεργασίες–πελάτες μοιράζονται αρχεία, γεγονός που οδηγεί σε λύσεις που έχουν σαν κύριο χαρακτηριστικό τη χρήση της κοινής μνήμης σαν κρυφή μνήμη για το δίσκο. Αντίθετα, όσον αφορά την αποθήκευση σελίδων μνήμης, δεν υπάρχουν κοινά δεδομένα. Ετσι, πολιτικές που αναπτύχθηκαν για τη βελτιστοποίηση της πρόσβασης των πελατών σε δεδομένα που τα διαχειρίζεται ένας εξυπηρετητής και που διευκολύνουν το μοίρασμα δεδομένων ανάμεσα στους πελάτες δεν είναι κατ' ανάγκη εφαρμόσιμες σε συστήματα αποθήκευσης σελίδων μνήμης, όπου δεν υπάρχουν κεντρικοί εξυπηρετητές και όπου τα δεδομένα κάθε πελάτη είναι ιδιωτικά.

5.8 Χρήση της απομακρυσμένης μνήμης σε πολυεπεξεργαστές

Στο πανεπιστήμιο Princeton έχει αναπτυχθεί ένα σύστημα χρήσης της απομακρυσμένης μνήμης από πολυεπεξεργαστές [ILP93]. Η ιεραρχία μνήμης έχει επεκταθεί με ένα επιπλέον επίπεδο μνήμης, όπως φαίνεται στο σχήμα 5.7. Οι κόμβοι του πολυεπεξεργαστή διακρίνονται σε κόμβους υπολογισμού και κόμβους που χρησιμοποιούνται αποκλειστικά για την αποθήκευση σελίδων μνήμης. Μ' αυτό τον τρόπο γίνεται δυνατή η εκτέλεση παράλληλων προγραμμάτων χωρίζοντας τον υπολογισμό που πρέπει να επιτελέσουν σε επεξεργαστές χωρίς αναγκαστικά η κύρια μνήμη κάθε κόμβου να επαρκεί για να αποθηκεύσει τα δεδομένα που απαιτούνται. Η χρήση των κόμβων ενός μόνο υπολογιστή εγγυάται την αξιοπιστία του συστήματος χωρίς να είναι απαραίτητη

η χρήση κάποιας πολιτικής αξιοπιστίας, ενώ το δίκτυο διασύνδεσης των κόμβων του επεξεργαστή αναλαμβάνει την ταχύτατη μεταφορά σελίδων.

Γενικά η εργασία αυτή ακολουθεί την ίδια φιλοσοφία με το σύστημά μας προσαρμοσμένη στις ανάγκες ενός πολυεπεξεργαστή. Βέβαια οι κόμβοι εξυπηρετητές είναι εξαρχής καθορισμένοι και δεν αλλάζουν δυναμικά, κάτι το οποίο μπορεί να είναι ανεφάρμοστο σε ένα δίκτυο σταθμών εργασίας. Ομως για τα περισσότερα ίσως παράλληλα προγράμματα κάτι τέτοιο δεν αποτελεί σημαντικό μειονέκτημα σε μια πολυεπεξεργαστική μηχανή.

5.9 RAIDs

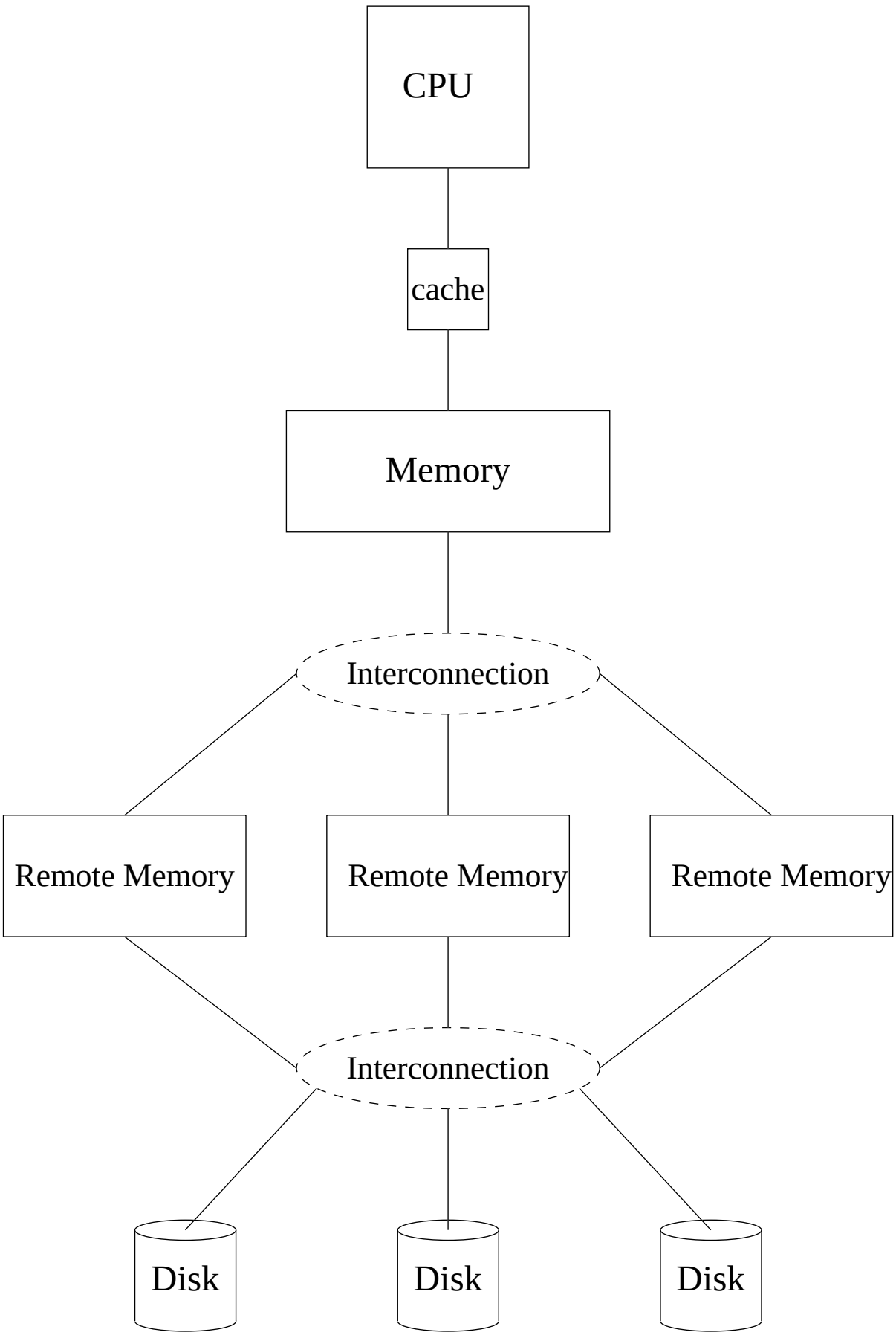
Τα RAIDs είναι συστήματα τα οποία αποτελούνται από πολλούς δίσκους στους οποίους αποθηκεύονται παράλληλα δεδομένα [CLG+94]. Η χρησιμοποίηση πολλών δίσκων παράλληλα αυξάνει σημαντικά το ρυθμό μεταγωγής δεδομένων. Ταυτόχρονα όμως η πιθανότητα να παρουσιάσει βλάβη κάποιος από τους δίσκους είναι σημαντική. Γι αυτό σχεδιάστηκαν και υλοποιήθηκαν σε υλικό διάφορες πολιτικές αξιοπιστίας, παραλλαγές της πολιτικής ισοτιμίας. Από τα RAIDs λοιπόν προέρχεται η πολιτική της ισοτιμίας στην οποία βασίζεται η πολιτική της μεταβλητής ισοτιμίας που προτείνεται στην εργασία αυτή.

5.10 Συστήματα Αρχείων που Βασίζονται σε Ημερολόγιο

Τα συστήματα αρχείων που βασίζονται σε ημερολόγιο δουλεύουν με τέτοιο τρόπο ώστε όλες οι εγγραφές στα αρχεία να γίνονται σε συνεχόμενα μπλοκ του δίσκου [RO92, SBMS93]. Με αυτόν τον τρόπο αποφεύγεται η καθυστέρηση αναζήτησης του δίσκου κατά τις εγγραφές, άρα αυξάνεται ο πραγματικός ρυθμός μεταγωγής δεδομένων του δίσκου. Αυτού του τύπου τα συστήματα αρχείων δημιουργούν έναν αποθηκευτικό χώρο κατατμημένο ο οποίος χρειάζεται να καθαριστεί συνενώνοντας τμήματα που περιέχουν ενεργά δεδομένα, όπως συμβαίνει και με τη μέθοδο της μεταβλητής ισοτιμίας που χρησιμοποιούμε.

Παρόλο που οι γενικές ιδέες είναι παρόμοιες, υπάρχουν σημαντικές διαφορές όσον αφορά τη διαχείριση του αποθηκευτικού χώρου ανάμεσα στα συστήματα αρχείων βασισμένα σε ημερολόγιο και στη χρήση απομακρυσμένης μνήμης με τη μέθοδο της μεταβλητής ισοτιμίας.

- Η κατάτμηση σε συστήματα αρχείων που βασίζονται σε ημερολόγιο συμβαίνει σε μεγάλα τμήματα αποθηκευτικού χώρου (της τάξης των αρκετών MB), ενώ η κατάτμηση στην απομακρυσμένη μνήμη με τη χρήση μεταβλητής ισοτιμίας συμβαίνει σε μικρά σύνολα ισοτιμίας τα οποία αποτελούνται από λίγες σελίδες.
- Τα συστήματα απομακρυσμένης μνήμης με τη χρήση μεταβλητής ισοτιμίας μπορούν να επαναχρησιμοποιήσουν σύνολα ισοτιμίας μόλις αυτά ελευθερωθούν, ενώ τα συστήματα αρχείων που βασίζονται σε ημερολόγιο δεν μπορούν να επαναχρησιμοποιήσουν άμεσα τα μπλοκ του δίσκου που ελευθερώνονται αφού κάτι τέτοιο θα προκαλούσε μετακίνηση της κεφαλής του δίσκου.
- Το καθάρισμα αποθηκευτικού χώρου με τη συνένωση ενεργών τμημάτων του συμβαίνει πολύ πιο σπάνια στα συστήματα αρχείων από ότι στα συστήματα απομακρυσμένης μνήμης, άρα θα πρέπει να γίνεται πολύ πιο αποδοτικά.



Σχήμα 5.7: **Η ιεραρχία μνήμης ενός πολυεπεξεργαστή.**

- Ο σκοπός ενός συστήματος αρχείων που βασίζεται σε ημερολόγιο είναι η ελαχιστοποίηση της κίνησης της κεφαλής του δίσκου, ενώ ο σκοπός ενός συστήματος απομακρυσμένης μνήμης είναι η ελαχιστοποίηση του αριθμού των σελίδων που μεταφέρονται.

Για τους παραπάνω λόγους οι μέθοδοι που έχουν αναπτυχθεί για συστήματα αρχείων βασισμένα σε ημερολόγιο δεν εφαρμόζονται αναγκαστικά όπως έχουν σε συστήματα απομακρυσμένης μνήμης.

5.11 Zebra

Η Zebra [HO93] είναι ένα σύστημα αρχείων δικτύου, το οποίο μοιράζει τα δεδομένα του σε πολλούς εξυπηρετητές προκειμένου να αυξήσει το ρυθμό μεταγωγής τους. Χρησιμοποιώντας αρκετούς εξυπηρετητές καταφέρνει η απόδοσή του να περιορίζεται μόνο από το ρυθμό μεταγωγής δεδομένων του τοπικού δικτύου. Για να μπορεί να λειτουργεί και μετά από τη βλάβη κάποιου εξυπηρετητή, χρησιμοποιεί την τεχνική της ισοτιμίας αποφεύγοντας όμως την επιπλέον επιβάρυνση που προκαλεί η μέθοδος αυτή κατά την εγγραφή δεδομένων αφού το σύστημα αρχείων είναι βασισμένο σε ημερολόγιο και έτσι όλες οι εγγραφές γίνονται σειριακά. Ενας κεντρικός διαχειριστής αρχείων είναι υπεύθυνος για την καταγραφή της πληροφορίας σχετικά με τη θέση κάθε μπλοκ αρχείου του συστήματος στο δίκτυο και την ενημέρωση των πελατών.

Γενικά η Zebra χρησιμοποιεί παρόμοιες ιδέες με το σύστημά μας, εφαρμοσμένες στα συστήματα αρχείων (κατανομή δεδομένων σε εξυπηρετητές, υπολογισμός πλεονάζουσας πληροφορίας). Βέβαια η πολυπλοκότητα της Zebra είναι αρκετά μεγαλύτερη εξαιτίας της ανάγκης διατήρησης των δεδομένων σε συνεπή κατάσταση κατά την ταυτόχρονη πρόσβαση σε κοινά δεδομένα από πολλούς πελάτες, ενώ στο δικό μας σύστημα τα δεδομένα που προσπελαύνει κάθε πελάτης είναι ιδιωτικά. Επιπλέον, το καθάρισμα του αποθηκευτικού χώρου είναι αρκετά πιο πολύπλοκο και χρονοβόρο, όπως σε όλα τα συστήματα αρχείων που βασίζονται σε ημερολόγιο.

5.12 Συστήματα Κατανεμημένης Μοιραζόμενης Μνήμης

Η εργασία μας έχει κάποια ομοιότητα με τα συστήματα κατανεμημένης μοιραζόμενης μνήμης [Del88, LH89] μια που και στις δυο περιπτώσεις χρησιμοποιείται η απομακρυσμένη μνήμη για την αποθήκευση σελίδων μνήμης εφαρμογών. Η κύρια διαφορά μας συνίσταται στο γεγονός ότι η δική μας εργασία επικεντρώνεται σε *σειριακές* εφαρμογές όπου οι σελίδες δε μοιράζονται ενώ τα συστήματα κατανεμημένης μοιραζόμενης μνήμης επικεντρώνονται σε *παράλληλες* εφαρμογές στις οποίες ο κύριος στόχος είναι η μείωση της επιβάρυνσης που προκαλείται από το μοίρασμα σελίδων.

Κεφάλαιο 6

Συμπεράσματα

Στην εργασία αυτή εξετάζουμε τη χρήση της απομακρυσμένης μνήμης για την αποθήκευση σελίδων μνήμης. Περιγράφουμε το πρωτότυπο που υλοποιήθηκε σαν οδηγός συσκευής πάνω από το DEC OSF/1 χωρίς να αλλαχθεί ούτε μια γραμμή από τον κώδικα του λειτουργικού συστήματος. Χρησιμοποιώντας πραγματικές εφαρμογές εξετάσαμε την απόδοση του συστήματος σε ένα τοπικό δίκτυο. Γενικά στην εργασία αυτή

- Περιγράφουμε τη σχεδίαση και υλοποίηση ενός συστήματος χρήσης της απομακρυσμένης μνήμης για αποθήκευση σελίδων χρησιμοποιώντας μια νέα τεχνική, την τεχνική της *μεταβλητής ισοτιμίας* προκειμένου να εξασφαλίσουμε την απρόσκοπτη λειτουργία του σε περίπτωση βλάβης κάποιας μηχανής του δικτύου.
- Περιγράφουμε την υλοποίηση ενός *προσαρμοζόμενου* σελιδοδιαχειριστή, ο οποίος μπορεί να επιλέγει δυναμικά ανάμεσα στις διαθέσιμες συσκευές αποθήκευσης σελίδων (συμπεριλαμβανομένης και της απομακρυσμένης μνήμης) ώστε να μεγιστοποιεί την απόδοσή του.
- Αποδεικνύουμε ότι η αξιόπιστη αποθήκευση σελίδων στην απομακρυσμένη μνήμη δίνει σημαντική βελτίωση στο χρόνο εκτέλεσης των εφαρμογών σε σχέση με τον τοπικό δίσκο.

Βασισμένοι στην υλοποίησή μας και στα πειραματικά αποτελέσματα που προέκυψαν, μπορούμε να συμπεράνουμε τα εξής

- *Η χρήση της απομακρυσμένης μνήμης για την αποθήκευση σελίδων μνήμης δίνει καλύτερη απόδοση από τη χρήση του τοπικού δίσκου.* Οι εφαρμογές που χρησιμοποιούν το σύστημά μας δίνουν ως και 60% καλύτερα αποτελέσματα ακόμη και πάνω από δίκτυα χαμηλού ρυθμού μεταγωγής δεδομένων, όπως το Ethernet. Χρησιμοποιώντας δίκτυα μεγαλύτερου ρυθμού μεταγωγής δεδομένων μπορούν να επιτευχθούν ακόμη καλύτερα αποτελέσματα.
- *Η χρήση της απομακρυσμένης μνήμης είναι ένας φθηνός τρόπος με τον οποίο οι εφαρμογές μπορούν να χρησιμοποιούν περισσότερη μνήμη από όση μπορεί να δώσει ένας μόνο υπολογιστής.* Η απομακρυσμένη μνήμη δίνει καλά αποτελέσματα χωρίς να απαιτεί επιπλέον υλικό. Ο μόνος τρόπος να επιτευχθεί κάτι τέτοιο χρησιμοποιώντας δίσκους είναι η χρήση ακριβών δίσκων RAIDs. Επιπλέον το ποσό της διαθέσιμης μνήμης αυξάνει όσο προσθέτονται περισσότερα μηχανήματα στο τοπικό δίκτυο.
- *Η εξασφάλιση αξιοπιστίας είναι πολύ φθηνή.* Η τεχνική της μεταβλητής ισοτιμίας προσφέρει αξιοπιστία με πολύ μικρή επιβάρυνση σε χρόνο εκτέλεσης και μνήμη.

Η απόδοσή της είναι ελάχιστα χειρότερη από την απόδοση της μη χρήσης αξιοπιστίας και πολύ καλύτερη από αυτήν του τοπικού δίσκου.

- *Η δυνατότητα της δυναμικής αλλαγής της συσκευής που χρησιμοποιείται για την αποθήκευση σελίδων μπορεί να βελτιώσει κατά μεγάλο βαθμό το χρόνο εκτέλεσης των εφαρμογών.* Η απόδοση των διαφόρων τεχνικών αξιοπιστίας είναι άμεση συνάρτηση του ρυθμού μεταγωγής δεδομένων που είναι σε θέση να παρέχουν το δίκτυο και ο τοπικός δίσκος. Αφού σε ένα περιβάλλον τοπικού δικτύου οι δυο αυτοί παράγοντες μεταβάλλονται συνεχώς, η αδυναμία προσαρμογής στις νέες συνθήκες και επιλογής της "καλύτερης" κάθε φορά πολιτικής μπορεί να οδηγήσει σε σημαντική επιβάρυνση του χρόνου εκτέλεσης.

- *Τα οφέλη από τη χρήση της απομακρυσμένης μνήμης για την αποθήκευση σελίδων θα αυξάνουν με το πέρασμα του χρόνου.* Οι σημερινές τάσεις στην αρχιτεκτονική υπο–λογιστών δείχνουν ότι ο χρόνος προσπέλασης του δίσκου μετρημένος σε κύκλους του επεξεργαστή συνεχίζει να αυξάνει με ταχείς ρυθμούς. Οι δίσκοι λοιπόν δεν αναμένεται να είναι σε θέση να προσφέρουν το ρυθμό μεταγωγής δεδομένων που απαιτείται για την αποθήκευση σελίδων μνήμης. Αντίθετα τα δίκτυα διασύνδεσης προσφέρουν ένα ολοένα αυξανόμενο ρυθμό μεταγωγής δεδομένων αυξάνοντας έτσι το πλεονέκτημά τους έναντι του δίσκου.

Σύμφωνα με τα πειραματικά μας αποτελέσματα, η χρήση της απομακρυσμένης μνήμης είναι ένας αποδοτικός και φθηνός τρόπος εκτέλεσης μνημοβόρων εφαρμογών σε ένα τοπικό δίκτυο υπολογιστών.

Παράρτημα Α

Ορολογία

ανενεργός	idle
ανενεργός	inactive
αξιοπιστία	reliability
απεικονισμένος στη μνήμη	memory mapped
απομακρυσμένη ανάγνωση	remote read
απομακρυσμένη εγγραφή	remote write
απομακρυσμένη κλήση διαδικασίας	remote procedure call
απομακρυσμένη μνήμη	remote memory
γέφυρα	bridge
διαφανής	transparent
διασύνδεση	interface
δίαυλος	bus
διεγγραφή	write through
διεργασία	process
έγκυρος	valid
ενεργός	busy
ενταμιευτής	buffer
ενταμιευτής εγγραφής	write buffer
εξυπηρετητής	server
εξυπηρετητής ισοτιμίας	parity server
επεκταμένος	extended
επιβεβαίωση	acknowlegement
ημερολόγιο	log
ιδεατή μνήμη	virtual memory
ισοτιμία	parity
ίχνος	track
κατανεμημένος	distributed
κατάρρευση	collapse
κατάσταση πυρήνα	kernel mode
κατάσταση χρήστη	user mode
κατάτμηση	fragmentation
κλήση συστήματος	system call
κρέμασμα	crash
κρυφή μνήμη	cache
λάθος σελίδας	page fault

λειτουργικό σύστημα	operating system
λέξη	word
λογισμικό	software
μεταβλητή ισοτιμία	parity logging
μεταγλώττιση	compilation
μεταφορά	porting
μεταφέρσιμος	portable
μοιραζόμενος	shared
νήμα	thread
οδηγός συσκευής	device driver
παλινδρόμηση	thrashing
πελάτης	client
πηγαίος κώδικας	source code
πλακέτα διασύνδεσης δικτύου	network interface
πολυνηματικός	multithreaded
πόρτα	port
πρίζα	socket
προνομιούχος	privileged
πρώιμη μεταφορά	prefetching
πυρήνας	kernel
ρυθμός μεταγωγής δεδομένων	bandwidth
σελιδοδιαχειριστής	pager
στρώμα διασύνδεσης δεδομένων	data link layer
στρώμα πρίζας	socket layer
σύγκρουση	collision
σύνδεση	linking
σύνολο εργασίας	working set
συσκευή μπλοκ	block device
συσκευή χαρακτήρων	character device
σύστημα αρχείων	file system
σύστημα αρχείων δικτύου	network file system
τεχνική πολλαπλών αντιγράφων	mirroring
τμήμα δίσκου	disk partition
τομέας	sector
τοπικό δίκτυο	local area network
τυχαία πρόσβαση	random access
υλικό	hardware
υπερχρήστης	super user
φόρτος μνήμης	memory load
χρόνος αναζήτησης	seek time
χρόνος περιστροφής	rotational time
χρόνος συστήματος	system time
χρόνος χρήστη	user time
χώρος διευθύνσεων	address space

Βιβλιογραφία

[ABB+86] Mike Accetta, Robert Baron, William Bolosky, David Golub, Richard Rashid, Avadis Tevanian, and Michael Young. Mach: A New Kernel Foundation for UNIX Development. In *Proceedings of the 1986 Summer Usenix Conference*, Atlanta, GA, USA, June 1986.

[ACP95] Thomas E. Anderson, David E. Culler, and David A. Patterson. A Case for NOW (Networks of Workstations). *IEEE Micro*, 15(1):54−−64, February 1995.

[ADN+95] Thomas E. Anderson, Michael D. Dahlin, Jeanna M. Neefe, David A. Patterson, Drew S. Roseli, and Randolph Y. Wang. Serverless Network File Systems. In *Proceedings of the 15th ACM Symposium on Operating System Principles*, pages 109−−126, Copper Mountain Resort, Colorado, USA, December 1995.

[Ame87] American National Standards Institute. *FDDI Media Access Control. American National Standard X3.139*, 1987.

[AN95] Eric A. Anderson and Jeanna M. Neefe. An Exploration of Network RAM. CS−252 class project, University of California at Berkeley, February 1995.

[Bac86] Maurice J. Bach. *The Design of the Unix Operating System*. Prentice−Hall, Inc, 1986.

[BCF+95] Nanette J. Boden, Danny Cohen, Robert E. Felderman, Alan E. Kulawic, Charles L. Seitz, Jakov N. Seizovic, and Wen−King Su. Myrinet − A Gigabit−per−Second Local−Area Network. *IEEE Micro*, 15(1):29−−36, February 1995.

[BJMW95] Greg Buzzard, David Jacobson, Scott Marovich, and John Wilkes. Hamlyn: a High−Performance Network Interface with Sender−Based Memory Management. In *Proceedings of the Hot Interconnects III Symposium*, August 1995.

[BLA+94] Matthias A. Blumrich, Kai Li, Richard Alpert, Cezary Dubnicki, Edward W. Felten, and Jonathan Sandberg. Virtual Memory Mapped Network Interface for the SHRIMP Multicomputer. In *Proceedings of the 21st International Symposium on Computer Architecture*, pages 142−−153, Chicago, IL, USA, April 1994.

[CG90] Douglas Comer and James Griffoen. A New Design for Distributed Systems: The Remote Memory Model. In *Proceedings of the 1990 Summer Usenix Conference*, pages 127−−135, Anaheim, CA, USA, June 1990.

[CGL95] Toni Cortes, Sergi Girona, and Jesus Labarta. PACA: A Distributed File System Cache for Parallel Machines. Performance under Unix−like Workload. Tech report UPC−DAC−1995−20, Universitat Politechnica de Catalunya, Department d' Arquitectura de Computadors, June 1995.

[CLG+94] Peter M. Chen, Edward K. Lee, Garth A. Gibson, Randy H. Katz, and David A. Patterson. RAID: High−Performance, Reliable Secondary Storage. *ACM Computing Surveys*, 26(2):145−−185, June 1994.

[Com91] Douglas Comer. *Internetworking with TCP/IP: Principles, Protocols and Architecture*. Prentice−Hall, Inc, 1991.

[Del88] G. Delp. *The Architecture and Implementation of Memnet: A High−Speed Shared Memory Computer Communication Network*. PhD thesis, University of Delaware, 1988.

[Dig93a] Digital Equipment Corporation. *DEC 3000 300/400/500/600/800 Models System Programmer's Manual*, 1993.

[Dig93b] Digital Equipment Corporation. *Guide to Writing Device Drivers: Volume 2 Reference*, 1993.

[DM96] George Dramitinos and Evangelos P. Markatos. Reliable Paging to Remote Main Memory in a Workstation Cluster. In R.H. Campbell and Nayeem Islam, editors, *Modern Operating Systems*. IEEE Computer Society Press, 1996.

[Dol94] Dolphin Interconnect Solutions AS, Oslo, Norway. *DIS301 Sbus−to−SCI Adapter User's Guide*, April 1994. e−mail scimktg@dolphin.no.

[FMP+95] Michael J. Feeley, William E. Morgan, Frederic H. Pighin, Anna R. Karlin, Henry M. Levy, and Chandramohan A. Thekkath. Implementing Global Memory Management in a Workstation Cluster. In *Proceedings of the 15th ACM Symposium on Operating System Principles*, pages 201−−212, Copper Mountain Resort, Colorado, USA, December 1995.

[FZ91] Edward W. Felten and John Zahorjan. Issues in the Implementation of a Remote Memory Paging System. Tech report TR 91−03−09, University of Washington, Department of Computer Science and Engineering, March 1991.

[Gal78] Robert G. Gallager. Variations on a Theme of Huffman. *IEEE Transactions on Information Theory*, 24(6):668−−674, November 1978.

[Gil95] R. Gillet. Memory Channel. In *Proceedings of the Hot Interconnects III Symposium*, August 1995.

[HO93] John H. Hartman and John K. Ousterhout. The Zebra Striped Network File System. In *Proceedings of the 14th ACM Symposium on Operating System Principles*, pages 29−−43, Asheville, NC, USA, December 1993.

[HP90] John L. Hennessy and David A. Patterson. *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann Publishers, Inc., 1990.

[IEE85a] IEEE. *802.3: Carrier Sense Multiple Access with Collision Detection*, 1985.

[IEE85b] IEEE. *802.5: Token Ring Access Method*, 1985.

[IEE87] IEEE. *VME Bus. IEEE Standard 1014*, 1987.

[ILP93] Liviu Iftode, Kai Li, and Karin Petersen. Memory Servers for Multicomputers. In *Proceedings of the COMPCON 93*, 1993.

[Int86] Intel Corporation. *Intel 80386 Programmer's Reference Manual*, 1986.

[LH89] Kai Li and Paul Hudak. Memory Coherence in Shared Virtual Memory Systems. *ACM Transactions on Computer Systems*, 7(4):321−−359, November 1989.

[LMKQ89] Samuel J. Leffler, Marshall Kirk McKusick, Micheal J. Karels, and John S. Quarterman. *The Design and Implementation of the 4.3BSD UNIX Operating System*. Addison−Wesley Publishing Company, Inc., 1989.

[LP91] Kai Li and Karin Petersen. Evaluation of Memory System Extensions. In *Proceedings of the 18th International Symposium on Computer Architecture*, pages 84−−93, Toronto, Canada, May 1991.

[MD95] Evangelos P. Markatos and George Dramitinos. Adding Flexibility to a Remote Memory Pager. In *Proceedings of the Fourth International Workshop on Object Orientation in Operating Systems*, pages 183−−186, Lund, Sweeden, August 1995.

[MD96] Evangelos P. Markatos and George Dramitinos. Implementation of a Reliable Remote Memory Pager. In *Proceedings of the 1996 Usenix Technical Conference*, pages 177−−190, San Diego, CA, USA, January 1996.

[MK96] Evangelos P. Markatos and Manolis G.H. Katevenis. Telegraphos: High−Performance Networking for Parallel Processing on Workstation Clusters. In *Proceedings of the Second International Symposium on High−Performance Computer Architecture (HPCA)*, San Jose, CA, USA, February 1996.

[MYW94] Alan Mainwaring, Chad Yoshikawa, and Kristin Wright. NOW White Paper: Network RAM Prototype, November 1994. University of California at Berkeley. http://now.cs.berkeley.edu/Nram/network−ram.html.

[New94] Peter Newman. ATM Local Area Networks. *IEEE Communications Magazine*, 32(3):86−−98, March 1994.

[Pos81a] J. Postel. *Internet Protocol−DARPA Internet Program Protocol Specification. Internet Network Working Group, RFC 791*. USC Information Sciences Institute, September 1981.

[Pos81b] J. Postel. *Transmission Control Protocol. Internet Network Working Group, RFC 793*. USC Information Sciences Institute, September 1981.

[RO92] Mendel Rosenblum and John K. Ousterhout. The Design and Implementation of a Log−Structured File System. *ACM Transactions on Computer Systems*, 10(1):26−−52, February 1992.

[SB90] Michael D. Schroeder and Michael Burrows. Performance of Firefly RPC. *ACM Transactions on Computer Systems*, 8(1):1−−17, February 1990.

[SBMS93] M. Seltzer, K. Bostic, M. K. McKusick, and C. Staelin. An Implementation of a Log−Structured File System for UNIX. In *Proceedings of the 1993 Winter Usenix Conference*, pages 307−−326, San Diego, CA, USA, January 1993.

[SD91] Bill N. Schilit and Dan Duchamp. Adaptive Remote Paging for Mobile Computers. Technical Report CUCS−004−91, Columbia University, Department of Computer Science, February 1991.

[SGK+85] R. Sandberg, D. Goldberg, S. Kleiman, D. Walsh, and B. Lyon. Design and Implementation of the Sun Network File System. In *Proceedings of the 1985 Summer Usenix Conference*, Portland, OR, USA, June 1985.

[Sit92] Richard L. Sites, editor. *Alpha Architecture Reference Manual*. Digital Press, 1992.

[SL91] Dimitrios N. Serpanos and Richard J. Lipton. The Design of a High−Speed Network. In *Proceedings of the Third IFIP WG 6.4 Conference on High Speed Networking*, Berlin, Germany, May 1991.

[SPA92] SPARC International. *The SPARC Architecture Version 9*, 1992.

[Sun92] SunSoft Inc. *SunOS 5.3 Writing Device Drivers*, 1992.

[Tan89] Andrew S. Tanenbaum. *Computer Networks*, chapter 3, page 128. Prentice−Hall, Inc, 1989.

[TSJ88] Charles P. Thacker, Lawrence C. Stewart, and Edward H. Satterthwaite Jr. Firefly: A Multiprocessor Workstation. *IEEE Transactions on Computers*, 37(8):909−−920, August 1988.

[Wel84] Terry A. Welch. A Technique for High Performance Data Compression. *IEEE Computer*, 17(6):8−−19, June 1984.

[WZ95] Michael Wu and Willy Zwaenepoel. eNVy: A Non−Volatile, Main Memory Storage System. In *Proceedings of the 6th Symposium on Architectural Support for Programming Languages and Operating Systems*, 1995.

[ZL77] Jacob Ziv and Abraham Lempel. A Universal Algorithm for Sequential Data Compression. *IEEE Transactions on Information Theory*, 23(3):337−−343, May 1977.