ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

Προσομοίωση Αρχιτεκτονικής Κοινόχρηστων Δίσκων για Συστήματα Επεξεργασίας Δοσοληψιών

Αρταβάνης Μιχαήλ-Ιωάννης

Μεταπτυχιακή Εργασία

Ηράκλειο, Οκτώβριος 1997

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

Προσομοίωση Αρχιτεκτονικής Κοινόχρηστων Δίσκων για Συστήματα Επεξεργασίας Δοσοληψιών

Εργασία που υποβλήθηκε από τον
Μιχαήλ-Ιωάννη Αρταβάνη
ως μερική εκπλήρωση των απαιτήσεων
για την απόκτηση
ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΔΙΠΛΩΜΑΤΟΣ ΕΙΔΙΚΕΥΣΗΣ

Συγγραφέας:

Αρταβάνης Μιχαήλ-Ιωάννης
Τμήμα Επιστήμης Υπολογιστών

Εισηγητική Επιτροπή:

Χρήστος Νικολάου, Αναπληρωτής Καθηγητής, Επόπτης

Κατερίνα Χούστη, Αναπληρώτρια Καθηγήτρια, Μέλος

Γεώργιος Σταμούλης, Επίκουρος Καθηγητής, Μέλος

Δεκτή:

Πάνος Κωνσταντόπουλος
Πρόεδρος Επιτροπής Μεταπτυχιακών Σπουδών

Ηράκλειο, Οκτώβριος 1997

Στους γονείς μου, Γιώργο και Φράνκα
και
στον αδερφό μου Ανδρέα

Ευχαριστίες

Θα ήθελα να ευχαριστήσω όλους εκείνους που με βοήθησαν και με στήριξαν προκειμένου να ολοκληρώσω την εκπόνηση της μεταπτυχιακής μου εργασίας.

Καταρχήν θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου Χρήστο Νικολάου για την καθοδήγησή του και τις πολύτιμες συμβουλές του κατά την διάρκεια της εργασίας αυτής. Ακόμη, θα ήθελα να ευχαριστήσω τον καθηγητή, Γιώργο Σταμούλη και την καθηγήτρια, Κατερίνα Χούστη για την συνεισφορά τους στην τελική διαμόρφωση της εργασίας μου αλλά και για την συμμετοχή τους στην επιτροπή εξέτασης.

Ιδιαίτερα θα ήθελα να ευχαριστήσω, τον αδελφό μου Ανδρέα για τις διορθώσεις που έκανε στο κείμενο, και όχι μόνο, τον Μανόλη Μαραζάκη για τον χρόνο και τις συμβουλές που μου έδωσε χάρις στις οποίες η εργασία αυτή εκπονήθηκε στον δεδομένο χρόνο, τον Σάββα Βελισσάρη για τις στιγμές που περάσαμε μαζί τόσα χρόνια και την συμπαράσταση που μου έδειξε στην αρχική φάση της εργασίας μου, τον Λάμπρο Γκανά για την συμπαράστασή του σε κρίσιμα σημεία της εργασίας, και τέλος όλους τους φίλους μου που αν και δεν είχαν άμεση σχέση με την εργασία μου, με στήριξαν σημαντικά. Επίσης θα ήθελα να ευχαριστήσω όλα τα μέλη της ομάδας των ΠΛΕΙΑΔΩΝ του Ινστιτούτου Πληροφορικής (ΙΠ/ΙΤΕ) αλλά και τους υπόλοιπους μεταπτυχιακούς φοιτητές του τμήματος για την παρέα που κάναμε τόσο καιρό.

Ευχαριστώ τέλος, το Τμήμα Επιστήμης Υπολογιστών του Πανεπιστημίου Κρήτης για την δυνατότητα των μεταπτυχιακών σπουδών που μου έδωσε και το Ινστιτούτο Πληροφορικής του Ιδρύματος Τεχνολογίας και Έρευνας για την υλικοτεχνική υποδομή που μου παρείχε.

Προσομοίωση Αρχιτεκτονικής Κοινόχρηστων Δίσκων για Συστήματα Επεξεργασίας Δοσοληψιών

Μιχάλης Αρταβάνης

Μεταπτυχιακή Εργασία

Τμήμα Επιστήμης Υπολογιστών
Πανεπιστήμιο Κρήτης

ΠΕΡΙΛΗΨΗ

Τα συστήματα επεξεργασίας δοσοληψιών όπως είναι π.χ. το τραπεζικό σύστημα, βασίζονται στην έννοια της δοσοληψίας και στις ιδιότητες που της έχουν αποδοθεί (γνωστές και ως ACID ιδιότητες- Atomicity, Consistency, Isolation, Durability), προκειμένου να υποστηριχθεί η ταυτόχρονη πρόσβαση σε κοινόχρηστα δεδομένα. Η σύζευξη υπολογιστικών κόμβων για την επεξεργασία δοσοληψιών έχει γίνει αντικείμενο μελέτης τα τελευταία χρόνια, για λόγους χωρητικότητας, διαθεσιμότητας και κόστους.

Μία προσέγγιση στον τρόπο σύζευξης υπολογιστικών κόμβων είναι η μέθοδος σύμφωνα με την οποία όλοι οι κόμβοι έχουν άμεση πρόσβαση στα κοινόχρηστα δεδομένα που είναι αποθηκευμένα στους δίσκους. Δύο διαφορετικές υλοποιήσεις υπάρχουν : Στην πρώτη μόνο οι δίσκοι είναι κοινοί (Shared Disks αρχιτεκτονική), ενώ στη δεύτερη υπάρχει επιπλέον και μία κοινόχρηστη ενδιάμεση μνήμη (Shared Intermediate Memory αρχιτεκτονική).

Στην παρούσα εργασία έγινε η υλοποίηση των δύο αυτών αρχιτεκτονικών σύζευξης υπολογιστικών κόμβων επεκτείνοντας τον προσομοιωτή συστημάτων επεξεργασίας δοσοληψιών, TPsim. Γι' αυτές τις αρχιτεκτονικές έγινε η υλοποίηση και η αξιολόγηση ενός νέου αλγορίθμου δρομολόγησης δοσοληψιών. Δίνεται αρχικά μία αναλυτική περιγραφή του αλγορίθμου και στη συνέχεια κάποια πειραματικά αποτελέσματα.

Τέλος έγινε μία μελέτη της επίδρασης αλγορίθμων ομαδοποίησης φόρτου εργασίας στην απόδοση αλγορίθμων δρομολόγησης. Η μελέτη αυτή έγινε με την βοήθεια ενός εργαλείου ομαδοποίησης φόρτου εργασίας, του CLUE και η μελέτη έγινε για την Shared Nothing αρχιτεκτονική σύζευξης υπολογιστικών κόμβων. Τρεις αλγόριθμοι ομαδοποίησης φόρτου εργασίας δοκιμάστηκαν σε συνδυασμό με τρεις αλγόριθμους δρομολόγησης.

Επόπτης : Χρήστος Νικολάου, Αναπληρωτής Καθηγητής

Simulation of the Shared Disks Architecture for Transaction Processing Systems

Michalis Artavanis

Master of Science Thesis

Department of Computer Science
University of Crete

ABSTRACT

Coupling multiple computing nodes for transaction processing for the sake of capacity, availability and cost, has received considerable interests. There are several different approaches for coupling. One method of coupling multiple node is the data sharing approach, where all coupled nodes have direct access to shared data stored on disks. Two alternative implementations of this approach are considered in this work. In the first case, only disks are shared (Shared Disk architecture) while in the second case, a shared intermediate level of memory is introduced on top of the Shared Disk architecture. This extension is referred to as the Shared Intermediate Memory architecture. Both were implemented on TPsim which is a simulator for parallel and distributed transaction processing systems.

For these architectures, a new efficient routing algorithm was implemented. This algorithm is explained in detail and then we show some experimental results of its use.

Finally a series of simulations were made, to observe the impact of workload clustering on the performance of some routing algorithms. For this purpose a clustering tool was used (CLUE), along with the Shared Nothing architecture which is also supported by the TPsim simulator. Three workload clustering algorithms weren considered along with three transaction routing algorithms.

Supervisor : Christos Nikolaou, Associate Professor

Περιεχόμενα

Περιεχόμενα **iii**

Κατάλογος Σχημάτων **vii**

Κατάλογος Πινάκων **ix**

1 Εισαγωγή **1**

1.1 Ιστορική αναδρομή 1

1.2 Αντικείμενο και συνεισφορά της εργασίας 2

1.3 Οργάνωση της εργασίας 3

2 Επιστημονικό υπόβαθρο **4**

2.1 Η έννοια της δοσοληψίας 4

2.2 Μοντέλα δοσοληψιών 5

2.2.1 Flat δοσοληψίες 6

2.3 Δείκτες επίδοσης (Benchmarks) 7

2.3.1 Επεκτάσεις του flat μοντέλου δοσοληψίας 8

2.4 Κατηγορίες δοσοληψιών με βάση την εκτέλεση τους 10

2.5 Συστήματα επεξεργασίας δοσοληψιών 11

2.6 Μοντέλα υπολογισμού 12

2.7 Επόπτης επεξεργασίας δοσοληψιών 14

2.8 Χαρακτηρισμός φόρτου εργασίας 16

2.9 Στόχοι επίδοσης 18

2.10 Αλγόριθμοι δρομολόγησης 20

2.10.1 Στατικοί αλγόριθμοι 21
2.10.2 Δυναμικοί αλγόριθμοι 21
2.10.3 Αλγόριθμοι προσανατολισμένοι στους στόχους επίδοσης (goal oriented) 22
2.10.4 Μικροοικονομικοί αλγόριθμοι 22

3 Κατανεμημένη επεξεργασία δοσοληψιών 23

3.1 Αρχιτεκτονικές σύζευξης υπολογιστικών κόμβων 24
3.1.1 Αρχιτεκτονική Shared Nothing 25
3.1.2 Αρχιτεκτονική Κοινόχρηστων Δίσκων (Shared Disk) 27
3.1.3 Αρχιτεκτονική Κοινόχρηστης Ενδιάμεσης Μνήμης (Shared Intermediate Memory) 29
3.1.4 Αρχιτεκτονική Shared Everything 31
3.2 Επισκόπηση σχετικής βιβλιογραφίας 32
3.2.1 Άλλες δημοσιεύσεις 35
3.3 Συμπεράσματα επισκόπησης βιβλιογραφίας 36

4 Επέκταση προσομοιωτή TPsim 38

4.1 Ο προσομοιωτής TPsim 38
4.2 Γλώσσα προδιαγραφής 40
4.2.1 Αποθηκευτικά μέσα 41
4.2.2 Υπολογιστικοί κόμβοι 42
4.2.3 Ομάδες κόμβων 45
4.2.4 Κοινόχρηστος ενταμιευτής 47
4.2.5 Βάση Δεδομένων 47
4.2.6 Φόρτος εργασίας 49
4.3 Προσομοίωση 51
4.4 Προσομοίωση υποσυστημάτων υλικού 52
4.5 Προσομοίωση υποσυστημάτων λογισμικού 52
4.5.1 Έλεγχος ταυτόχρονης πρόσβασης 55

5 Δρομολόγηση δοσοληψιών σε αρχιτεκτονικές κοινόχρηστων δεδομένων 62

5.1 Το πρόβλημα 62
5.2 Ο αλγόριθμος δρομολόγησης 63
5.2.1 Υπολογισμός του εκτιμώμενου μέσου χρόνου απόκρισης R 64
5.3 Υπολογισμός του πίνακα Nopt 65
5.3.1 Υπολογισμός της συνάρτησης computeAvgNumOfLocalMisses() 69
5.3.2 Υπολογισμός της συνάρτησης computeAvgNumOfDASDMisses() 71
5.3.3 Υπολογισμός των αναγκών επεξεργασίας 72
5.3.4 Υπολογισμός του βαθμού χρήσης των κόμβων 73
5.4 Πειραματικά αποτελέσματα χρήσης αλγορίθμου δρομολόγησης 73
5.5 Κριτική αλγορίθμου δρομολόγησης 89

6 Επίδραση της ομαδοποίησης δοσοληψιών στην δρομολόγηση δοσοληψιών 94

6.1 Ομαδοποίηση φόρτου εργασίας 94
6.2 Ομαδοποίηση φόρτου εργασίας με χρήση του CLUE 95
6.3 Αξιολόγηση ομαδοποίησης 96
6.4 Επίπτωση της ομαδοποίησης του φόρτου εργασίας στην δρομολόγηση δοσοληψιών 99
6.5 Πειραματικά αποτελέσματα για το *Reduced DOA* αρχείο ιχνών 101
6.6 Πειραματικά αποτελέσματα για το *Reduced PULS* αρχείο ιχνών 105
6.7 Συμπεράσματα 106

7 Συμπεράσματα και μελλοντικές κατευθύνσεις 108

7.1 Αποτελέσματα εργασίας 108
7.2 Μελλοντικές επεκτάσεις και κατευθύνσεις 109
7.2.1 Επεκτάσεις TPsim 109
7.2.2 Άλλες μελλοντικές κατευθύνσεις 110

A Token γλώσσας προδιαγραφής 111

B Γραμματική γλώσσας προδιαγραφής 115

C Αναλυτική περιγραφή αλγορίθμου δρομολόγησης 124

Βιβλιογραφία **128**

Κατάλογος Σχημάτων

2.1 Δυνατοί τρόποι τερματισμού μίας flat δοσοληψίας. 6

2.2 Διάγραμμα καταστάσεων / μεταβάσεων μίας δοσοληψίας. 7

2.3 Κατηγορίες δοσοληψιών με βάση την εκτέλεση τους. 11

2.4 Μοντέλα υπολογισμού 13

2.5 Μοντέλο για τους επόπτες επεξεργασίας δοσοληψιών 15

3.1 Βασικές κατηγορίες στα κατανεμημένα συστήματα επεξεργασίας δοσοληψιών 25

3.2 Αρχιτεκτονική Shared Nothing 26

3.3 Αρχιτεκτονική Shared Disk 29

3.4 Αρχιτεκτονική Shared Intermediate Memory 30

3.5 Αρχιτεκτονική Shared Everything 32

4.1 Βασικά δομικά στοιχεία του προσομοιωτή κατηγορίες 39

5.1 Παράδειγμα συστήματος με έναν κόμβο Front-End και N κόμβους Back-End . . 63

5.2 Η δρομολόγηση των δοσοληψιών για το πείραμα με τον SDR0.25 και την SIM αρχιτεκτονική. 91

5.3 Η δρομολόγησης των δοσοληψιών για το πείραμα με τον SDR0.50 και την SIM αρχιτεκτονική. 92

6.1 Συγκριτικά αποτελέσματα μαζί με τους χρόνους εκτέλεσης για τους τρεις αλγορίθμους ομαδοποίησης και για το αρχείο DOA 98

6.2 Συγκριτικά αποτελέσματα μαζί με τους χρόνους εκτέλεσης για τους τρεις αλγορίθμους ομαδοποίησης και για το αρχείο PULS 99

6.3 Μέσος χρόνος απόκρισης για όλες τις κλάσεις στην περίπτωση του *Reduced DOA* αρχείο ιχνών και για ταχύτητα επεξεργαστών 20 MIPS. 102

6.4 Μέσος χρόνος απόκρισης για όλες τις κλάσεις στην περίπτωση του *Reduced DOA* αρχείο ιχνών και για ταχύτητα επεξεργαστών 24 MIPS. 103

6.5 Μέσος χρόνος απόκρισης για όλες τις κλάσεις στην περίπτωση του *Reduced DOA* αρχείο ιχνών και για ταχύτητα επεξεργαστών 28 MIPS. 104

6.6 Μέσος χρόνος απόκρισης για όλες τις κλάσεις στην περίπτωση του *Reduced PULS* αρχείο ιχνών και για ταχύτητα επεξεργαστών 20 MIPS. 105

6.7 Μέσος χρόνος απόκρισης για όλες τις κλάσεις στην περίπτωση του *Reduced PULS* αρχείο ιχνών και για ταχύτητα επεξεργαστών 25 MIPS. 106

6.8 Μέσος χρόνος απόκρισης για όλες τις κλάσεις στην περίπτωση του *Reduced PULS* αρχείο ιχνών και για ταχύτητα επεξεργαστών 28 MIPS. 107

Κατάλογος Πινάκων

5.1 Βασικές παράμετροι του πρώτου πειράματος. 74

5.2 Διάφορες μετρήσεις για το σύστημα του πειράματος 1 (SD αρχιτεκτονική). . . . 76

5.3 Διάφορες μετρήσεις για το σύστημα του πειράματος 1 (SD αρχιτεκτονική).. . . 77

5.4 Διάφορες μετρήσεις για το σύστημα του πειράματος 1 (SIM αρχιτεκτονική). . . 78

5.5 Διάφορες μετρήσεις για το σύστημα του πειράματος 1 (SIM αρχιτεκτονική). . . 79

5.6 Βασικές παράμετροι του δεύτερου πειράματος. 81

5.7 Access Pattern για τις κλάσεις δοσοληψιών του πειράματος 2. 81

5.8 Πίνακας με τα υπόλοιπα χαρακτηριστικά για κάθε κλάση δοσοληψιών που εφαρμόστηκαν στο πείραμα 2. 82

5.9 Πίνακας με το αποτέλεσμα του αλγορίθμου δρομολόγησης (πίνακας πιθανοτήτων) για το πείραμα 2 και την SD αρχιτεκτονική. 84

5.10 Πίνακας με το αποτέλεσμα του αλγορίθμου δρομολόγησης (πίνακας πιθανοτήτων) για το πείραμα 2 και την SIM αρχιτεκτονική. 85

5.11 Διάφορες μετρήσεις για το σύστημα του πειράματος 2 (SD αρχιτεκτονική). . . . 85

5.12 Μετρήσεις για το buffer hit ratio για το σύστημα του πειράματος 2 (SD αρχιτεκτονική). 86

5.13 Μετρήσεις για το ποσοστό των δοσοληψιών που δεν έπιασαν τον στόχο επίδοσής τους (ανά κλάση) για το σύστημα του πειράματος 2 (SD αρχιτεκτονική) I. 86

5.14 Μετρήσεις για το ποσοστό των δοσοληψιών που δεν έπιασαν τον στόχο επίδοσής τους (ανά κλάση) για το σύστημα του πειράματος 2 (SD αρχιτεκτονική) II. 87

5.15 Διάφορες μετρήσεις για το σύστημα του πειράματος 2 (SIM αρχιτεκτονική). . . 88

5.16 Μετρήσεις για το buffer hit ratio για το σύστημα του πειράματος 2 (SIM αρχιτεκτονική). 88

5.17 Μετρήσεις για το ποσοστό των δοσοληψιών που δεν έπιασαν τον στόχο επίδοσής τους (ανά κλάση) για το σύστημα του πειράματος 2 (SIM αρχιτεκτονική) I. 90

5.18 Μετρήσεις για το ποσοστό των δοσοληψιών που δεν έπιασαν τον στόχο επίδοσής τους (ανά κλάση) για το σύστημα του πειράματος 2 (SIM αρχιτεκτονική) II. 90

6.1 Χαρακτηριστικά των αρχείων ιχνών *DOA* και *PULS* 97

6.2 Βασικές παράμετροι της προσομοίωσης που έγινε με τον προσομοιωτή TPsim. . 101

Κεφάλαιο 1

Εισαγωγή

1.1 Ιστορική αναδρομή

Η έννοια της δοσοληψίας έχει τις ρίζες της στον λαό των Σουμέριων, οι οποίοι πριν από έξι χιλιάδες χρόνια χρησιμοποιούσαν πλάκες από πηλό για να καταγράφουν τις εμπορικές συναλλαγές μεταξύ τους, αλλά και με τους άλλους λαούς που συναλλάσονταν. Στη συνέχεια αυτή η συνήθεια υιοθετήθηκε και από άλλους λαούς και σιγά σιγά ο πηλός αντικαταστάθηκε από τον πάπυρο και μετά από το χαρτί.

Με τον όρο δοσοληψία στον κλάδο της επιστήμης υπολογιστών εννοούμε τον μηχανισμό εκείνο, με τον οποίο δύο ή περισσότερα μέρη διαπραγματεύονται, με απώτερο σκοπό την έγκριση ή την απόρριψη μιας προτεινόμενης σύμβασης.

Στα τέλη τού προηγούμενου αιώνα έκανε και την εμφάνισή της η πρώτη μηχανή που ήταν σε θέση να καταγράφει τέτοιου είδους πληροφορίες. Αυτή ήταν μία μηχανή που κατασκεύασε ο Hollerith και χρησιμοποιούσε διάτρητες κάρτες προκειμένου να αποθηκεύει τις διάφορες πληροφορίες. Αυτή η μηχανή αποτέλεσε και τον πρόδρομο για τις λεγόμενες μηχανές επεξεργασίας δοσοληψιών. Εξέλιξή της αποτέλεσαν στη συνέχεια τα συστήματα μαζικής επεξεργασίας δοσοληψιών (batch systems), καθώς και τα on-line συστήματα που κυριαρχούν σήμερα.

Παραδείγματα χρήσης συστημάτων επεξεργασίας δοσοληψιών μπορούμε να δούμε στις τράπεζες για τη διαχείριση των τραπεζικών λογαριασμών αλλά και άλλων ειδών συναλλαγών που μπορούν να γίνουν στις τράπεζες, στους τηλεπικοινωνιακούς οργανισμούς, στα ταξιδιωτικά γραφεία αλλά και στα πρακτορεία εταιρειών μαζικής μεταφοράς για την κράτηση θέσεων, στον ασφαλιστικό κλάδο, σε οργανισμούς δημόσιους και μη, για την τήρηση λογιστικών βιβλίων, έκδοση λογαριασμών ή ακόμα και παροχή υπηρεσιών προς τους πελάτες τους. Τέλος, σιγά σιγά μπορούμε να πούμε ότι αρχίζουν να αποτελούν και μέρος του αυτοματισμού γραφείου μετά την παγκόσμια διασύνδεση των υπολογιστών μέσω του Internet. Είναι φανερό λοιπόν ότι σήμερα αυτά τα συστήματα στηρίζουν την λειτουργία πολλών οργανισμών και ο τρόπος καθώς και η ποιότητα της λειτουργία τους έχει άμεσο αντίκτυπο στην καθημερινή μας ζωή.

Η όλο και αυξανόμενη ανάγκη για υψηλό ρυθμό εξυπηρέτησης σε συνδυασμό με τις απαιτήσεις για ποιότητα εξυπηρέτησης, επεκτασιμότητα, μικρότερο κόστος και με την παράλληλη εκρηκτική αύξηση του πλήθους των αποθηκευμένων πληροφοριών, οδήγησαν γρήγορα σε αντικατάσταση τα σχεσιακά συστήματα βάσεων δεδομένων (relational database systems) που κυριάρχησαν την προηγούμενη δεκαετία. Αυτά τα συστήματα βασίζονταν σε υπερυπολογιστές (mainframes). Γρήγορα όμως φάνηκε ότι δεν μπορούσαν τέτοιες μηχανές να ανταποκριθούν στις αυξημένες απαιτήσεις των νέων εφαρμογών, στην εξυπηρέτηση δηλαδή μεγάλου αριθμού χρηστών και την επεξεργασία terabyte δεδομένων (δηλαδή στα θέματα I/O και CPU), καθώς οι νέες γενιές αποθηκευτικών μέσων και επεξεργαστών δεν ήταν ανάλογα γρήγορες ώστε να καλύψουν τον ρυθμό εξυπηρέτησης, ο οποίος αυξανόταν με ένα ρυθμό της τάξης του 40% με 50%.

Έτσι λοιπόν για τις εφαρμογές που διαχειρίζονται μεγάλο όγκο πληροφοριών, όπως είναι τα συστήματα βάσεων δεδομένων με online επεξεργασία δοσοληψιών, φάνηκε η ανάγκη για την χρήση κατανεμημένων συστημάτων. Σε αυτά τα συστήματα η επεξεργασία δεν γίνεται κάπου "κεντρικά" αλλά σε έναν αριθμό από υποσυστήματα τα οποία μπορεί να απέχουν μεταξύ τους ακόμα και εκατοντάδες χιλιόμετρα.

Υπάρχουν τρία βασικά είδη πόρων (resources): επεξεργαστής (processor), μνήμη (main memory) και αποθηκευτικά μέσα (secondary storage) συνήθως μαγνητικοί δίσκοι. Οι υπολογιστικοί κόμβοι (processing nodes) αποτελούνται από αυτά τα τρία είδη των πόρων. Ο αριθμός τους καθώς και ο τρόπος και η τοπολογία σύνδεσης είναι αυτός που ποικίλει. Έτσι στα παράλληλα και κατανεμημένα συστήματα διακρίνουμε τρεις βασικές αρχιτεκτονικές [YD94a], [Rah92a], [Bhi88]: την Shared Everything (SE), την Shared Nothing (SN) και την Shared Disk (SD) αρχιτεκτονική. Υποπερίπτωση της Shared Disk αρχιτεκτονικής είναι και η Shared Intermediate Memory (SIM). Για τα ιδιαίτερα χαρακτηριστικά τους θα αναφερθούμε στα επόμενα κεφάλαια.

Η επίδοση των αρχιτεκτονικών αυτών, δεδομένου του συστήματος και των χαρακτηριστικών του, εξαρτάται σε έναν μεγάλο βαθμό από τον χαρακτηρισμό του φόρτου εργασίας (workload characterization) που καλείται να εξυπηρετήσει το σύστημα. Για την βελτίωση της επίδοσης του ένα τέτοιο σύστημα θα πρέπει να είναι σε θέση να επιλέξει τον κατάλληλο κόμβο που θα εξυπηρετήσει την κάθε νέα δοσοληψία που φτάνει στο σύστημα. Πρέπει δηλαδή να μπορεί να δρομολογεί δοσοληψίες (transaction routing) στους κατάλληλους κόμβους βάσει των διαθέσιμων πόρων, της κατάστασης που βρίσκονται οι πόροι και μίας σειράς άλλων παραμέτρων που είναι δυνατό να λαμβάνουν υπόψη τους οι δρομολογητές (routers).

1.2 Αντικείμενο και συνεισφορά της εργασίας

Αντικείμενο αυτής της εργασίας αποτελεί η μελέτη των κατανεμημένων αρχιτεκτονικών που προαναφέρθηκαν (πλην της Shared Everything αρχιτεκτονικής).

Πιο συγκεκριμένα στην παρούσα εργασία έγινε επέκταση ενός προσομοιωτή κατανεμημένων συστημάτων επεξεργασίας δοσοληψιών, του **TPsim**, που αναπτύχθηκε

στο Ινστιτούτο Πληροφορικής του Ι.Τ.Ε. και στην ερευνητική ομάδα των Παράλληλων και Κατανεμημένων Συστημάτων από τον Μαραζάκη στα πλαίσια της μεταπτυχιακής του εργασίας [Μαρ95b]. Ο προσομοιωτής επεκτάθηκε ώστε να υποστηρίζει της αρχιτεκτονικές Shared Disk και Shared Intemediate Memory.

Συγχρόνως αναπτύχθηκε ένας νέος αλγοριθμος δρομολόγησης ειδικά για αυτές τις αρχιτεκτονικές και έγιναν κάποια πειράματα προκειμένου να εξετάσουν την χρησιμότητα και αποδοτικότητά του. Σε αυτά τα πειράματα πιστοποιήθηκε ότι πράγματι η χρησιμοποίηση του βελτιώνει σημαντικά την απόκριση του συστήματος.

Επίσης έγιναν πειράματα για την επίδραση αλγορίθμων δρομολόγησης σε συστήματα Shared Nothing με χρήση πραγματικών δεδομένων και αφού προηγουμένως τα δεδομένα αυτά είχαν υποστεί μία επεξεργασία από ένα άλλο εργαλείο που αναπτύχθηκε στο Ινστιτούτο Πληροφορικής του Ι.Τ.Ε., το CLUE [Λαμ95a], [Κλο97], το οποίο έχει ως σκοπό να ομαδοποιήσει έναν αριθμό από δοσοληψίες σε ομάδες με παρόμοια χαρακτηριστικά, με σκοπό την καλύτερη απόδοση του συστήματος με χρήση αλγορίθμων δρομολόγησης. Και σε αυτήν την περίπτωση η επίδραση της ομαδοποίησης στην δρομολόγηση δοσοληψιών ήταν μεγάλη. Μία καλή ομαδοποίηση βελτίωνε την απόδοση των αλγορίθμων δρομολόγησης, ενώ στις περιπτώσεις όπου ο αλγόριθμος ομαδοποίησης δεν έδινε καλά αποτελέσματα τότε οι αλγόριθμοι δρομολόγησης να χειροτέρευαν σε μεγάλο βαθμό την απόδοση του συστήματος.

1.3 Οργάνωση της εργασίας

Στο κεφάλαιο 2 θα αναφερθούμε σε χρήσιμες έννοιες και θέματα που άπτονται του επιστημονικού υπόβαθρου της εργασίας. Στο κεφάλαιο 3 θα αναπτύξουμε τις αρχιτεκτονικές επεξεργασίας δοσοληψιών που έχουν επικρατήσει. Στη συνέχεια και στο κεφάλαιο 4 θα αναφερθούμε στις λεπτομέρειες της υλοποίησης των αρχιτεκτονικών *Shared Disk* και *Shared Intermediate Memory* στον προσομοιωτή συστημάτων επεξεργασίας δοσοληψιών, TPsim. Με βάση την υλοποίηση των αρχιτεκτονικών ένας νέος αλγόριθμος δρομολόγησης αναπτύχθηκε και δοκιμάστηκε και τα αποτελέσματα βρίσκονται στο κεφάλαιο 5. Το κεφάλαιο 6 μελετάται η επίπτωση της ομαδοποίησης δοσοληψιών στην δρομολόγηση δοσοληψιών για την αρχιτεκτονική επεξεργασίας δοσοληψιών *Shared Nothing* . Τέλος κάποια συμπεράσματα και μελλοντικές κατευθύνσεις δίνονται στο κεφάλαιο 7.

Κεφάλαιο 2

Επιστημονικό υπόβαθρο

Στο κεφάλαιο αυτό θα δούμε ορισμένες από τις πιο βασικές έννοιες σχετικές με το πεδίο ενασχόλησης της εργασίας. Αποτελούν συνοπτικές αλλά αρκετά χρήσιμες περιγραφές που θα βοηθήσουν έναν αναγνώστη που δεν είναι εξοικειωμένος με την συγκεκριμένη περιοχή ειδίκευσης, δίνοντας του παράλληλα και ορισμένες κατευθύνσεις για επιπλέον μελέτη.

2.1 Η έννοια της δοσοληψίας

Η δοσοληψία στις βάσεις δεδομένων είναι ουσιαστικά ένα είδος αφαίρεσης για τον τρόπο μεταβολής της κατάστασης της βάσεως από μία κατάσταση σε μία άλλη [GR93]. Αποτελείται από μία σειρά από απλές "πράξεις" οι οποίες όλες μαζί εκτελούν μία πιο πολύπλοκη διαδικασία. Ο όρος αυτός αποτελεί βασική έννοια για τον εμπορικό και τραπεζικό κλάδο. Η χρέωση και η πίστωση λογαριασμού αποτελούν παραδείγματα τραπεζικών δοσοληψιών. Η μετάβαση από μία κατάσταση σε μία άλλη θα πρέπει πάντα να γίνεται με ένα συγκεκριμένο και "συνεπή" τρόπο για την βάση δεδομένων. Με τον όρο συνεπή, εννοούμε ότι δεν θα πρέπει να παραβιάζονται κάποιοι κανόνες και περιορισμοί που υπάρχουν, π.χ. κατά την μεταφορά ενός ποσού από τον λογαριασμό Α στον λογαριασμό Β θα πρέπει προφανώς το άθροισμα του ποσού των δύο λογαριασμών πριν και μετά να είναι το ίδιο. Αυτοί οι περιορισμοί είναι από πριν καθορισμένοι από τους σχεδιαστές της βάσεως δεδομένων και θα πρέπει οι δοσοληψίες που επενεργούν πάνω στην κατάσταση της βάσης να "φροντίζουν" να διαφυλάτουν αυτούς τους περιορισμούς.

Άλλωστε η δοσοληψία είναι συνυφασμένη με τις παρακάτω ιδιότητες, οι οποίες στην διεθνή βιβλιογραφία έχουν επικρατήσει με την ονομασία ACID ιδιότητες, από τα αρχικά γράμματα των αγγλικών ονομασιών των ιδιοτήτων. Οι ιδιότητες αυτές είναι οι εξής:

- **Ατομικότητα** (Atomicity) : Οι αλλαγές στην κατάσταση του συστήματος θα πρέπει κατά το τέλος της δοσοληψίας, ή όλες να γίνονται αποδεκτές ή καμία από όλες. Στο προηγούμενο παράδειγμα με τους λογαριασμούς θα πρέπει στο τέλος στον έναν λογαριασμό να έχει αφαιρεθεί το ποσό και στον άλλο να έχει προστεθεί το ίδιο ποσό.

Δεν μπορεί δηλαδή να γίνει αποδεκτή μόνο η μία από τις δύο "πράξεις" στη βάση. Η ιδιότητα αυτή είναι γνωστή και ως ιδιότητα "όλα ή τίποτα" ("all or nothing").

- **Συνέπεια** (**C**onsistency) : Η δοσοληψία δεν θα πρέπει να παραβιάζει τους περιορισμούς ακεραιότητας (integrity constraints) του συστήματος. Θα πρέπει δηλαδή η δοσοληψία-πρόγραμμα να είναι μία σωστή πράξη πάνω στην κατάσταση του συστήματος, αποδεκτή και σύμφωνη με τους περιορισμούς που έχει θέσει ο σχεδιαστής της και που μπορεί να υπαγορεύονται από την κοινή λογική ή από συγκεκριμένες ανάγκες.
- **Απομόνωση** (**I**solation) : Αν δύο οι περισσότερες δοσοληψίες εκτελούνται ταυτόχρονα, θα πρέπει κάθε μία από αυτές να "νομίζει" ότι όλες οι υπόλοιπες εκτελούνται πριν ή μετά από αυτήν. Η ιδιότητα αυτή διασφαλίζει ότι τα αποτελέσματα ενός αριθμού δοσοληψιών που εκτελούνται ταυτόχρονα θα είναι το ίδιο με αυτό που θα προέκυπτε αν εκτελούνταν οι ίδιες δοσοληψίες σειριακά, με οποιαδήποτε σειρά. Η ιδιότητα αυτή έχει επικρατήσει διεθνώς και με τον όρο σειριοποιησιμότητα (serializability).
- **Μονιμότητα** (**D**urability) : Όταν μία δοσοληψία τερματίσει επιτυχώς τότε οι αλλαγές που αυτή επέφερε δεν μπορούν να ακυρωθούν από οποιαδήποτε βλάβη του συστήματος. Η αναίρεση των αλλαγών που αυτή επέφερε είναι δυνατή μόνο με τον επιτυχή τερματισμό διορθωτικών δοσοληψιών.

Λόγω της ατομικότητας δύο είναι οι πιθανοί τερματισμοί της εκτέλεσης μιας δοσοληψίας:

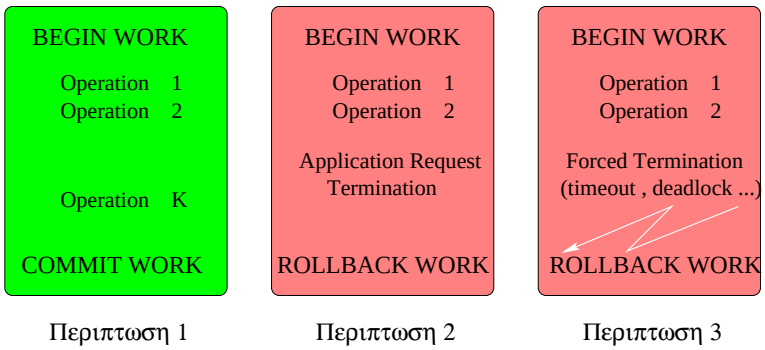
- Επιτυχής τερματισμός (commit) οπότε και οι αλλαγές γίνονται μόνιμες και ορατές από τις υπόλοιπες δοσοληψίες.
- Ανεπιτυχής τερματισμός (abort) οπότε καμιά αλλαγή από αυτές που έχει επιφέρει η δοσοληψία δεν γίνεται αντιληπτή από άλλη δοσοληψία.

Η έννοια της δοσοληψίας σε συνδυασμό με τις ιδιότητες που την συνοδεύουν την καθιστούν βασικό δομικό στοιχείο για τον σχεδιασμό και τη υλοποίηση συστημάτων που διαχειρίζονται πληροφορίες και αντιμετωπίζουν το ενδεχόμενο βλαβών.

2.2 Μοντέλα δοσοληψιών

Τα μοντέλα δοσοληψιών [GR93], έχουν ως σκοπό την παροχή μιας ικανής αναπαράστασης των διαφορετικών μοντέλων επεξεργασίας (processing patterns) που είναι δυνατό να υπάρχουν σε εφαρμογές που απευθύνονται σε ένα σύστημα επεξεργασίας δοσοληψιών. Το μοντέλο της απλής δοσοληψίας τύπου ενός επιπέδου (flat) εδώ και 25 περίπου χρόνια αποτελεί το βασικό μοντέλο δοσοληψιών που υποστηρίζουν τα περισσότερα συστήματα. Με την ανάπτυξη όμως μεγαλυτέρων αλλά και πολύ πιο πολύπλοκων εφαρμογών, φάνηκε η ανάγκη υποστήριξης μοντέλων πιο πολύπλοκων από αυτό της απλής περίπτωσης της flat δοσοληψίας.

Στην παράγραφο αυτή θα δούμε περιληπτικά τα βασικά μοντέλα δοσοληψιών που υπάρχουν.



Σχήμα 2.1: Δυνατοί τρόποι τερματισμού μίας flat δοσοληψίας.

2.2.1 Flat δοσοληψίες

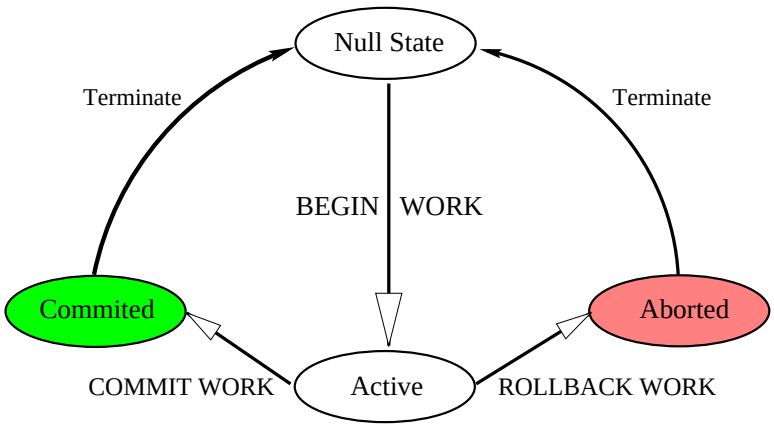
Οι ενός επιπέδου (flat) δοσοληψίες, είναι το πιο απλό μοντέλο δοσοληψίας και συνήθως είναι και το μοναδικό που υποστηρίζεται σε προγραμματιστικό επίπεδο εφαρμογών (application programming level) από τα υπάρχοντα συστήματα επεξεργασίας δοσοληψιών. Πολλά συστήματα παρουσιάζουν ιδιότητες πιο γενικών μοντέλων, χωρίς όμως να επιτρέπουν την χρήση αυτών των μοντέλων στο γενικό programming level. Γι' αυτό, η μελέτη και η έμφαση που έχει δοθεί στις flat δοσοληψίες είναι πολύ μεγαλύτερη από τα υπόλοιπα μοντέλα.

Η flat δοσοληψία είναι το βασικό δομικό στοιχείο για την οργάνωση εφαρμογών σε μία σειρά από ατομικές πράξεις, πράξεις δηλαδή για τις οποίες θα πρέπει να ισχύει η ιδιότητα της ατομικότητας. Μπορεί να περιλαμβάνει έναν οποιοδήποτε αριθμό από τέτοιες πράξεις. Από την πλευρά της εφαρμογής, αυτές οι πράξεις φαίνονται σαν μία "αδιάσπαστη" πράξη, δηλαδή χωρίς ενδιάμεσα ορατά αποτελέσματα. Αυτή η "αδιάσπαστη" πράξη μπορεί να εκτελείται σειριακά, παράλληλα ή ακόμα και κατανεμημένα αρκεί να μπορεί να τηρεί την προϋπόθεση της μη ύπαρξης ενδιάμεσων αποτελεσμάτων.

Στο σχήμα 2.1 φαίνονται οι δυνατοί τρόποι τερματισμού μίας δοσοληψίας. Στη πρώτη περίπτωση έχουμε τον επιτυχή τερματισμό της δοσοληψίας (96% περίπου όλων των δοσοληψιών). Στη δεύτερη περίπτωση έχουμε ανεπιτυχή τερματισμό εξαιτίας αίτησης της εφαρμογής (3% περίπου όλων των δοσοληψιών) και τέλος στην τρίτη περίπτωση έχουμε ανεπιτυχή τερματισμό εξαιτίας ενός αδιεξόδου (deadlock) ή ενός timeout που συνέβη (1% περίπου όλων των δοσοληψιών). Τα ποσοστά αυτά είναι τυπικά για τα συστήματα επεξεργασίας flat δοσοληψιών.

Το BEGIN WORK καθορίζει την αρχή μίας δοσοληψίας. Οποιαδήποτε πράξη βρίσκεται ανάμεσα στα BEGIN WORK και COMMIT WORK βρίσκεται στο ίδιο επίπεδο, με την έννοια ότι, εάν η δοσοληψία τερματίσει επιτυχώς (commit), τότε θα γίνουν μόνιμες και οι εκτελεσθέντες από αυτήν πράξεις. Αλλιώς, σε περίπτωση δηλαδή που η δοσοληψία τερματίσει ανεπιτυχώς (abort) τότε οι εκτελεσθέντες από αυτήν πράξεις θα πρέπει να ακυρωθούν (roll - back). Για όλες τις πράξεις που βρίσκονται ανάμεσα στα BEGIN WORK και COMMIT WORK ισχύουν οι ACID ιδιότητες που είδαμε στην προηγούμενη παράγραφο. Μετά το COMMIT WORK έχουμε πλέον μία νέα συνεπή κατάσταση του συστήματος.

Στο σχήμα 2.2 φαίνονται οι δυνατές μεταβολές στην κατάσταση μίας δοσοληψίας. Η



Σχήμα 2.2: Διάγραμμα καταστάσεων / μεταβάσεων μίας δοσοληψίας.

κατάσταση που σημειώνεται ως NULL συμβολίζει την κατάσταση μίας δοσοληψίας μετά τον τερματισμό αλλά και πριν ακόμα ξεκινήσει η επεξεργασία της. Όσο διαρκεί η επεξεργασία της παραμένει σε μία ενεργή (active) κατάσταση και ανάλογα με τον τρόπο τερματισμού της, επιτυχή ή ανεπιτυχή, η κατάστασή της μεταβάλλεται σε committed ή aborted κατάσταση αντίστοιχα.

Η δοσοληψία πίστωσης/χρέωσης (Debit/Credit)

Η δοσοληψία πίστωσης/χρέωσης είναι η πιο δημοφιλής flat δοσοληψία. Η δοσοληψία ξεκίνησε αρχικά από τον τραπεζικό τομέα και η συγκεκριμένη δοσοληψία είναι μία πολύ απλοποιημένη μορφή τραπεζικής δοσοληψίας. Αποτελεί δε βάση τους δείκτες επίδοσης (benchmarks) που χρησιμοποιούνται για την μέτρηση της επίδοσης των συστημάτων και συγκεκριμένα για τα TPC-A και TPC-B benchmarks [Gra91].

Σε γενικές γραμμές αυτό που κάνει η δοσοληψία πίστωσης/χρέωσης είναι το εξής: η δοσοληψία παίρνει αρχικά ένα μήνυμα από το τερματικό για να πιστώσει ή να χρεώσει το ποσό ενός λογαριασμού. Η δοσοληψία κάνει την αλλαγή του ποσού, ενημερώνει την αντίστοιχη εγγραφή του καταθέτη (teller) και του υποκαταστήματος της τράπεζας (branch) και στη συνέχεια προσθέτει μία εγγραφή στο ιστορικό (history) που κρατάει. Το νέο ποσό στη συνέχεια επιστρέφει πίσω στην εφαρμογή η οποία αναλαμβάνει να δώσει μία απάντηση στον χρήστη (ότι η συναλλαγή τελείωσε επιτυχώς).

2.3 Δείκτες επίδοσης (Benchmarks)

Η πρόβλεψη και μέτρηση της επίδοσης ενός συστήματος επεξεργασίας δοσοληψιών δεν είναι εύκολη υπόθεση διότι τα συστήματα αυτά είναι πολύ πολύπλοκα και γιατί η συμπεριφορά τους σε διαφορετικές καταστάσεις μπορεί να είναι πολύ διαφορετική και μη αναμενόμενη. Γι' αυτό και έχουν καθιερωθεί συγκεκριμένα σενάρια φόρτου αλλά και μετρητές επίδοσης που περιγράφονται αναλυτικά στο [Gra91].

Το βιομηχανικό consortium, Transaction Processing Performance Council (TPC), το οποίο απαρτίζουν κατασκευαστές συστημάτων, έχει καθορίσει τέσσερα benchmarks. Αυτά έχουν ονομαστεί TPC-A [Cou90a], TPC-B [Cou90b], TPC-C [Cou93], και TPC-D [Cou95]. Καθένα από αυτά συνοδεύεται από μία βάση δεδομένων που μπορεί να μεταβληθεί σε μέγεθος αλλά και ένα συγκεκριμένο workload το οποίο τρέχει πάνω στο προς παρατήρηση σύστημα. Χρησιμοποιώντας τα, ένα σύστημα χαρακτηρίζεται από αυτούς τους δείκτες τα οποία παριστάνουν τις επιδόσεις του συστήματος στα σενάρια αυτά. Οι δείκτες αυτοί ονομάζονται tps-A, tps-B, tps-C και tps-D (από τα αρχικά transactions per second). Επίσης τα benchmarks αυτά χρησιμοποιούνται για να δώσουν και την σχέση κόστους/επίδοσης.

2.3.1 Επεκτάσεις του flat μοντέλου δοσοληψίας

Βασικός περιορισμός των flat δοσοληψιών είναι ότι δεν υπάρχει τρόπος να γίνουν commit ή abort τμήματα μόνο της δοσοληψίας. Αυτό σε ορισμένες εφαρμογές είναι επιθυμητό και ώθησε στην δημιουργία πιο γενικών μοντέλων [GR93]. Ας δούμε πρώτα όμως ένα παράδειγμα για να καταλάβουμε γιατί οι flat δοσοληψίες δεν είναι κατάλληλες για όλες τις εφαρμογές.

Έστω η περίπτωση ενός πρακτορείου ταξιδιών. Πολύ συχνά υπάρχει η ανάγκη, μετά από απαίτηση ενός πελάτη, να προγραμματιστεί ένα ταξίδι (trip planning) το οποίο απαιτεί την αλλαγή μέσου μεταφοράς, π.χ. να ταξιδέψει κανείς από το Ηράκλειο προς μία πόλη των ΗΠΑ που δεν έχει απευθείας αεροπορική πτήση από Ευρώπη (έστω Ντιτρόιτ). Για ένα τέτοιο αεροπορικό ταξίδι μπορεί να χρειαστεί να προγραμματιστεί μία πτήση Ηράκλειο - Αθήνα, μία Αθήνα - Ατλάντα και τέλος μία πτήση Ατλάντα - Ντιτρόιτ. Έστω ότι για την πρώτη και δεύτερη πτήση υπάρχει θέση στο αεροπλάνο αλλά για την τελευταία δεν υπάρχει. Αυτό θα οδηγούσε μία flat δοσοληψία σε abort οπότε όλες οι θέσεις που βρήκε στα αεροπλάνα μέχρι την Ατλάντα θα έπρεπε να φανούν και πάλι ελεύθερες στο σύστημα. Στη συνέχεια ο πράκτορας πιθανόν θα προσπαθούσε να προγραμματίσει μία εναλλακτική διαδρομή μέσω Ορλάντο. Όμως λόγω του ότι το σύστημα έχει έναν τεράστιο αριθμό από τερματικά, είναι δυνατόν τώρα στην πτήση Ηράκλειο - Αθήνα να μην υπάρχει διαθέσιμη θέση. Μπορεί αυτό το παράδειγμα να φαίνεται αρκετά "ανώδυνο" γιατί ο πελάτης δεν θα καταλάβει τίποτα, αλλά υπάρχουν περιπτώσεις που η ακύρωση των αποτελεσμάτων της δοσοληψίας να επιφέρει σημαντική καθυστέρηση σε μία επιχείρηση (τοκισμός των λογαριασμών μιας τράπεζας).

Συχνά λοιπόν είναι επιθυμητό να είναι δυνατό το roll-back τμήματος μόνο της δοσοληψίας οπότε θα πρέπει ορισμένα ενδιάμεσα αποτελέσματα να αποθηκεύονται. Βασικές επεκτάσεις της flat δοσοληψίας είναι οι εξής :

1. **Σφαίρες ελέγχου (Spheres of control)** : Η βασική ιδέα στις σφαίρες ελέγχου (Davies 1978), είναι ότι θα πρέπει να κρατείται όλη η ιστορία της επεξεργασίας των δοσοληψιών. Αυτό απαιτεί την ύπαρξη ενός προσωρινού μοντέλου δεδομένων (temporal data model), όπου κρατούνται οι εξαρτήσεις των τιμών, δηλαδή ο τρόπος που μία τιμή επηρεάζει τις άλλες. Έτσι εάν κάποια τιμή αλλάξει, το σύστημα αναλαμβάνει να βρει ποιές άλλες τιμές πρέπει να αλλαχθούν αλλά και με ποιό τρόπο. Η υλοποιήση αυτής της ιδέας δεν αποτελεί μέρος του μοντέλου που πρότεινε ο Davies [Dav78]. Αυτό αποτελεί και το βασικό μειονέκτημα αυτής της κατηγορίας των δοσοληψιών μια και γενικά κάτι τέτοιο

είναι πολύ δύσκολο να υλοποιηθεί. Αποτελεί δε το πιο γενικό μοντέλο δοσοληψίας από όλα όσα έχουν προταθεί.

2. **Flat δοσοληψίες με σημεία όπου κρατούνται ενδιάμεσες καταστάσεις (Flat transactions with savepoints)** : Δίνουν την δυνατότητα να γίνουν roll-back τμήματα μίας δοσοληψίας, χωρίς να πρέπει να γίνει roll-back όλη η δοσοληψία. Δηλαδή δίνουν την δυνατότητα για επαναφορά σε μία προϋπάρχουσα κατάσταση που είχε προκύψει **κατά την διάρκεια** εκτέλεσης της δοσοληψίας. Αυτό δεν είναι δυνατό να συμβεί στις flat δοσοληψίες διότι έχουμε πει ότι οι ενδιάμεσες καταστάσεις δεν είναι ορατές. Εάν δεν είναι επιθυμητό να υπάρχουν δοσοληψίες μέσα σε άλλες δοσοληψίες (η άλλη λύση που θα δούμε παρακάτω), τότε η χρήση των savepoints αναγκάζει το σύστημα να αποθηκεύσει την τρέχουσα κατάσταση του συστήματος ούτως ώστε να είναι σε θέση να την επαναφέρει σε περίπτωση που κάτι δεν πάει καλά. Ένα savepoint γίνεται με την κλήση μίας SAVE_WORK διαδικασίας. Η χρήση τους είναι ιδιαίτερα ελκυστική σε περιπτώσεις που το roll-back μίας δοσοληψία έχει μεγάλο κόστος.

3. **Φωλιασμένες δοσοληψίες (Nested transactions)** : Αποτελούν γενίκευση των savepoints. Ενώ η χρήση των savepoints επιτρέπει την οργάνωση μίας δοσοληψίας σε μία ακολουθία πράξεων που μπορούν να γίνουν roll-back ανεξάρτητα, οι φωλιασμένες δοσοληψίες δημιουργούν μία ιεραρχία από πράξεις. Στην περίπτωση του trip planning προβλήματος, θα υπήρχε μία (top-level) δοσοληψία που θα έλεγχε τον όλο προγραμματισμό και μία σειρά από (low-level) υποδοσοληψίες (subtransactions), οι οποίες θα αναλάμβαναν τον έλεγχο των επιμέρους τμημάτων του ταξιδιού. Κάθε μία από subtransactions μπορεί να γίνει commit ή roll-back ανεξάρτητα από τις υπόλοιπες. Επίσης είναι δυνατό να επεκταθεί η ιεραρχική αυτή δομή σε περισσότερα επίπεδα.

4. **Αλυσιδωτές δοσοληψίες (Chained transactions)** : Οι δοσοληψίες αυτές, επιτρέπουν να γίνουν commit ενδιάμεσα αποτελέσματα ώστε να μην χαθούν σε περίπτωση βλάβης. Παράλληλα όμως φροντίζεται, από τον διαχειριστή τέτοιων δοσοληψιών, να μην παραχωρούνται οι πόροι που έχουν δεσμευτεί από την δοσοληψία σε άλλες.

5. **Δοσοληψίες πολλών επιπέδων (Multi-level transactions)** : Είναι ένα είδος φωλιασμένων δοσοληψιών που επιτρέπουν να γίνει νωρίς το commit ενδιαμέσων αποτελεσμάτων, αρκεί η ιδιότητα της απομόνωσης να ελέγχεται από τα ανώτερα επίπεδα της ιεραρχικής δομής της δοσοληψίας και να είναι δυνατό το roll-back αυτών των πρόωρα commit αποτελεσμάτων.

6. **Κατανεμημένες δοσοληψίες (Distributed transactions)** : Είναι τυπικές κατά τα άλλα flat δοσοληψίες, που εκτελούνται σε ένα κατανεμημένο περιβάλλον και γι' αυτό είναι δυνατό να επισκέπτονται διάφορους υπολογιστικούς κόμβους του συστήματος προκειμένου να αποκτήσουν πρόσβαση στα δεδομένα που χρειάζονται. Η διαφορά τους σε σχέση με τις φωλιασμένες δοσοληψίες είναι ότι η δομή τους εξαρτάται από την κατανομή των δεδομένων στο σύστημα και όχι από τις λειτουργικές ενότητες της εφαρμογής.

7. **Δοσοληψίες μεγάλης διάρκειας "ζωής" (Long-lived transactions)** : Είναι δοσοληψίες που επιτρέπουν την συνέχιση της εκτέλεσής τους μετά από μία βλάβη, από το

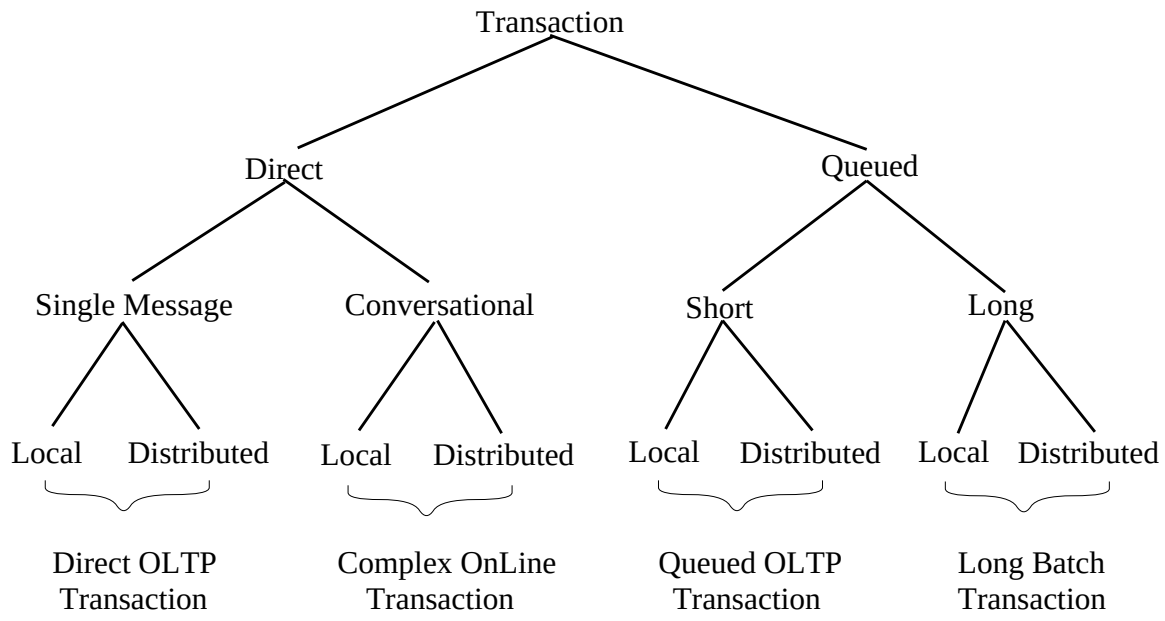
σημείο ακριβώς που βρίσκονταν πριν. Για να μην χαθεί λοιπόν εργασία, θα πρέπει εκτός της αποθήκευσης των ενδιάμεσων αποτελεσμάτων, να αποθηκεύεται και το σημείο επανεκκίνησης. Το κέρδος από την χρησιμοποίησή τους για την διαδικασία τοκισμού των λογαριασμών μίας τράπεζας είναι προφανής διότι σε περίπτωση βλάβης η διαδικασία μπορεί να συνεχίσει από εκεί που σταμάτησε χωρίς να χρειάζεται ακύρωση όλων των αλλαγών όπως θα απαιτούσε η χρήση της απλής δοσοληψίας.

Στην παρούσα εργασία το μοντέλο δοσοληψίας που έχει χρησιμοποιηθεί είναι οι κατανεμημένες δοσοληψίες.

2.4 Κατηγορίες δοσοληψιών με βάση την εκτέλεση τους

Ανάλογα με τον τρόπο και τόπο που εκτελούνται οι δοσοληψίες, μπορούμε να τις διακρίνουμε σε κατηγορίες (σχήμα 2.3) [GR93]. Κάθε επίπεδο του δέντρου του σχήματος αντιστοιχεί και σε μία από τις εξής κατηγορίες: χρονοπρογραμματισμός εκτέλεσης, διαδικασία εισόδου/εξόδου ανάμεσα στο πρόγραμμα εφαρμογής και το σύστημα, και τοποθεσίας των πόρων που λαμβάνουν μέρος στην επεξεργασία. Αν θεωρήσουμε ως επίπεδο 0 την ρίζα του δέντρου τότε αναλυτικά για κάθε επίπεδο έχουμε:

- **Επίπεδο 1** Άμεσα εκτελέσιμη - μη διαλογική δοσοληψία (direct - queued transaction) : Στις άμεσα εκτελέσιμες δοσοληψίες το τερματικό αλληλεπιδρά άμεσα με τον πρόγραμμα που έχει αναλάβει την εξυπηρέτηση της αίτησης. Αντίθετα στις μη διαλογικές δοσοληψίες, οι δοσοληψίες δεν εξυπηρετούνται άμεσα αλλά μπαίνουν σε μία ουρά αναμονής και ανάλογα με την πολιτική χρονοπρογραμματισμού γίνεται και η επεξεργασία της αίτησης.
- **Επίπεδο 2** Απλή - πολύπλοκη δοσοληψία (simple - complex transaction) : Η απλή δοσοληψία μπορεί να είναι όπως φαίνεται στο σχήμα 2.3,δοσοληψία ενός μηνύματος (single message), ή μικρής χρονικής διάρκειας εκτέλεσης (short) ενώ η πολύπλοκη δοσοληψία μπορεί να είναι διαλογική (conversational) οπότε απαιτείται η ανταλλαγή αρκετών μηνυμάτων ή μεγάλης χρονικής διάρκειας εκτέλεσης (long). Οι απλές δοσοληψίες είναι μικρές και επηρεάζουν ένα μικρό αριθμό από αντικείμενα της βάσης, ενώ οι πολύπλοκες έχουν πολύπλοκη δομή και επηρεάζουν αρκετά την βάση δεδομένων κάνοντας ενημερώσεις. Παράλληλα οι τελευταίες παραμένουν στο σύστημα για πολύ μεγαλύτερο χρονικό διάστημα, κάνουν χρήση περισσότερων πόρων και γενικά απαιτούν περισσότερη επεξεργασία. Ανταλλάσσουν και πολύ μεγαλύτερο αριθμό μηνυμάτων σε αντίθεση με τις απλές που αρκεί ένα μήνυμα. Στις μη άμεσα εκτελέσιμες δοσοληψίες υπάρχει επιπλέον και η καθυστέρηση στις ουρές προκειμένου τα μηνύματα αυτά να επεξεργαστούν, διότι κάθε απάντηση που δίνεται σε ένα μήνυμα πηγαίνει σε μία ουρά και ανάλογα με την πολιτική εξυπηρέτησης που υπάρχει υφίσταται και τις ανάλογες καθυστερήσεις.
- **Επίπεδο 3** Τοπική - κατανεμημένη δοσοληψία (local - distributed transaction) : Οι δοσοληψίες μπορεί να εκτελεστούν στον τοπικό κόμβο όπου και έγινε η αίτηση ή μπορεί να εκτελεστούν με την βοήθεια ενός αριθμού από κατανεμημένους κόμβους.



Σχήμα 2.3: Κατηγορίες δοσοληψιών με βάση την εκτέλεση τους.

2.5 Συστήματα επεξεργασίας δοσοληψιών

Ένα σύστημα επεξεργασίας δοσοληψιών (transaction processing system) παρέχει μηχανισμούς προκειμένου να διευκολύνει την εκτέλεση και διαχείριση εφαρμογών που βασίζονται στην έννοια της δοσοληψίας. Οι εφαρμογές αυτές διατηρούν κάποια βάση δεδομένων και παρέχουν την δυνατότητα σε ένα σύνολο από συσκευές να αλληλεπιδρούν με την βάση, κάνοντας ερωτήσεις, ενημερώσεις και συνεπώς καθοδηγούν αυτές τις εφαρμογές ώστε να μεταβάλλουν την κατάσταση του συστήματος.

Η τάση στα συστήματα επεξεργασίας δοσοληψιών είναι να γίνονται ολοένα και πιο γεωγραφικά κατανεμημένα και συγχρόνως να περιλαμβάνουν συσκευές και προγράμματα ετερογενή, οι οποίες θα πρέπει να είναι διαρκώς διαθέσιμες. Συχνά επιπλέον τίθεται και θέμα ποιότητας εξυπηρέτησης με την έννοια ότι θα πρέπει να παρέχουν εγγυήσεις χρονικής απόκρισης.

Προκειμένου αυτά τα συστήματα να εγγυηθούν τις ACID ιδιότητες των δοσοληψιών που καλούνται να επεξεργαστούν, κάνουν χρήση ορισμένων βασικών μηχανισμών οι οποίοι είναι δυνατόν να είναι υλοποιημένοι με αρκετά διαφορετικό τρόπο μεταξύ τους. Τέτοιοι μηχανισμοί είναι:

- Ο έλεγχος συνδρομικότητας (concurrency control) : Συνήθως επιτυγχάνεται βάση ενός πρωτοκόλλου κλειδώματος σελίδων της βάσης (locking protocol) ώστε να διασφαλιστεί η απομόνωση των δοσοληψιών από τις υπόλοιπες οι οποίες εκτελούνται ταυτόχρονα και είναι δυνατό να προσπελαύνουν την ίδια χρονική στιγμή τα ίδια δεδομένα.
- Ο μηχανισμός επαναφοράς (recovery mechanism) : Συνήθως επιτυγχάνεται με χρήση ενός μηχανισμού καταγραφής μεταβολών (logging mechanism). Διασφαλίζεται έτσι η

ατομικότητα αλλά και η μονιμότητα των αλλαγών που επιφέρει στην κατάσταση της βάσης μία δοσοληψια. Οι μεταβολές αυτές καταγράφονται σε μία μόνιμη μνήμη έτσι ώστε μην είναι δυνατό να χαθούν οι μεταβολές που έγιναν μετά από μία πιθανή βλάβη. Επίσης είναι δυνατόν να αναιρεθούν οι μεταβολές μίας δοσοληψίας εφόσον αυτή δεν τερματίσει επιτυχώς.

Ένα σύστημα επεξεργασίας δοσοληψιών αποτελείται από έναν αριθμό διαχειριστών πόρων και ένα υποσύστημα που ονομάζεται επόπτης επεξεργασίας δοσοληψιών (transaction processing monitor).

Ειδικά τα on-line συστήματα επεξεργασίας δοσοληψιών (On-Line Transaction Processing systems ή αλλιώς OLTP συστήματα) τα οποία είναι υλοποιημένα κατανεμημένα, πρέπει να έχουν μία σειρά από χαρακτηριστικά όπως:

- Συνεχή διαθεσιμότητα καθώς θα πρέπει η πιθανότητα βλάβης όπου το σύστημα δεν μπορεί να εξυπηρετήσει αιτήσεις να είναι πολύ μικρή.
- Προβλεψιμότητα, εφόσον οι μέγιστοι χρόνοι απόκρισης των δοσοληψιών δεν θα πρέπει να μεταβάλλονται σημαντικά σε σχέση με την ώρα, ημέρα, εποχή κ.ο.κ.
- Ακεραιότητα δοσοληψιών, διότι η δοσοληψία που επεξεργάζεται το σύστημα θα πρέπει να έχει άμεσο αντίκτυπο στον πραγματικό κόσμο, π.χ. όταν κάνει κανείς κράτηση θέσης σε ένα αεροπλάνο για μία συγκεκριμένη πτήση τότε αυτή η θέση εμφανίζεται στο σύστημα ως μη διαθέσιμη για την συγκεκριμένη πτήση.
- Ασφάλεια δοσοληψίας, διότι το σύστημα θα πρέπει να προφυλάσεται από παρεμβάσεις μη εξουσιοδοτημένων ατόμων.

Το αποτέλεσμα που έχουν αυτά τα συστήματα είναι ότι γλυτώνουν την επιχείρηση ή τον οργανισμό από απλές και μηχανικές εργασίες (low-level), απασχολώντας το δυναμικό τους σε άλλες εργασίες.

Συγχρόνως η αυτοματοποίηση αυτή οδηγεί σε μεγαλύτερη παραγωγικότητα εφόσον οι μηχανές κάνουν πιο γρήγορα και σωστά τέτοιες πράξεις (εφόσον είναι σωστά προγραμματισμένες).

2.6 Μοντέλα υπολογισμού

Τα υπολογιστικά συστήματα χρησιμοποιούνται για πολύ διαφορετικές εφαρμογές οπότε και οι απαιτήσεις από τα συστήματα είναι πολλές φορές σημαντικά διαφορετικές [GR93]. Για παράδειγμα, το μέγεθος της θεωρούμενης μονάδας φόρτου εργασίας μπορεί να είναι πολύ διαφορετικό κάτι που με την σειρά του επηρεάζει μία σειρά από παραμέτρους όπως τον αριθμό των πόρων που αναθέτουμε σε κάθε τέτοια μονάδα φόρτου, το χρονικό διάστημα ανάθεσης, το βαθμό παραλληλισμού αλλά και έναν μεγάλο αριθμό από άλλες παραμέτρους. Στο σχήμα 2.4 φαίνονται οι βασικές διαφορές μεταξύ των μοντέλων υπολογισμού (computing

	Batch Processing	Time-Sharing	Real-Time Processing	Client-Server	Transaction Oriented Processing
Data	Private	Private	Private	Shared	Shared
Duration	Long	Long	Very-Short	Long	Short
Guarantees of .. Reliability	Normal	Normal	Very-High	Normal	Very-High
Consistency	None	None	None	None (?)	ACID
Work Pattern	Regular	Regular	Random	Random	Random
Number of Work Sources/Destination	10	100	1000	10000	100000
Provided Services	Virtual Processor	Virtual Processor	Simple Function	Simple Request	Simple or Complex Function
Performance Criteria	Throughput	Response Time	Response Time	Throughput & Response Time	Throughput & Response Time
Availability	Normal	Normal	High	High	High
Unit of Authorization	Job	User	None (?)	Request	Request

Σχήμα 2.4: Μοντέλα υπολογισμού

styles). Σε αυτό το σχήμα διακρίνονται πέντε διαφορετικά μοντέλα υπολογισμού: το μοντέλο μαζικής επεξεργασίας (batch processing), το μοντέλο διαμοίρασης του χρόνου (time sharing), το μοντέλο επεξεργασίας πραγματικού χρόνου (real time processing), το μοντέλο πελάτη - εξυπηρετητή (client - server) και το μοντέλο επεξεργασίας που είναι προσανατολισμένο στην επεξεργασία δοσοληψιών (transaction oriented processing). Επίσης φαίνονται μία σειρά από υπηρεσίες - χαρακτηριστικά μαζί με τον βαθμό με τον οποίο τα μοντέλα αυτά υποστηρίζουν ή έχουν αυτές τις υπηρεσίες - χαρακτηριστικά.

Η βασική διαφορά των συστημάτων επεξεργασίας δοσοληψιών (TP) είναι ότι έχουν μία καλά ορισμένη (βλέπε ACID ιδιότητες) μονάδα φόρτου εργασίας όπως είναι η δοσοληψία, την οποία κανένα άλλο σύστημα από τα παραπάνω δεν έχει. Ειδικά τα on-line συστήματα επεξεργασίας (OLTP) έχουν ένα μεγάλο αριθμό από πελάτες οι οποίοι μοιράζονται μία βάση δεδομένων. Η επιβάρυνση για την ανάθεση πόρων θα πρέπει να είναι μικρή διότι η μονάδα φόρτου εργασίας είναι συνήθως μικρή χωρίς όμως να αποκλείει και δοσοληψίες που είναι πολύ μεγαλύτερες. Σε αυτά τα συστήματα λόγω του ότι υπάρχουν κοινόχρηστα δεδομένα υπάρχει ανάγκη για μηχανισμούς επαναφοράς αλλά και απότρεψης από την προσπέλαση δύο δοσοληψιών σε κοινόχρηστα δεδομένα όταν μία από αυτές θέλει να πραγματοποιήσει αλλαγές.

Σε σχέση με τα άλλα συστήματα τα πραγματικού χρόνου (real-time) συστήματα έχουν αρκετές ομοιότητες με τα TP συστήματα. Συχνά άλλωστε απαιτείται και από τα TP συστήματα να παρέχουν χρονικές εγγυήσεις εξυπηρέτησης για ένα μεγάλο ποσοστό από τις δοσοληψίες που επεξεργάζονται (συνήθως κοντά στο 90%). Γι' αυτό και συχνά καλούνται και soft real-time συστήματα.

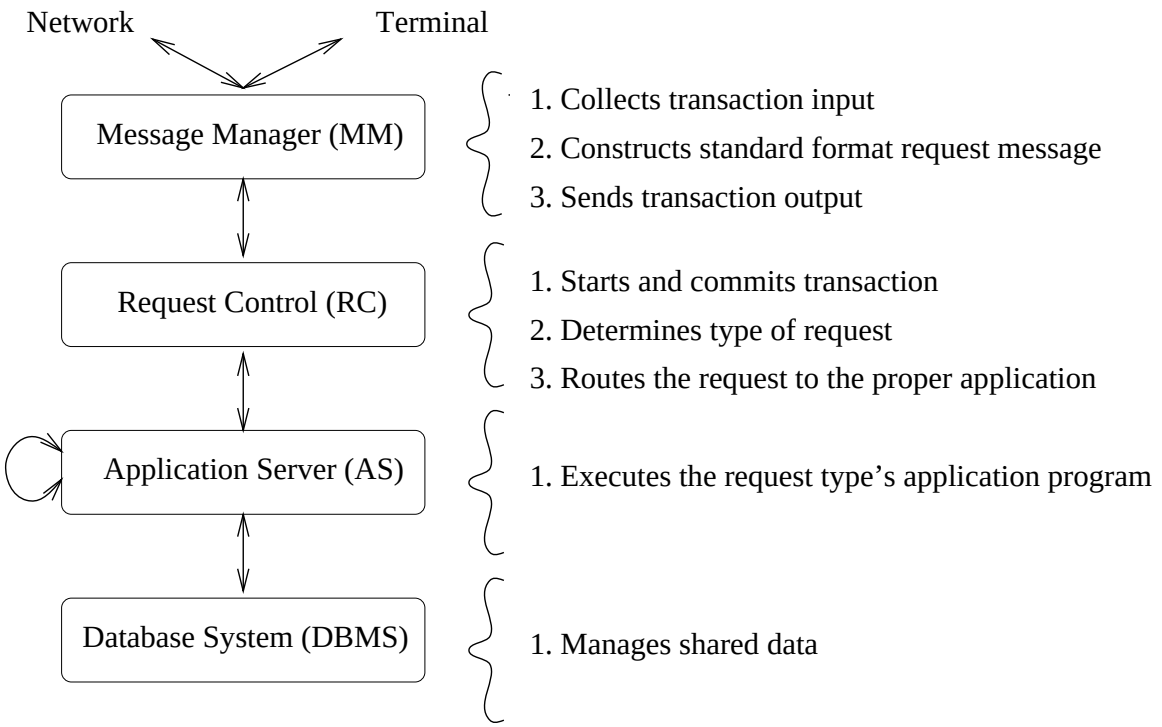
2.7 Επόπτης επεξεργασίας δοσοληψιών

Ο επόπτης επεξεργασίας δοσοληψιών (Transaction Processing Monitors ή TP monitors) [Ber90], [DGMH+93], [GR93], είναι ένα λογισμικό υποσύστημα που συχνά αποκαλείται και ως μεσάζων (middleware), το οποίο παρέχει το περιβάλλον εκτέλεσης, ανάπτυξης και επεξεργασίας εφαρμογών δοσοληψιών. Οι εφαρμογές αυτές εκτελούνται υπό την επίβλεψη του επόπτη και πραγματοποιούν ενημερώσεις, ερωτήσεις και επεξεργασία σε μία βάση δεδομένων.

Οι επόπτες δοσοληψιών είναι ενεργά συστήματα τα οποία παρέχουν υπηρεσίες για επεξεργασία εισόδου από τερματικά, διαχείριση δεδομένων, πρόσβαση σε δίκτυο, έλεγχο πρόσβασης (authorization), ασφάλεια, επανεκκίνηση των δοσοληψιών και επαναφορά (recovery) της κατάστασης του συστήματος σε συνεπή κατάσταση μετά από βλάβη, κάτι που συνεπάγεται επίβλεψη και έλεγχος εφαρμογών και πόρων από μέρους των εποπτών. Άλλωστε ο έλεγχος των πόρων του συστήματος είναι βασική αρμοδιότητα των TP monitors. Στα κατανεμημένα μάλιστα συστήματα ο έλεγχος των πόρων του συστήματος είναι πολύ κρίσιμος για την όλη απόδοση των κατανεμημένων συστημάτων. Παράδειγμα εάν κάποιος κόμβος είναι υπερφορτωμένος ο επόπτης έχει την δυνατότητα να μεταφέρει ένα μέρος από τον φόρτο εργασίας του κόμβου σε έναν άλλο. Δηλαδή μπορεί δυναμικά να παίρνει αποφάσεις οι οποίες επηρεάζουν σημαντικά την απόδοση. Παράλληλα φροντίζει και για την "τήρηση" των ACID ιδιοτήτων. Εποπτεύει δηλαδή την όλη διαδικασία παρέχοντας εγγυήσεις λειτουργίας και συγχρόνως αποκρύπτει από τις εφαρμογές που καλείται να εξυπηρετήσει, τις σχεδιαστικές λεπτομέρειες του συστήματος.

Ο επόπτης επεξεργασίας δοσοληψιών είναι φανερό ότι αποτελεί βασικό δομικό στοιχείο τέτοιων συστημάτων και η χρησιμότητά του γίνεται όλο και πιο μεγάλη όσο πιο ετερογενές και κατανεμημένο είναι το σύστημα. Προκειμένου λοιπόν να επεξεργαστεί κάποια εφαρμογή επιβάλλει μία συγκεκριμένη δομή. Έτσι λοιπόν έχει επικρατήσει οι εφαρμογές να έχουν γενικά μία δομή λειτουργιών που περιγράφεται εν συντομία από τα παρακάτω βήματα:

1. Αλληλεπίδραση με το τερματικό ή γενικά την συσκευή εισόδου προκειμένου να αποφασιστεί η είσοδος του συστήματος.
2. Επεξεργασία της εισόδου, με πιθανή μετατροπή της σε μία σειρά από τυποποιημένες αιτήσεις οι οποίες είναι αναγνωρίσιμες από το σύστημα.
3. Έναρξη της δοσοληψίας.
4. Αναγνώριση του τύπου της.
5. Εκτέλεση του αντίστοιχου προγράμματος με βάση τον αναγνωρισθέντα τύπο.
6. Επιτυχή τερματισμό της δοσοληψίας (επανάληψη εως ότου υπάρξει επιτυχής τερματισμός).
7. Αποστολή των αποτελεσμάτων πίσω στο τερματικό.



Σχήμα 2.5: Μοντέλο για τους επόπτες επεξεργασίας δοσοληψιών

Αυτή είναι σε γενικές γραμμές η δομή που έχει επικρατήσει να "απαιτούν" από τις εφαρμογές οι επόπτες επεξεργασίας δοσοληψιών. Στο σχήμα 2.5 φαίνονται οι διαφορετικές συνιστώσες οι οποίες αλληλεπιδρούν προκειμένου ο επόπτης να εξυπηρετήσει μία δοσοληψία. Αποτελώντας ουσιαστικά τον μεσάζοντα μεταξύ ενός αριθμού από διαχειριστές πόρων και των εφαρμογών, ο επόπτης θα πρέπει με την σειρά του να παρέχει μία σειρά από υπηρεσίες-μηχανισμούς. Έτσι στην ευθύνη του επόπτη είναι :

- Οι αιτήσεις που φτάνουν στο σύστημα να αντιστοιχίζονται με ένα πρόγραμμα το οποίο θα εκτελεί τις πράξεις που απαιτεί η εφαρμογή. Θα πρέπει να είναι κάτι το οποίο να είναι πιο δυναμικό από μία απλή κλήση ρουτίνας αλλά και πιο οικονομικό, από άποψη χρονικού κόστους, από την δημιουργία μίας νέας διεργασίας. Η λύση σε αυτό το πρόβλημα είναι ο μηχανισμός για κλήση μακρινής διαδικασίας (transactional remote procedure call). Με την χρήση ενός τέτοιου μηχανισμού είναι δυνατή η κλήση υπηρεσιών που απευθύνονται σε μονάδες εξυπηρετητών που βρίσκονται "μακριά" κάτι απαραίτητο σε ένα κατανεμημένο σύστημα. Με τον όρο "μακριά" εννοούμε ότι δεν βρίσκονται στον τοπικό κόμβο επεξεργασίας.
- Να είναι σε θέση να πραγματοποιεί ένα είδος εξισορρόπησης φόρτου δημιουργώντας πιθανώς κάποια νέα διεργασία που θα αναλάβει να κάνει αυτή τη εργασία.
- Να επιβάλλει έλεγχο πρόσβασης (authentication/authorization) σε κάθε αίτηση εξυπηρέτησης απαιτώντας να δίνονται τα χαρακτηριστικά του χρήστη, του τερματικού, το όνομα προγράμματος, αλλά και άλλων παραμέτρων προκειμένου να πιστοποιηθεί ότι η πρόσβαση είναι επιτρεπτή.

- Να κρατάει ένα μεγάλο αριθμό από πληροφορίες εφόσον παρακολουθεί το όλο σύστημα. Έχοντας το προνόμιο να είναι το μοναδικό υποσύστημα που κρατάει τέτοιου είδους πληροφορίες θα πρέπει να είναι σε θέση να εξυπηρετεί τις εφαρμογές που ζητούν να μάθουν κάποιες από αυτές τις πληροφορίες. Παράλληλα θα πρέπει να παρέχει και στους διαχειριστές του συστήματος αρκετές πληροφορίες για την κατάσταση του συστήματος ώστε να είναι δυνατό να γίνουν διορθωτικές κινήσεις εφόσον αυτές χρειάζονται.

- Να επιτελεί την διαχείριση δοσοληψίας (transaction management). Συντονίζει τις ενέργειες προκειμένου να τερματίσει επιτυχώς μία δοσοληψία εξασφαλίζοντας τις ιδιότητες της ατομικότητας και της μονιμότητας. Σε περίπτωση βλάβης αναλαμβάνει να επαναφέρει το σύστημα σε συνεπή κατάσταση κάνοντας χρήση του μηχανισμού καταγραφής μεταβολών (recovery mechanism). Αυτός επιτρέπει την καταγραφή των μεταβολών στην κατάσταση του συστήματος ώστε να είναι δυνατή η επαναφορά του σε συνεπή κατάσταση μετά από βλάβη.

- Μηχανισμός ελέγχου ταυτόχρονης προσπέλασης (concurrency control mechanism). Εξασφαλίζει την ιδιότητα της απομόνωσης των δοσοληψιών κατά τη πιθανή ταυτόχρονη προσπέλαση τους σε κοινόχρηστους πόρους-δεδομένα.

Οι διαχειριστές πόρων του συστήματος (όπως είναι οι διαχειριστές επικοινωνίας, βάσεων δεδομένων κ.α) κάνουν χρήση των παραπάνω υπηρεσιών. Στα κατανεμημένα συστήματα αντιστοιχεί ένα ξεχωριστό TP monitor σύστημα για κάθε κόμβο. Οπότε προκειμένου να παρέχουν στους χρήστες ένα κατανεμημένο περιβάλλον εκτέλεσης δοσοληψιών θα πρέπει να έχουν την δυνατότητα να επικοινωνούν μεταξύ τους ανταλλάσσοντας πληροφορίες για την τοπική κατάσταση του συστήματος.

2.8 Χαρακτηρισμός φόρτου εργασίας

Με τον όρο φόρτο εργασίας (workload), εννοούμε το σύνολο των μονάδων εργασίας (units of work) που καλείται να εξυπηρετήσει ένα παράλληλο ή κατανεμημένο σύστημα. Ο ποιοτικός και ποσοτικός χαρακτηρισμός του φόρτου εργασίας (workload characterization) ενός συστήματος είναι βασικός προκειμένου να γίνει οποιαδήποτε μελέτη της απόδοσης του συστήματος [YCHL92].

Η συμπεριφορά ενός πραγματικού φόρτου εργασίας είναι πολύπλοκη και είναι δύσκολο να αναπαραχθεί. Γι' αυτό είναι αναγκαία η χρήση ενός μοντέλου προκειμένου να την αναπαραστήσουμε. Ένα τέτοιο μοντέλο θα πρέπει να αναπαριστά τόσο την στατική όσο και την δυναμική συμπεριφορά του πραγματικού φόρτου και παράλληλα να είναι όσο το δυνατόν πιο συμπαγές, επαναλαμβανόμενο και ακριβές [CHS88], [Fer84], [CF86], [CS93].

Προκειμένου να κατασκευαστεί ένα τέτοιο μοντέλο, είναι αναγκαίο να γίνουν μία σειρά από βήματα - διαδικασίες. Έτσι

- Θα πρέπει να οριστεί η βασική μονάδα φόρτου εργασίας.

- Να αναλυθούν οι κατανομές των παραμέτρων του φόρτου εργασίας.
- Να γίνει δειγματοληψία στα δεδομένα του πραγματικού φόρτου ώστε να μειωθεί ο όγκος πληροφορίας.
- Να γίνει κάποιο είδος στατιστικής ανάλυσης, με σκοπό να εξαχθούν μετρήσεις που θα βοηθήσουν στη μείωση του όγκου πληροφορίας.
- Να αποφασιστεί η αντιπροσωπευτικότητα του μοντέλου.

Σκοπός αυτής της διαδικασίας είναι να δώσει ως αποτέλεσμα τα βασικά χαρακτηριστικά του πραγματικού φόρτου εργασίας σε ένα μοντέλο, όπου έχει γίνει σε πολύ μεγάλο βαθμό μείωση της πληροφορίας και χρήση τυπικών αντιπροσώπων για κάθε διαφορετική ομάδα φόρτου εργασίας που αναγνωρίζεται. Με αυτόν τον τρόπο είναι δυνατή η κατάταξη κάθε μονάδας φόρτου σε μία ομάδα, ώστε κατά την εμφάνισή της στο σύστημα, να είναι δυνατό να χρησιμοποιηθεί ένας αλγόριθμος δρομολόγησης ο οποίος (βάση της κατάταξής της σε κάποια ομάδα) θα την δρομολογήσει στον πιο κατάλληλο κόμβο επεξεργασίας.

Στα συστήματα επεξεργασίας δοσοληψιών, με τα οποία ασχολούμαστε, οι μονάδες φόρτου εργασίας είναι οι δοσοληψίες. Αυτές προσπελαύνουν κοινόχρηστα δεδομένα με αυστηρούς περιορισμούς συνέπειας και διαθεσιμότητας. Η δοσοληψία, ως μονάδα φόρτου, διαφέρει σημαντικά από την έννοια της διεργασίας που υποστηρίζει ένα λειτουργικό σύστημα. Η διαφορά αυτή κυρίως έγκειται στις ιδιότητες ACID που θα πρέπει να εξασφαλίζει το σύστημα.

Οι απαιτήσεις των δοσοληψιών σε πόρους του συστήματος είναι μεταβλητές. Παρόλα αυτά είναι δυνατό, με χρήση των τεχνικών που προαναφέρθηκαν, να ταξινομηθούν σε κλάσεις λαμβάνοντας υπόψη ομοιότητες που παρατηρούνται στις απαιτήσεις πόρων αλλά και στην κατανομή των προσπελάσεων που πραγματοποιούν στα δεδομένα του συστήματος. Η συγγένεια στα δεδομένα (data affinity) [YD94b] μεταξύ των δοσοληψιών αποτελεί μία ακόμα βασική διαφορά τους σε σχέση με τις διεργασίες.

Η ταξινόμηση αυτή είναι γνωστή ως **ομαδοποίηση του φόρτου εργασίας** (workload clustering) [Λαμ95a], [Κλο97]. Η ομαδοποίηση αποσκοπεί στην εύρεση χαρακτηριστικών ομάδων με παρόμοιο φόρτο εργασίας και η μετέπειτα χρησιμοποίησή τους προκειμένου να μελετηθεί ένα σύστημα. Ο αριθμός των ομάδων αυτών είναι από μερικές δεκάδες ως μερικές εκατοντάδες. Με χρήση μίας τέτοιας γνώσης είναι δυνατή η εξισορρόπηση του φόρτου εργασίας ανάμεσα στους υπολογιστικούς κόμβους του συστήματος, βελτιώνοντας έτσι την απόδοση του συστήματος. Στις μελέτες [YD92a], [YD94a], φαίνεται η επίδραση που έχει η ομαδοποίηση του φόρτου εργασίας και η εξισορρόπηση του φόρτου μεταξύ των υπολογιστικών κόμβων για τις διάφορες αρχιτεκτονικές σύζευξης υπολογιστικών κόμβων που είδαμε σε προηγούμενη παράγραφο.

Η γνώση των χαρακτηριστικών του φόρτου εργασίας και η ομαδοποίησή τους σε κλάσεις με παρόμοια χαρακτηριστικά είναι πολύ σημαντική για την επιτυχία αλγορίθμων δρομολόγησης δοσοληψιών [NLB+97]. Η επιτυχία τους συνίσταται στη βελτίωση της απόδοσης του συστήματος. Η απόδοση του συστήματος μετριέται από μία σειρά στόχων επίδοσης. Γι' αυτό πριν αναφερθούμε στους αλγορίθμους δρομολόγησης θα δούμε ορισμένους από τους βασικούς στόχους επίδοσης που έχουν επικρατήσει.

2.9 Στόχοι επίδοσης

Είναι πλέον συνηθισμένο φαινόμενο σήμερα οι χρήστες ενός συστήματος να απαιτούν από τον προμηθευτή του συστήματος ένα είδος συμφωνίας για την απόδοση του συστήματος κάτω από συγκεκριμένο φόρτο εργασίας (service level agreement). Ειδικά αυτό είναι πολύ συνηθισμένο σε εφαρμογές που κάνουν χρήση εικόνας και ήχου, όπου το σύστημα θα πρέπει να παρέχει μια πολύ καλή ποιότητα εξυπηρέτησης. Αντίθετα με τον χρήστη, ο υπεύθυνος για την διαχείριση του συστήματος επιδιώκει την εξισορρόπηση της χρήσης των πόρων του συστήματος ώστε να έχει το μέγιστο δυνατό κέρδος από την χρησιμοποίησή τους.

Η εξισορρόπηση της χρήσης των πόρων του συστήματος ουσιαστικά αποτελεί μία προσπάθεια ελαχιστοποίησης της μέγιστης χρήσης των πόρων του συστήματος. Εναλλακτικά είναι δυνατό να γίνεται ελαχιστοποίηση του κόστους επικοινωνίας στο σύστημα.

Οι χρήστες από την άλλη μεριά συνήθως ζητούν περιορισμούς για τις μονάδες φόρτου που εισαγάγουν στο σύστημα. Η ελαχιστοποίηση της απόκρισης του συστήματος για το σύνολο του φόρτου εργασίας είναι ένα τυπικός στόχος που θέτουν οι χρήστες. Αυτό συνήθως αφορά μονάδες φόρτου που έχουν παρόμοια χαρακτηριστικά ή μονάδες στις οποίες οι χρήστες δεν απαιτούν συγκεκριμένους περιορισμούς για την εκτέλεσή τους. Σε αντίθετη περίπτωση μπορεί οι χρήστες να προσδιορίσουν δικούς τους στόχους για κάθε μονάδα εργασίας ή ανάγκες για επίπεδο παροχής υπηρεσιών (quality of service), χωρίς να καθορίζεται ο τρόπος που θα επιτευχθούν αυτοί. Τέτοια παραδείγματα [ESP], [GHK+95] είναι τα εξής:

- **Στόχοι προθεσμίας (Deadline)** : Ορίζεται το μέγιστο χρονικό διάστημα κατά το οποίο θα πρέπει να αποπερατωθεί η κάθε μονάδα φόρτου εργασίας.
- **Στόχοι μέσου χρόνου απόκρισης (Average response time goal)** : Ορίζεται ένα επιθυμητό άνω όριο για το μέσο χρόνο απόκρισης για την κάθε κλάση φόρτου εργασίας.
- **Στόχοι για την κατανομή του χρόνου απόκρισης (Percentile response time goal)** : Σε αυτούς ορίζεται ότι ο χρόνος απόκρισης θα πρέπει για ένα ποσοστό μιας κλάσης του φόρτου εργασίας να είναι μικρότερος από ένα επιθυμητό όριο.
- **Στόχοι ρυθμού εξυπηρέτησης (Service rate goal)** : Σε αυτήν την κατηγορία ορίζεται ένα ποσοστό χρήσης ενός πόρου σαν ελάχιστη απαίτηση για μια κλάση μονάδας φόρτου. Με αυτόν τρόπο διασφαλίζεται ότι ο ρυθμός αποπεράτωσής τους θα είναι ικανοποιητικός. Τέτοιοι στόχοι διατυπώνονται σε περιπτώσεις όπου δεν είναι εύκολο να διατυπωθεί κάποιος άλλος στόχος λόγω π.χ. έλλειψης γνώσης για τις απαιτήσεις της μονάδας φόρτου εργασίας.

Προκειμένου να ικανοποιηθούν οι στόχοι επίδοσης για κάθε κλάση φόρτου εργασίας το σύστημα πρέπει να κάνει χρήση των διαφόρων μηχανισμών διαχείρισης πόρων. Τέτοιοι μηχανισμοί είναι για παράδειγμα οι αλγόριθμοι δρομολόγησης δοσοληψιών, οι αλγόριθμοι χρονοπρογραμματισμού επεξεργαστή και δίσκων, οι αλγόριθμοι διαχείρισης ενταμιευτών καθώς και μια σειρά από άλλους αλγόριθμους που αναλαμβάνουν την διαχείριση των πόρων

του συστήματος. Το σύστημα έχει την δυνατότητα, λαμβάνοντας υπόψη τους στόχους επίδοσης για κάθε κλάση και σε συνάρτηση με την εκάστοτε κατάσταση του συστήματος, να μεταβάλει μία σειρά από παραμέτρους αυτών των αλγορίθμων, δίνοντας την δυνατότητα και ανάλογα με την πολιτική που υλοποιεί, να βελτιώνει την συμπεριφορά του συστήματος όσο αφορά την ικανοποίηση των στόχων επίδοσης.

Είναι πιθανό οι στόχοι που έχουν οριστεί να είναι ανέφικτο να ικανοποιηθούν, είτε επειδή οι χρήστες δώσαν μικρούς στόχους επίδοσης μη λαμβάνοντας υπόψη τις ανάγκες των κλάσεων, είτε επειδή το σύστημα είναι υπερφορτωμένο και δεν μπορεί να τους ικανοποιήσει. Σε κάθε περίπτωση χρειάζεται ειδική μεταχείριση διότι προσπαθώντας το σύστημα να ικανοποιήσει τέτοιους στόχους, είναι δυνατό να οδηγήσει το σύστημα σε μεγάλη μείωση της γενικής απόδοσης του. Σε αυτές τις περιπτώσεις καλό είναι να έχουν οριστεί στο σύστημα ποιές είναι κρίσιμες κλάσεις και ποιές όχι, ώστε να τούς δοθούν οι κατάλληλες προτεραιότητες.

Οι τεχνικές ικανοποίησης στόχων επίδοσης βασίζονται σε ένα σχήμα ανάδρασης (feedback). Το σύστημα παρακολουθεί την εξέλιξη των στόχων επίδοσης στο χρόνο και σε τακτά συνήθως χρονικά διαστήματα λαμβάνει μια σειρά από διορθωτικές (εάν χρειάζεται) παρεμβάσεις στις παραμέτρους ελέγχου, προκειμένου να βελτιώσει την επίδοση ορισμένων κλάσεων φόρτου εργασίας που η συμπεριφορά τους αποκλίνει από την επιθυμητή.

Στην βιβλιογραφία έχουν αρχίσει να κάνουν την εμφάνιση τους μηχανισμοί που λαμβάνουν υπόψη τους τους στόχους επίδοσης για τις κλάσεις φόρτου εργασίας. Κυρίως είναι αλγόριθμοι δρομολόγησης δοσοληψιών [FNGD93a], [FNGD93b]. Όμως υπάρχουν και μηχανισμοί διαχείρισης μνήμης [CFW+94], [BCL93] αλλά και χρονοπρογραμματισμού του επεξεργαστή [BGTV90].

Στον μηχανισμό διαχείρισης ενταμιευτών [CFW+94], έχουμε την περιοδική ενεργοποίηση ενός αλγορίθμου ο οποίος αναλαμβάνει να μεταβάλει το μέγεθος των ενταμιευτών στους κόμβους του συστήματος, διατηρώντας το συνολικό τους μέγεθος τους σταθερό. Για τον λόγο αυτό το σύστημα παρακολουθεί, για κάθε ενταμιευτή ξεχωριστά, το κατά πόσο ικανοποιούνται οι στόχοι επίδοσης που έχουν τεθεί για τις διάφορες κλάσεις και ανάλογα προβαίνει σε μεταβολές ώστε να βελτιώσει την κατάσταση.

Ο δεύτερος μηχανισμός διαχείρισης ενταμιευτών [BCL93] ελέγχει την πιθανότητα η σελίδα που χρειάζεται μία προσπέλαση να βρίσκεται στον ενταμιευτή. Αυτό το επιτυγχάνει (με χρήση των στόχων επίδοσης) ορίζοντας κάτω φράγματα για το πλήθος των σελίδων από κάθε τμήμα της βάσης δεδομένων που βρίσκονται στους ενταμιευτές. Στη συνέχεια στην αναφορά [BMCL94] γίνεται μία επέκταση όπου παράλληλα ρυθμίζεται ο βαθμός πολυπρογραμματισμού του συστήματος. Γίνεται επίσης διάκριση μεταξύ κλάσεων που ο χρόνος απόκρισής τους εξαρτάται από το μέγεθος των ενταμιευτών και κλάσεων όπου ο χρόνος απόκρισής τους εξαρτάται κυρίως από τη μνήμη.

2.10 Αλγόριθμοι δρομολόγησης

Ένα βασικό πρόβλημα για τα παράλληλα και κατανεμημένα συστήματα επεξεργασίας δοσοληψιών είναι το πώς θα γίνει σωστή χρήση όλων των υπολογιστικών κόμβων του συστήματος, ώστε να επιτευχθούν υψηλοί ρυθμοί εξυπηρέτησης και μικροί χρόνοι απόκρισης. Ουσιαστικά δηλαδή είναι πρόβλημα ανάθεσης του φόρτου εργασίας στους κόμβους του συστήματος [Rah92a]. Αυτήν την εργασία αναλαμβάνει να κάνει σε ένα σύστημα επεξεργασίας δοσοληψιών ο δρομολογητής δοσοληψιών (transaction router). Αποφασίζει ποιός κόμβος θα πρέπει να επεξεργαστεί μία νέα αίτηση εξυπηρέτησης (δοσοληψία) που φτάνει στο σύστημα.

Η εξισορρόπηση του φόρτου είναι κάτι αρκετά πολύπλοκο σε ένα κατανεμημένο σύστημα. Πολλοί παράγοντες επηρεάζουν τον φόρτο κάθε κόμβου όπως : η συχνότητα επικοινωνίας, η συχνότητα με την οποία ο κόμβος ζητά πρόσβαση σε δίσκους, ο ανταγωνισμός για τα δεδομένα (data contention) εξαιτίας ταυτόχρονης πρόσβασης κ.α.

Στην [YD94b] ορίζεται η έννοια της συγγένεια (affinity) μίας κλάσης δοσοληψιών ως προς μία σχέση του σχεσιακού μοντέλου της βάσεως δεδομένων, ως το ποσοστό των δεδομένων της σχέσης αυτής για τα οποία γίνεται αναφορά από τη κλάση δοσοληψιών. Π.χ. εάν το 60% των προσπελάσεων μίας κλάσης δοσοληψιών γίνεται σε μία σχέση Α, τότε λέμε ότι αυτή η κλάση δοσοληψιών έχει συγγένεια 0.6 με την σχέση Α. Για την Shared Nothing (παράγραφος 3.1.1) αρχιτεκτονική μπορούμε αντίστοιχα να ορίσουμε και ως συγγένεια με κάποιο κόμβο, το ποσοστό των προσπελάσεων που γίνονται στον συγκεκριμένο κόμβο από μία κλάση δοσοληψιών. Οι προσπελάσεις αυτές αφορούν πλέον μία συλλογή από σχέσεις της βάσης δεδομένων.

Στην αρχιτεκτονική Shared Nothing (SN) οι αλγόριθμοι δρομολόγησης πρέπει να λαμβάνουν υπόψη ότι οι δοσοληψίες παρουσιάζουν μια συγγένεια με κάποιους κόμβους (site affinity). Αυτό συμβαίνει διότι στη SN αρχιτεκτονική τα δεδομένα είναι μοιρασμένα στους κόμβους του συστήματος. Έτσι μία δοσοληψία που εξυπηρετείται σε έναν συγκεκριμένο κόμβο, όταν θέλει ένα δεδομένο που βρίσκεται σε έναν άλλο κόμβο πρέπει να κάνει μία μακρινή αίτηση (remote request) σε αυτόν τον κόμβο. Η αίτηση αυτή έχει μεγάλο χρονικό κόστος οπότε είναι φανερό ότι κόμβοι που ελαχιστοποιούν την πιθανότητα τέτοιων αιτήσεων, μειώνουν την ανάγκη και για επικοινωνία και γενικά το χρόνο απόκρισης αυτών των δοσοληψιών. Σε ορισμένες περιπτώσεις όμως η εξισορρόπηση του φόρτου είναι πολύ δύσκολη λόγω της διαμέρισης που έχει γίνει στα δεδομένα αλλά και της φύσης ορισμένων αιτήσεων όπου αναγκαστικά κάποιος κόμβος υπερφορτώνεται (scan operations).

Στην αρχιτεκτονική Shared Disk (SD) τα δεδομένα είναι κοινόχρηστα (παράγραφος 3.1.2). Παρόλα αυτά οι χρόνοι απόκρισης των δοσοληψιών εξαρτώνται από τα δεδομένα που αποθηκεύονται ιδιωτικά σε κάθε κόμβο. Δημιουργούνται έτσι κόμβοι όπου οι χρόνοι πρόσβασης στα δεδομένα είναι πολύ πιο μικροί από τους άλλους κόμβους. Η συγγένεια εδώ είναι χρονικά εξαρτώμενη, εξαιτίας του φαινομένου της ακύρωσης δεδομένων στους τοπικούς ενταμιευτές, πράγμα που περιπλέκει ακόμα περισσότερο την κατάσταση. Στην περίπτωση της SD αρχιτεκτονικής είναι πιο εύκολο να επιτευχθεί μια καλή εξισορρόπηση, θυσιάζοντας όμως στο βωμό της εξυπηρέτησης την απόκριση του συστήματος διότι

ενισχύεται το φαινόμενο της ακύρωσης σελίδων αποθηκευμένων στους ενταμιευτές.

Στην αναφορά [YD94a] γίνεται μια παρουσίαση της επίπτωσης που έχει η συγγένεια δεδομένων (data affinity) στην απόδοση διαφόρων αρχιτεκτονικών σύζευξης υπολογιστικών κόμβων χρησιμοποιώντας μία ανάλυση μόνιμης κατάστασης (steady state analysis) ενώ στην αναφορά [YD94b] μελετάται η απόδοση των ίδιων συστημάτων σε περιπτώσεις διακύμανσης του φόρτου (load surge) και βλάβης κόμβων.

Να σημειωθεί ότι στην διεθνή βιβλιογραφία υπάρχει ένα μεγάλο πλήθος από αλγόριθμους για την αρχιτεκτονική SN, ενώ για τις αρχιτεκτονικές κοινόχρηστων δεδομένων δεν υπάρχουν ανάλογες δημοσιεύσεις. Στη συνέχεια θα δούμε συνοπτικά τις κύριες κατηγορίες αλγορίθμων δρομολόγησης μαζί με ορισμένα από τα άρθρα στα οποία έχουν γίνει αναφορές για τα αποτελέσματα που είχε η χρησιμοποίηση τους. Η κατηγοριοποίησή τους βασίζεται στους αλγορίθμους για SN συστήματα επεξεργασίας δοσοληψιών [ESP].

Καταρχήν, μία βασική διάκριση των αλγορίθμων βασίζεται στο γεγονός της ύπαρξης ενός κεντρικοποιημένου δρομολογητή ή πολλών κατανεμημένων. Στα σημερινά συστήματα για την online επεξεργασία δοσοληψιών έχει επικρατήσει ο ένας και μοναδικός δρομολογητής που αναλαμβάνει να αποφασίσει πού θα γίνει η επεξεργασία κάθε δοσοληψίας. Στην περίπτωση του ενός δρομολογητή υπάρχουν συγκεντρωμένες σε ένα σημείο όλες οι πληροφορίες που χρειάζεται ο δρομολογητής προκειμένου να κάνει την επιλογή του. Αντίθετα στην περίπτωση όπου υπάρχουν πολλοί κατανεμημένοι δρομολογητές, τότε απαιτείται να υπάρχει επικοινωνία μεταξύ τους ώστε να επιλεχθεί ο πιο κατάλληλος κόμβος.

2.10.1 Στατικοί αλγόριθμοι

Οι αλγόριθμοι αυτοί κάνουν χρήση ενός στατικού πίνακα δρομολόγησης και συνεπώς προσθέτουν ελάχιστη επιβάρυνση κατά την διαδικασία δρομολόγησης. Ο πίνακας αυτός βασίζεται στον τρόπο διαμέρισης των δεδομένων στους διάφορους κόμβους όταν πρόκειται για SN αρχιτεκτονική ή στη συγγένεια που έχουν μεταξύ τους οι δοσοληψίες, προκειμένου δρομολογώντας τες σε συγκεκριμένους κόμβους, να αυξήσουν την πιθανότητα ο ενταμιευτής να έχει τα δεδομένα που ζητούν οι κλάσεις (SD αρχιτεκτονική). Δύο βασικές κατηγορίες αλγορίθμων δρομολόγησης μπορούμε να διακρίνουμε :

1. *Αιτιοκρατικοί (deterministic) αλγόριθμοι* όπου ολόκληρη η κλάση ανατίθεται σε έναν συγκεκριμένο κόμβο [CDY86].
2. *Πιθανοκρατικοί (probabilistic) αλγόριθμοι* όπου ένα ποσοστό του συνολικού φόρτου εργασίας ανατίθεται σε κάθε κόμβο [YCDT86], [YCDT89].

2.10.2 Δυναμικοί αλγόριθμοι

Οι δυναμικοί αλγόριθμοι δρομολόγησης είναι ιδανικοί σε περιπτώσεις όπου οι καταστάσεις μεταβάλλονται (αλλαγή στην δυναμική του φόρτου εργασίας, βλάβη κόμβου κ.τ.λ) ενώ οι στατικοί αλγόριθμοι δεν είναι σε θέση να δώσουν λύση σε τέτοια προβλήματα.

Βέβαια, προσθέτουν μια επιβάρυνση σε σχέση με τους στατικούς αλγόριθμους η οποία ποικίλει ανάλογα με το είδος της πληροφορίας που διαχειρίζονται και η οποία προκύπτει από την κατάσταση του συστήματος. Μπορούμε να διακρίνουμε τις εξής κατηγορίες δυναμικών αλγορίθμων δρομολόγησης :

1. *Δυναμικοί αλγόριθμοι βασιζόμενοι στο μέγεθος ουράς* [YBL88].
2. *Δυναμικοί αλγόριθμοι βασιζόμενοι στην ιστορία δρομολόγησης (routing history)* [YBL88].
3. *Παλινδρομικοί (regression based adaptive) αλγόριθμοι* [YLL91], [LYL88].
4. *Αλγόριθμοι με ανάδραση (feedback)* [WHMZ93].

2.10.3 Αλγόριθμοι προσανατολισμένοι στους στόχους επίδοσης (goal oriented)

Οι αλγόριθμοι αυτοί βασίζονται στην ικανοποίηση στόχων επίδοσης που έχουν δοθεί για τις μονάδες φόρτου του συστήματος. Είναι προσανατολισμένοι προς τους στόχους επίδοσης που έχουν δοθεί από το χρήστη ή από τον ίδιο τον διαχειριστή του συστήματος. Βασικές αναφορές για goal oriented αλγόριθμους μπορεί κανείς να βρει στις αναφορές [FNGD93a], [FNY92]. Είναι σημαντικό να πούμε ότι οι αλγόριθμοι αυτοί μπορούν να βελτιώσουν σημαντικά το βαθμό εξυπηρέτησης μέσα στα ζητούμενα χρονικά πλαίσια για τις κλάσεις δοσοληψιών που εξυπηρετεί το σύστημα.

2.10.4 Μικροοικονομικοί αλγόριθμοι

Οι μικροοικονομικοί αλγόριθμοι δρομολόγησης [FNY89], [FNY92], [Ανα96], είναι μία ειδική περίπτωση αλγορίθμων δρομολόγησης αφού η πολιτική δρομολόγησης που εφαρμόζουν βασίζεται σε ανταγωνιστικά οικονομικά κριτήρια. Σε αυτούς του αλγόριθμους, κόμβοι και δοσοληψίες ανταγωνίζονται μεταξύ τους, σε μικροοικονομικό επίπεδο, προκειμένου να κάνουν χρήση των πόρων του συστήματος. Βασικό στοιχείο διαφοροποίησης αποτελεί ο ανταγωνισμός και όχι η συνεργασία μεταξύ των μερών του συστήματος.

Κεφάλαιο 3

Κατανεμημένη επεξεργασία δοσοληψιών

Οι λόγοι που ώθησαν την επεξεργασία δοσοληψιών στο να γίνεται κατανεμημένα είναι λόγοι κόστους, διαθεσιμότητας, χωριτικότητας (capacity). Πιο συγκεκριμένα τα κατανεμημένα συστήματα επεξεργασίας δοσοληψιών έχουν τα εξής πλεονεκτήματα έναντι των κεντροποιημένων συστημάτων :

- **Τοπική αυτονομία**, διότι είναι δυνατό να χρησιμοποιηθεί ένας οποιοσδήποτε υπολογιστικός κόμβος (με πιθανή αντιγραφή δεδομένων) για την ίδια δουλειά, χωρίς να είναι ανάγκη να γίνει μεταβίβαση αίτησης προσπέλασης. Βέβαια για να υπάρχει πλήρης τοπική αυτονομία θα πρέπει κάθε κόμβος να έχει όλη την βάση τοπικά (ή αντίγραφό της) γεγονός που δεν είναι οικονομικά συμφέρον.
- **Βελτιωμένη επίδοση**, εφόσον σε ένα γεωγραφικά και υπολογιστικά κατανεμημένο σύστημα (π.χ. τραπεζικό σύστημα) δεν είναι ανάγκη οι αιτήσεις εξυπηρέτησης να προωθούνται σε έναν υπολογιστικό κόμβο που βρίσκεται κάπου κεντρικά και χρειάζεται να έχει πολύ μεγάλες υπολογιστικές δυνατότητες.
- **Διαθεσιμότητα**, διότι εφόσον η επεξεργασία γίνεται κατανεμημένα το όλο σύστημα δεν εξαρτάται από ένα μηχάνημα και μόνο. Έτσι ακόμα και εάν ένα μηχάνημα παρουσιάσει βλάβη, το σύστημα συνεχίζει να δουλεύει κανονικά.
- **Επεκτασιμότητα**, αφού αρκεί να προστεθούν δίσκοι και επεξεργαστές στο δίκτυο ώστε να επεκταθεί ένα τέτοιο σύστημα. Είναι ίσως και το πιο βασικό πλεονέκτημα των κατανεμημένων συστημάτων. Επιτρέπει έτσι με ελάχιστο κόστος την προσθήκη υπολογιστικών κόμβων. Το κόστος αυτό είναι ουσιαστικά η εισαγωγή των νέων κόμβων στο δίκτυο και η ενδεχόμενη ανάγκη για επαναπροσδιορισμό ορισμένων παραμέτρων που σχετίζονται λόγου χάρη με τη δρομολόγηση των δοσοληψιών. Τα κατανεμημένα συστήματα επεκτείνονται εύκολα συμπεριλαμβάνοντας ακόμα και χιλιάδες υπολογιστικούς κόμβους.
- **Οικονομία**, διότι δεν είναι ανάγκη να χρησιμοποιηθούν ακριβές συσκευές τόσο για μονάδες I/O όσο και για επεξεργαστές αφού μπορούν να χρησιμοποιηθούν πιο

πολλές συσκευές με πιο περιορισμένες δυνατότητες (επεξεργάζονται μέρος μόνο του συνολικού αριθμού των αιτήσεων) και συνεπώς πιο μικρό κόστος. Σε ένα γεωγραφικά κατανεμημένο σύστημα οικονομία γίνεται και στην επικοινωνία αφού δεν χρειάζεται επικοινωνία για κάθε αίτηση.

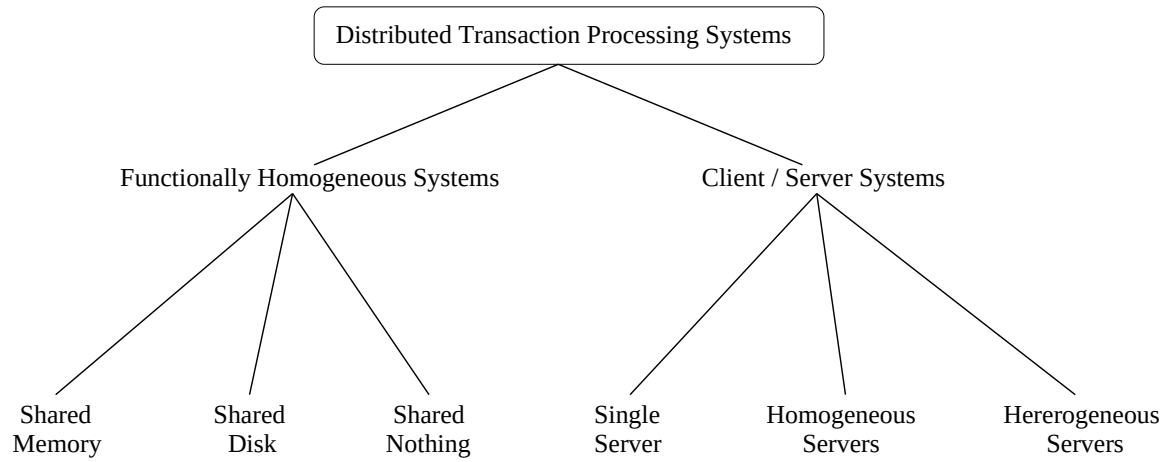
Συγχρόνως όμως νέα προβλήματα εμφανίστηκαν ή παλιά προβλήματα διογκώθηκαν και σαφώς επηρεάζουν την απόδοση των κατανεμημένων συστημάτων. Τα σημαντικότερα από αυτά είναι :

- Η **πολυπλοκότητα**, αφού τα κατανεμήμένα συστήματα αποτελούνται εν γένει από ετερογενή συστήματα, που κάνει την διαχείριση τους αρκετά δύσκολη. Πολλές είναι οι παράμετροι που επηρεάζουν τα κατανεμημένα συστήματα και χρειάζεται πολύ προσοχή αλλά και εμπειρία για την κατάλληλη επιλογή τους.
- Ο **συγχρονισμός και ο συντονισμός** που απαιτείται αφού τα δεδομένα είναι κοινόχρηστα και θα πρέπει η πρόσβαση σε αυτά να γίνεται με τέτοιο τρόπο ώστε να μην οδηγείται το σύστημα σε κατάσταση που δεν είναι συνεπής.
- Η **ασφάλεια**, η οποία σε αυτά τα συστήματα είναι δύσκολο να την εγγυηθεί κανείς. Η μη κεντροποιημένη φύση των κατανεμημένων συστημάτων δίνει την δυνατότητα ελέγχου από πολλά σημεία. Ακόμα και αν αυτός ο έλεγχος είναι σχετικά περιορισμένος στα περισσότερα από τα σημεία του συστήματος δεν μπορεί να διασφαλίσει το συστήμα από μη εξουσιοδοτημένες προσβάσεις.
- Το **κόστος για επικοινωνιακό υλικό και υπηρεσίες**, που αποτελεί μεγάλο ποσοστό του κόστους τους. Συχνά απαιτείται η σύνδεση υπολογιστικών κόμβων που βρίσκονται σε απόσταση εκατοντάδων χιλιόμετρων με παράλληλη απαίτηση για μικρή σχετικά καθυστέρηση.

Τα κατανεμημένα συστήματα φαίνονται σήμερα ως η λύση για τα συστήματα επεξεργασίας δοσοληψιών. Μόνο αυτά τα συστήματα έχουν αποδείξει, με τα σημερινά δεδομένα, πως μπορούν να καλύψουν τις ανάγκες που δημιουργούνται και που χρόνο με τον χρόνο παρουσιάζουν έντονα αυξητικές τάσεις. Μεγάλο μέρος της έρευνας τα τελευταία χρόνια επικεντρώνεται στην εύρεση αρχιτεκτονικών σύζευξης των υπολογιστικών κόμβων αλλά και αλγορίθμων ώστε να βελτιωθεί η απόδοσή τους.

3.1 Αρχιτεκτονικές σύζευξης υπολογιστικών κόμβων

Σε αυτή την ενότητα θα δούμε τις διάφορες αρχιτεκτονικές σύζευξης υπολογιστικών κόμβων. Οι αρχιτεκτονικές αυτές έχουν αποκτήσει τεράστιο ενδιαφέρον διότι αποτελούν, τουλάχιστον με τα σημερινά δεδομένα, την καλύτερη λύση στο πρόβλημα της ικανοποίησης της ολοένα και μεγαλύτερης ζήτησης για υψηλό ρυθμό εξυπηρέτησης, με παράλληλη απαίτηση για επεκτασιμότητα, διαθεσιμότητα και μικρότερο δυνατό κόστος. Καταρχήν μπορούμε να διακρίνουμε δύο βασικές κατηγορίες:



Σχήμα 3.1: Βασικές κατηγορίες στα κατανεμημένα συστήματα επεξεργασίας δοσοληψιών

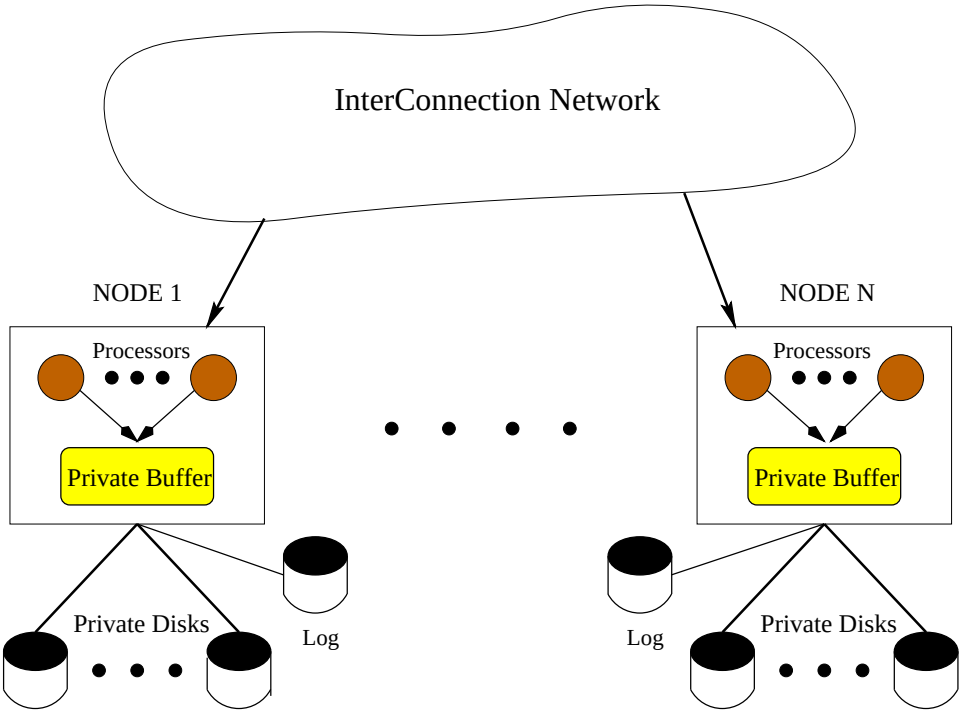
- Ομογενείς (homogeneous) αρχιτεκτονικές, όπου κάθε κόμβος έχει την ίδια ικανότητα ως προς την επεξεργασία δοσοληψιών.
- Ετερογενείς (heterogeneous) αρχιτεκτονικές, όπως είναι τα μοντέλα πελάτη / εξυπηρετητή (client-server) όπου ο ρόλος των κόμβων δεν είναι ίδιος.

Με την αρχιτεκτονική των client-server μοντέλων δεν θα ασχοληθούμε παραπάνω από την παράθεση του σχήματος και των αρχιτεκτονικών που εντάσσονται σε αυτήν την κατηγορία (σχήμα 3.1).

Αντίθετα θα ασχοληθούμε αναλυτικά με την κατηγορία των ομογενών αρχιτεκτονικών, όπου όλοι οι κόμβοι του συστήματος, επιτελούν λειτουργικά τις ίδιες εργασίες (σε αντίθεση με τα συστήματα που υιοθετούν την αρχιτεκτονική των client-server μοντέλων) και παράλληλα είναι ευρύτατα διαδεδομένα για την επεξεργασία δοσοληψιών. Σε αυτήν εντάσσονται οι αρχιτεκτονικές shared nothing, shared disk, shared everything καθώς και η αρχιτεκτονική shared intermediate memory η οποία όπως θα δούμε είναι υποπερίπτωση της shared disk αρχιτεκτονικής. Στη συνέχεια θα αναφερθούμε στις αρχιτεκτονικές αυτές και θα παρουσιάσουμε τις εργασίες που έχουν γίνει σε αυτό το πεδίο των κατανεμημένων συστημάτων με έμφαση στις αρχιτεκτονικές shared disk και shared intermediate memory, καθώς η αρχιτεκτονική shared nothing έχει περιγραφεί αναλυτικά [Μαρ95b], ενώ η αρχιτεκτονική shared everything δεν αποτελεί μέρος της παρούσας εργασίας.

3.1.1 Αρχιτεκτονική Shared Nothing

Στην αρχιτεκτονική αυτή (σχήμα 3.2), οι υπολογιστικοί κόμβοι έχουν έναν αριθμό από ιδιωτικούς δίσκους καθώς και έναν ιδιωτικό ενταμιευτή όπου αποθηκεύονται σελίδες της βάσης δεδομένων, προκειμένου να μειωθεί η ανάγκη για προσπέλαση στους δίσκους (οι δίσκοι ως γνωστόν, είναι ένα αποθηκευτικό μέσο που αν και γρήγορο σαν αποθηκευτικό μέσο, είναι πολύ αργό σε σχέση με τους επεξεργαστές). Η βάση δεδομένων είναι λοιπόν μοιρασμένη ανάμεσα στους κόμβους του συστήματος βάσει ενός αλγορίθμου διαμέρισης



Σχήμα 3.2: Αρχιτεκτονική Shared Nothing

(declustering ή partitioning όπως αλλιώς λέγεται) των δεδομένων. Εφόσον τα δεδομένα δεν είναι όλα τοπικά, χρειάζεται ένας μηχανισμός για να μπορεί ένας κόμβος να έχει πρόσβαση σε δεδομένα που είναι αποθηκευμένα σε άλλους κόμβους. Ο μηχανισμός μεταβίβασης αίτησης προσπέλασης (function request shipping) όπως ονομάζεται, επιτρέπει σε μία δοσοληψία να προσπελάσει δεδομένα που βρίσκονται αποθηκευμένα σε διαφορετικό κόμβο από τον κόμβο που γίνεται η εξυπηρέτηση της δοσοληψίας. Προκειμένου στη συνέχεια να γίνουν μόνιμες οι αλλαγές που επέφερε η δοσοληψία σε έναν αριθμό από κόμβους απαιτείται η χρήση ενός πρωτοκόλλου δέσμευσης δύο φάσεων (two-phase commit protocol) ώστε να διαφυλαχθεί η 'all or nothing' ιδιότητα των δοσοληψιών. Η επικοινωνία μεταξύ των κόμβων γίνεται με χρήση μηνυμάτων.

Τα πλεονεκτήματα της αρχιτεκτονικής αυτής συνοψίζονται στα εξής:

- Στην επεκτασιμότητα αυτής της αρχιτεκτονικής, η οποία είναι το βασικό της πλεονέκτημα. Το σύστημα μπορεί να επεκταθεί ώστε να περιλαμβάνει χιλιάδες υπολογιστικούς κόμβους. Άλλωστε υπάρχουν και σήμερα συστήματα που υποστηρίζουν τέτοιους αριθμούς από κόμβους με επιτυχία.
- Στο ότι οι κόμβοι μπορούν να είναι γεωγραφικά απομακρυσμένοι.
- Στο ότι είναι δυνατό να τρέχουν διαφορετικό λειτουργικό σύστημα.
- Στην αυξημένη αξιοπιστία και διαθεσιμότητα αυτής της αρχιτεκτονικής ειδικά με χρήση τη τεχνικών όπως: η αντιγραφή δεδομένων που βρίσκονται σε έναν κόμβο και κάπου αλλού, ώστε να είναι διαθέσιμα σε περίπτωση βλάβης του κόμβου ή ακόμα η δυνατότητα πρόσβασης επεξεργαστων που βρίσκονται σε άλλους κόμβους στα

δεδομένα που είναι αποθηκευμένα σε έναν κόμβο στην περίπτωση που ο επεξεργαστής του συγκεκριμένου κόμβου παρουσιάζει βλάβη.

- Στην αυξημένη πιθανότητα να βρεθεί ένα δεδομένο στον τοπικό ενταμιευτή εφόσον υπάρχει τοπικά μόνο ένα μικρό ποσοστό της συνολικής βάσης δεδομένων.

Παράλληλα όμως η αρχιτεκτονική αυτή παρουσιάζει και μια σειρά από προβλήματα και μειονεκτήματα όπως είναι :

- Η ανάγκη για χρήση του μηχανισμού μεταβίβασης αίτησης προσπέλασης και της συνεπαγώμενης χρήσης του πρωτοκόλλου δέσμευσης δύο φάσεων επιβραδύνει σημαντικά την απόδοση του συστήματος διότι καθυστερεί το ξεκλείδωμα των δεδομένων. Αυτό εντείνει το πρόβλημα του ανταγωνισμού δεδομένων (data contention). Αυτό συμβαίνει διότι εφόσον καθυστερεί το ξεκλείδωμα των δεδομένων η πιθανότητα κάποια άλλη δοσοληψία να θέλει να προσπελάσει τα ίδια δεδομένα αυξάνει, με αποτέλεσμα πιο συχνά δοσοληψίες να περιμένουν τον τερματισμό κάποιων άλλων, προκειμένου να προσπελάσουν τα δεδομένα που έχουν αυτές κλειδωμένα. Ειδικά σε περιπτώσεις όπου ορισμένα δεδομένα είναι πολύ δημοφιλή, το φαινόμενο αυτό εντείνεται και μπορεί να μειώσει σε πολύ μεγάλο βαθμό την απόδοση του συστήματος.
- Υπάρχει επίσης ανάγκη χρήσης ενός αλγορίθμου διαμέρισης των δεδομένων στους υπάρχοντες κόμβους. Σε κάθε περίπτωση επέκτασης χρειάζεται να γίνει νέο μοίρασμα των δεδομένων ώστε και οι νέοι κόμβοι να αποκτήσουν ένα τμήμα της βάσεως. Αυτό μπορεί να είναι ιδιαίτερα δύσκολο σε περιπτώσεις που η βάση δεδομένων είναι μεγάλη σε μέγεθος οπότε και η χρονική επιβάρυνση θα είναι ανάλογα μεγάλη.
- Η εξισορόπηση της χρήσης μεταξύ των κόμβων (load balance) του συστήματος, απαραίτητη προκειμένου να έχει αυτό καλή απόδοση, μπορεί να είναι πολύ δύσκολη. Αυτό μπορεί να συμβεί σε περιπτώσεις όπου για ορισμένα δεδομένα η πιθανότητα πρόσβασης είναι πολύ μεγάλη σε σχέση με όλα τα υπόλοιπα δεδομένα. Το γεγονός αυτό μπορεί να οδηγήσει έναν κόμβο σε υπερφόρτωση και συνεπώς καθυστέρηση όλου του συστήματος καθώς οι υπόλοιποι κόμβοι που πραγματοποιούν μεταβιβάσεις αιτήσεων προσπέλασης σε αυτόν τον κόμβο, καθυστερούν εξαιτίας του. Λύση του προβλήματος μπορεί να αποτελέσει η εύρεση ενός κατάλληλου αλγορίθμου διαμέρισης των δεδομένων αλλά δεν μπορεί να αντιμετωπίσει ξαφνικές καταστάσεις (όπως π.χ. σελίδων που για μικρό διάστημα μίας ημέρας μπορεί να γίνουν πολύ δημοφιλείς, χωρίς αυτό να έχει προβλεφθεί).

Παραδείγματα τέτοιων συστημάτων είναι τα: IBM's CICS, NonStop SQL και Tandem's Encompass.

3.1.2 Αρχιτεκτονική Κοινόχρηστων Δίσκων (Shared Disk)

Αναφέρεται συχνά στη διεθνή βιβλιογραφία και ως αρχιτεκτονική κοινόχρηστων δεδομένων (data sharing architecture) και ο λόγος είναι ότι σε αυτή την αρχιτεκτονική ένας

αριθμός από κόμβους που απαρτίζουν μία ομάδα (cluster) έχουν πρόσβαση σε κοινόχρηστη βάση δεδομένων, στο επίπεδο των συσκεών αποθήκευσης (συνήθως μαγνητικούς δίσκους), όπου και είναι αποθηκευμένη η βάση δεδομένων (σχήμα 3.3).

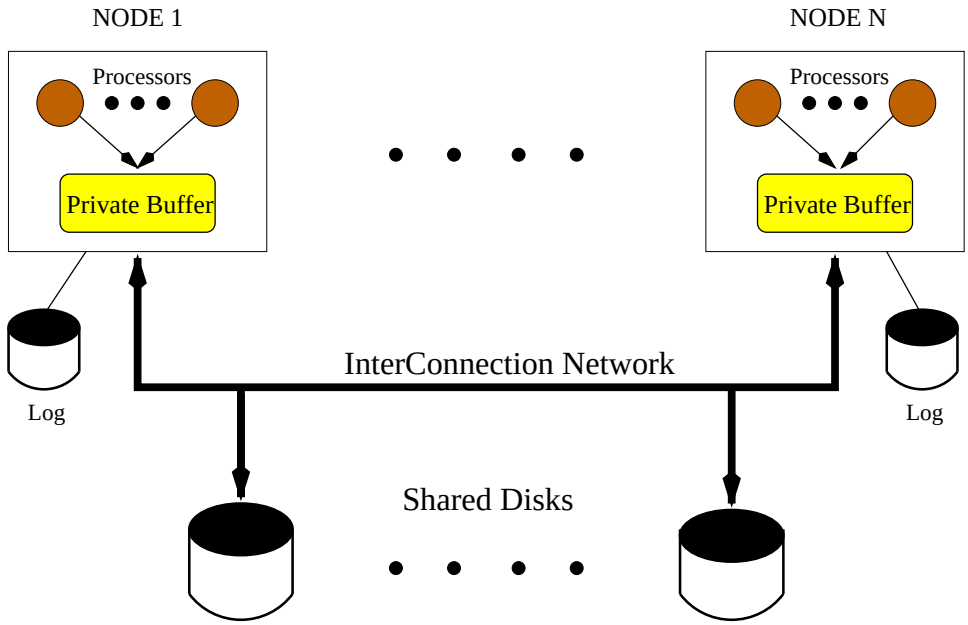
Κάθε κόμβος είναι αυτόνομος με την έννοια ότι έχει ιδιωτική μνήμη, ιδιωτικό ενταμιευτή και μπορεί να έχει και το δικό του λειτουργικό σύστημα. Επειδή σε αυτήν την αρχιτεκτονική είναι δυνατόν οποιοσδήποτε από τους κόμβους να έχει άμεση πρόσβαση σε όλη την βάση, απαιτείται η ύπαρξη ενός ελέγχου της συνδρομικότητας μεταξύ των κόμβων. Επιπλέον σε περίπτωση όπου μία σελίδα μεταβάλλεται θα πρέπει οποιοδήποτε αντίγραφό της στους ιδιωτικούς ενταμιευτές των άλλων κόμβων να ακυρωθεί (invalidation effect). Αυτό γίνεται διότι όλοι οι κόμβοι θα πρέπει στο εξής να βλέπουν την σελίδα έτσι όπως έχει προκύψει μετά την τελευταία αλλαγή.

Οι κόμβοι επικοινωνούν με την ανταλλαγή μηνυμάτων γι' αυτό και η σύζευξη αυτή αυτή συχνά αναφέρεται και ως χαλαρή σύζευξη κόμβων (loosely coupled nodes). Πλεονεκτήματα αυτής της αρχιτεκτονικής μπορούν να θεωρηθούν τα εξής :

- Η δυνατότητα κάθε κόμβος του συστήματος να "τρέχει" διαφορετικό λειτουργικό σύστημα.
- Η δυνατότητα επέκτασης με την απλή προσθήκη δίσκων ή επεξεργαστών χωρίς να χρειάζεται κάτι παραπάνω για να επαναλειτουργήσει το σύστημα. Δεν χρειάζεται δηλαδή κάποια αναδιοργάνωση των δεδομένων όπως στην SN αρχιτεκτονική.
- Η δυνατότητα αντιμετώπισης βλάβης ενός κόμβου με σχετική ευκολία, εφόσον όλοι οι κόμβοι της ομάδας που μοιράζεται την βάση δεδομένων έχουν την δυνατότητα να επιτελέσουν την εργασία του κόμβου που έπαθε βλάβη.
- Η δυνατότητα ένας οποιοσδήποτε κόμβος να μπορεί να επιτελεί την ίδια εργασία καθιστά αυτή την αρχιτεκτονική ιδανική για την εξισορροπηση του φόρτου εργασίας ανάμεσα στους κόμβους του cluster.
- Η μη απαίτηση της χρήσης του μηχανισμού μεταβίβασης αίτησης προσπέλασης αλλά ούτε και του πρωτοκόλλου δέσμευσης δύο φάσεων.

Ως βασικά της μειονεκτήματα μπορούν να θεωρηθούν τα εξής:

- Καταρχήν η ανάγκη για συντονισμό μεταξύ των κόμβων του cluster για το κλείδωμα τμημάτων της βάσης. Απαιτείται η ανταλλαγή μηνυμάτων η οποία μπορεί να καθυστερήσει σημαντικά το σύστημα.
- Η ανάγκη για ένα δίκτυο διασύνδεσης των κόμβων που να είναι αρκετά γρήγορο ώστε να μην επιβάλλει σημαντική καθυστέρηση στα μηνύματα που αναταλλάσονται. Η ύπαρξη ενός τέτοιου δικτύου είναι εξαιρετικά ασύμφορη όταν θα πρέπει να καλύπτει μεγάλες αποστάσεις. Οπότε και οι κόμβοι σε μία τέτοια αρχιτεκτονική θα πρέπει να βρίσκονται σε σχετικά μικρή γεωγραφικά απόσταση.



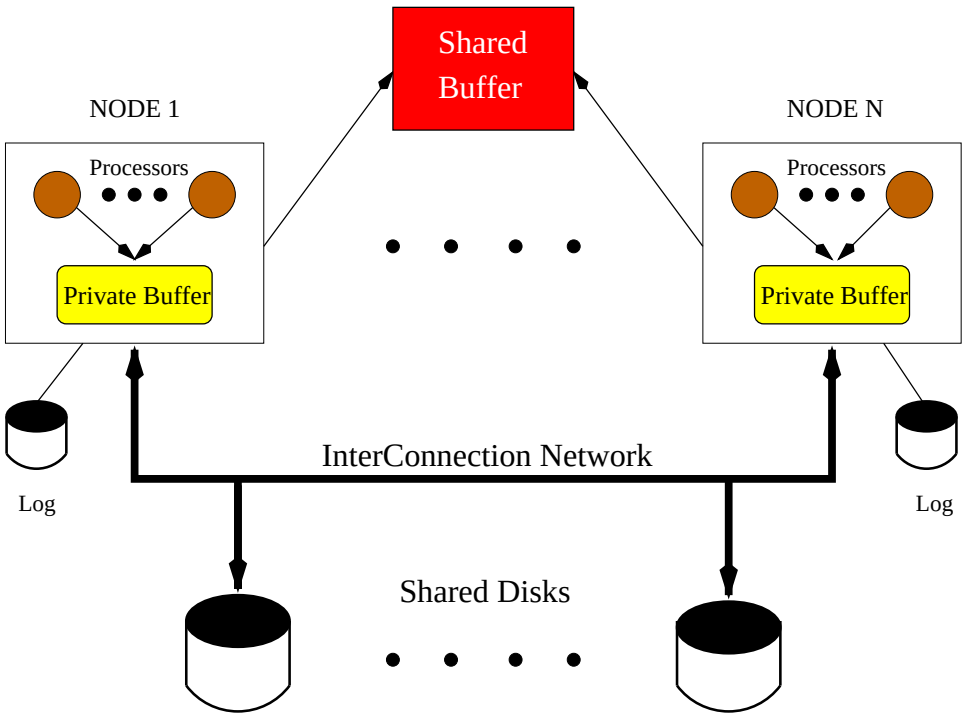
Σχήμα 3.3: Αρχιτεκτονική Shared Disk

- Η ύπαρξη του προβλήματος της ακύρωσης των σελίδων που βρίσκονται στους ιδιωτικούς ενταμιευτές, σε συνδυασμό με το ότι κάθε κόμβος χρειάζεται να αποθηκεύει σε αυτούς ολόκληρη την βάση δεδομένων (και όχι μέρος της όπως στην SN αρχιτεκτονική) μειώνει σε μεγάλο βαθμό τη πιθανότητα να βρεθεί η σελίδα στον ιδιωτικό ενταμιευτή ενός κόμβου. Ελάχιστη λοιπόν είναι η επίδραση του μεγέθους των ενταμιευτών αν δεν υπάρχει κάποια στρατηγική στον τρόπο δρομολόγησης των δοσοληψιών στο σύστημα. Αναφορά για τους τρόπους αντιμετώπισης του προβλήματος της ακύρωσης σελίδων δίνονται στην επίσκόπηση της σχετικής βιβλιογραφίας.
- Η επιλογή του μεγέθους της σελίδας που κλειδώνεται σε μία προσπέλαση.Προκειμένου να μειωθούν τα μηνύματα είναι ανάγκη να επιλεγθεί ένα μεγαλύτερο μέγεθος σελίδας που κλειδώνεται (granularity of locking). Αυτό όμως αυξάνει το πρόβλημα του ανταγωνισμό για τα δεδομένα με αποτέλεσμα να καθυστερούν οι δοσοληψίες εν αναμονη για την απόκτηση του κλειδώματος. Λόγω αυτού του σημείου συμβιβασμού (tradeoff), πολλές δεν είναι προφανής η καλύτερη επιλογή του μεγέθους της σελίδας που κλειδώνεται (granularity of locking).

Παραδείγματα τέτοιων συστημάτων είναι τα: IBM's IMS data sharing facility, TPF και DEC's DBMS for VAX Clusters, Oracle's parallel server, SparcCluster, Sequent Encore κ.τ.λ.

3.1.3 Αρχιτεκτονική Κοινόχρηστης Ενδιάμεσης Μνήμης (Shared Intermediate Memory)

Ουσιαστικά αποτελεί μία παραλλαγή της SD αρχιτεκτονικής. Η μόνη της διαφορά είναι ότι παράλληλα με τους κοινόχρηστους δίσκους που μοιράζονται οι κόμβοι, υπάρχει και ένας σχετικά μεγάλος κοινόχρηστος ενταμιευτής (buffer), δηλαδή ένα επιπλέον επίπεδο



Σχήμα 3.4: Αρχιτεκτονική Shared Intermediate Memory

στην ιεραρχία της μνήμης [CKB89] (σχήμα 3.4). Το χαρακτηριστικό του είναι ότι δεν είναι πτητικός (non-volatile memory) οπότε τα περιεχόμενά του διαφυλάσσονται από μία ενδεχόμενη βλάβη.

Σε περίπτωση που δεν βρεθεί μία σελίδα στον ιδιωτικό ενταμιευτή τότε εξετάζεται ο κοινόχρηστος ενταμιευτής. Αν βρεθεί εκεί τότε μεταφέρεται στον ιδιωτικό ενταμιευτή. Αν δεν βρεθεί ούτε στον κοινόχρηστο ενταμιευτή τότε μεταφέρεται από τον δίσκο. Κατά τον επιτυχή τερματισμό μίας δοσοληψίας οι σελίδες που άλλαξαν μεταφέρονται στον κοινόχρηστο ενταμιευτή ο οποίος ανάλογα με την πολιτική αντικατάστασης των σελίδων που υιοθετεί, αποφασίζει πότε θα μεταφερθούν στους δίσκους του συστήματος. Σημειωτέον ότι οι σελίδες του κοινόχρηστου ενταμιευτή δεν κλειδώνονται καθώς αυτή η διαδικασία γίνεται μόνο στους τοπικούς ενταμιευτές.

Τα μοναδικά του μειονέκτηματα του σε σχέση με την SD αρχιτεκτονική είναι ότι χρειάζεται να ανταλλαγούν επιπλέον μηνύματα μεταξύ των κόμβων και του κόμβου - επεξεργαστή που διατηρεί τον κοινόχρηστο ενταμιευτή και το επιπλέον κόστος της μη πτητικής μνήμης. Το δεύτερο μειονέκτημα τείνει να γίνει αμελητέο διότι το κόστος της μνήμης έχει μειωθεί σε πολύ μεγάλο βαθμό ώστε να είναι αμελητέο μπροστά στος κόστος του συστήματος.

Στον αντίποδα, τα πλεονεκτήματα της παραλλαγής αυτής είναι τα εξής :

- Η δραστική μείωση του χρόνου απόκρισης μίας δοσοληψίας, διότι ανάλογα με το μέγεθος του, ο κοινόχρηστος ενταμιευτής μειώνει δραστικά την πιθανότητα να χρειαστεί ναα μεταβεί ο κόμβος στον δίσκο για να πάρει την σελίδα. Η ανάκτηση της

σελίδας από τον ενταμιευτή είναι πολύ πιο γρήγορη σε σχέση με αυτή από τον δίσκο οπότε και μπορεί να γίνει καλύτερη χρήση των πόρων του συστήματος αλλά και να εξυπηρετηθεί πολύ πιο γρήγορα μία δοσοληψία. Αν μάλιστα υπάρχει μία τοπικότητα στις αιτήσεις προσπέλασης των δεδομένων τότε μπορεί να επιτευχθεί ένα πολύ καλό ποσοστό επιτυχίας (buffer hit ratio) στον κοινόχρηστο buffer.

- Η δυνατότητα να μειωθεί το μέγεθος των τοπικών ενταμιευτών καθώς σε ένα μεγάλο μέρος ο ρόλος τους αντικαθίσταται από τον κοινόχρηστο.
- Η μείωση σε μεγάλο βαθμό του ανταγωνισμού για το κλείδωμα σελίδων. Εφόσον κατά το commit μίας δοσοληψίας αρκεί να μεταφερθούν οι σελίδες στον κοινόχρηστο ενταμιευτή και όχι στον δίσκο, μειώνεται αισθητά η καθυστέρηση για το commit. Η βελτίωση αυτή γίνεται περισσότερο αισθητή όταν υπάρχει μεγάλη τοπικότητα στις αιτήσεις προσπέλασης των δεδομένων αλλά και όσο αυξάνεται η πιθανότητα αυτή η προσπέλαση να είναι για μεταβολή (write or update operation) της σελίδας.

Ο κοινόχρηστος ενταμιευτής μπορεί να έχει αρκετά διαφορετικές χρήσεις. Μπορεί να χρησιμοποιηθεί προκειμένου να αποθηκεύει δεδομένα της βάσης, μπορεί να χρησιμοποιηθεί για την ανταλλαγή μηνυμάτων αλλά και για την αποθήκευση του αρχείου μεταβολών (log file). Παράδειγμα υλοποίησης αυτής της αρχιτεκτονικής είναι το σύστημα SYSPLEX της IBM με την χρήση της coupling facility συσκευής.

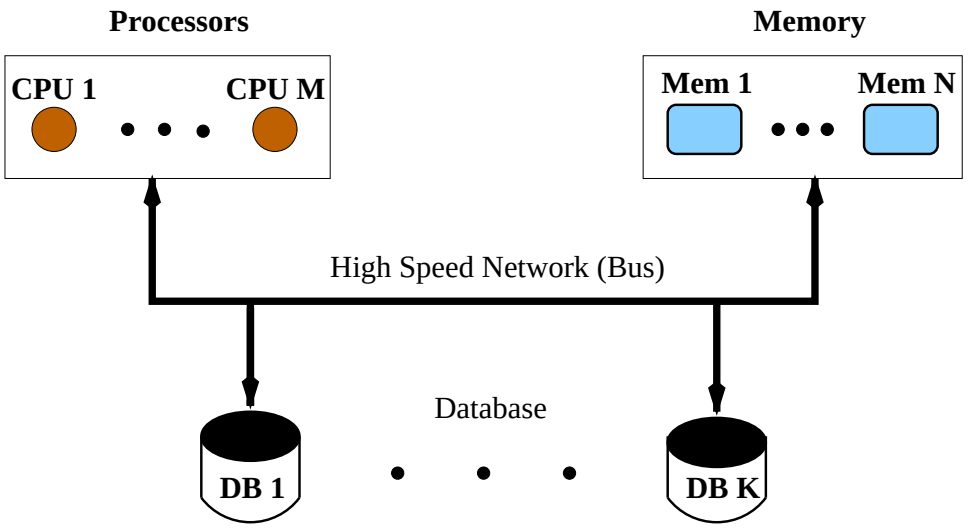
3.1.4 Αρχιτεκτονική Shared Everything

Στην αρχιτεκτονική αυτή όπως αντιλαμβάνεται κανείς από την ονομασία της οι επεξεργαστές επιτελούν οποιαδήποτε διεργασία καθώς έχουν μία κοινόχρηστη μνήμη και κοινόχρηστες όλες τις περιφεριακές μονάδες αποθήκευσης όπως είναι οι μαγνητικοί δίσκοί αλλά και τα διάφορα τερματικά (σχήμα 3.5). Υπάρχει ένα λειτουργικό σύστημα το οποίο είναι διαθέσιμο σε όλους τους επεξεργαστές μέσω της κοινής μνήμης. Ο τρόπος αυτός διασύνδεσης των υπολογιστικών κόμβων αναφέρεται και ως ισχυρή σύζευξη (tightly coupled) κόμβων.

Βασικά της πλεονεκτήματα είναι τα εξής :

- Το μικρό κόστος επικοινωνίας διότι αυτή γίνεται μέσω της κοινόχρηστης μνήμης.
- Είναι ιδανική αρχιτεκτονική για την εξισορρόπηση του φόρτου καθώς όλοι οι επεξεργαστές μπορούν να επιτελέσουν την ίδια εργασία.
- Ο συγχρονισμός μεταξύ των επεξεργαστών στις προσβάσεις τους στην μνήμη αλλά και στα κοινόχρηστα δεδομένα, μπορεί να πραγματοποιηθεί με χρήση απλών μηχανισμών.
- Η επαναφορά μετά από μία βλάβη είναι πολύ εύκολη καθώς το αρχείο μεταβολών είναι ενιαίο.

Στην αρχιτεκτονική αυτή όμως υπάρχουν σοβαρά μειονεκτήματα που την καθιστούν ως ασύμφορη για την χρήση της σε ένα κατανεμημένο σύστημα.



Σχήμα 3.5: Αρχιτεκτονική Shared Everything

- Το σύστημα εξαρτάται σε μεγάλο βαθμό από το εύρος ζώνης (bandwidth) της μνήμης αλλά και του διαύλου (bus) διασύνδεσης των κόμβων. Γι' αυτό και οι κόμβοι θα πρέπει να βρίσκονται πολύ κοντά μεταξύ τους (συνήθως στο ίδιο δωμάτιο).
- Το παραπάνω γεγονός θέτει ένα άνω όριο πάνω από το οποίο γίνεται ασύμφορη η σύζευξη μεγαλύτερου αριθμού από επεξεργαστές. Σε αυτό συνηγορεί επίσης και ότι η κοινόχρηστη μνήμη πολύ γρήγορα γίνεται σημείο συμφόρησης περιορίζοντας την δυνατότητα προσθήκης άλλων επεξεργαστών στο σύστημα. Στα σύγχρονα συστήματα αυτός ο αριθμός είναι το πολύ (32) επεξεργαστές.
- Η περίπτωση βλάβης της κύριας μνήμης ενός τέτοιου συστήματος χρειάζεται ειδική μεταχείριση διότι χάνεται ένα μεγάλο μέρος της πληροφορίας που έχει μεταβληθεί. Περιοδικά λοιπόν θα πρέπει να μεταφέρονται σε σταθερή μνήμη όλες οι αλλαγές που έχουν γίνει ώστε σε περίπτωση βλάβης το σύστημα να μην επιστρέφει σε μία κατάσταση που απαιτεί την επανάληψη μεγάλου αριθμού δοσοληψιών. Αυτό περιοδικά θα καθυστερεί σημαντικά το σύστημα κάτι που δεν είναι επιθυμητό.

Παραδείγματα τέτοιων συστημάτων είναι τα: Sequent, Encore, Sequoia, Elxi etc.

3.2 Επισκόπηση σχετικής βιβλιογραφίας

Οι εργασίες που έχουν δημοσιευτεί κυρίως προέρχονται από δύο ερευνητικά κέντρα. Αυτά είναι το IBM Thomas J. Watson Research Center, Yorktown Heights NY, και το πανεπιστήμιο του Kaiserslautern. Σε αυτά τα δύο κέντρα έχει γίνει συστηματική δουλειά στις data sharing αρχιτεκτονικές. Υπάρχουν ακόμα αρκετές ενδιαφέρουσες εργασίες που διαπραγματεύονται συγκεκριμένα θέματα στις αρχιτεκτονικές αυτές. Αρχικά θα αναφερθούμε στην δουλειά που έχει γίνει στα δύο ερευνητικά κέντρα και στη συνέχεια

θα αναφερθούμε σε μερικές από τις πιο σημαντικές εργασίες που έχουν γίνει από άλλους ερευνητές.

Στην αναφορά [CDY86] οι συγγραφείς αναλύουν την *Shared Nothing* αρχιτεκτονική και συγκεκριμένα προτείνουν ένα τρόπο διαμερισμού της βάσης δεδομένων που σε συνδυασμό με έναν αλγόριθμο δρομολόγησης συντελεί στην προσπάθεια μείωσης του ποσοστού των προσπελάσεων που χρειάζεται να μεταβιβαστούν σε άλλους κόμβους.

Στην επόμενη τους δουλειά [YCDI86] προτείνουν έναν αλγόριθμο δρομολόγησης για την περίπτωση της *Shared Disk* αρχιτεκτονικής, με σκοπό να μειώσουν την αλληλεπίδραση των κόμβων και παράλληλα να επιτύχουν μία καλή εξισορρόπηση του φόρτου εργασίας μεταξύ των κόμβων. Εισάγουν την έννοια των σχέσεων συγγένειας (affinity relations) η οποία ορίζεται ανάμεσα σε τμήματα της βάσης και δοσοληψίες. Ανάλογα με το μοντέλο πρόσβασης στη βάση δεδομένων (database call reference pattern) οι δοσοληψίες κατατάσσονται σε συγγενείς ομάδες και βάσει αυτών των ομάδων γίνεται και η δρομολόγησή τους στους κόμβους. Δοσοληψίες που ανήκουν στην ίδια ομάδα δρομολογούνται αν είναι δυνατό στον ίδιο κόμβο ώστε να αυξήσουν την πιθανότητα οι σελίδες να βρίσκονται στον ενταμιευτή αλλά και παράλληλα να μειώσουν την επίδραση της ακύρωσης των σελίδων στο σύστημα. Στο σύστημα χρησιμοποιούν δύο μορφές κλειδώματος: το κλείδωμα σελίδων μέσα σε ένα σύστημα και το κλείδωμα των σελίδων ανάμεσα στους κόμβους. Το πρωτόκολλο pass-the-buck χρησιμοποιείται για το κλείδωμα σελίδων σε επίπεδο κόμβων. Σύμφωνα με αυτό ένα μήνυμα ανταλλάσσεται ανάμεσα στους κόμβους με πληροφορίες για τα lock αλλά και για τις ακυρώσεις (invalidation) που πρέπει οι άλλοι κόμβοι να πραγματοποιήσουν στα δεδομένα της βάσης. Επίσης κάθε κλείδωμα αντικειμένων της βάσης κατακερματίζεται (γίνεται hash) σε μία κλάση. Μαζί με κάθε κλάση υπάρχει και ένα bit για κάθε σύστημα, που φανερώνει το εάν το σύστημα έχει ή περιμένει lock της συγκεκριμένης κλάσης. Αυτό το σχήμα επιτρέπει χωρίς επικοινωνία να ανατίθενται σε ένα σύστημα ολόκληρες κλάσεις από τον global lock manager.

Στο άρθρο [YDR+87] γίνεται αναφορά στην επίπτωση που έχουν οι διάφοροι παράμετροι του συστήματος στη *Shared Disk* αρχιτεκτονική. Γίνεται και πάλι χρήση ενός κεντρικοποιημένου διαχειριστή κλειδωμάτων αλλά και του πρωτοκόλλου pass-the-buck. Ειδικά μελετάται η επίπτωση του ανταγωνισμού για το κλείδωμα σελίδων στην απόδοση του συστήματος.

Στην αναφορά [DIRY89] προτείνουν ένα πρωτόκολλο που ενσωματώνει τόσο τον έλεγχο συνδρομικότητας όσο και τον έλεγχο συνέπειας (integrated concurrency coherency control) για την αρχιτεκτονική *Shared Disk* με δυνατότητα επέκτασής της ώστε να υποστηρίζει και την *Shared Intermediate Memory* αρχιτεκτονική.

Μελετώντας τις δύο αρχιτεκτονικές κατέληξαν στο να προτείνουν και ένα υβριδικό σχήμα μεταξύ της *Shared Nothing* και *Shared Disk* αρχιτεκτονικής. Αυτό περιγράφεται στην αναφορά [WDIY89] και σκοπός του είναι να κρατήσει τα θετικά από κάθε μία από τις αρχιτεκτονικές. Για τον λόγο αυτό αρχικά με μία ευριστική μέθοδο που βασίζεται στην οικογένεια αλγορίθμων simulated annealing, εκτιμάται ποιά τμήματα της βάσης θα είναι κοινόχρηστα, ποιά θα μοιραστούν και πώς στους κόμβους και τέλος πώς θα γίνει η δρομολόγηση των δοσοληψιών ώστε το σύστημα να βελτιώσει την επίδοσή του.

Στην αναφορά [YD92b] γίνεται λόγος για ένα υβριδικό τρόπο κλειδώματος σελίδων που βασίζεται στον αισιόδοξο έλεγχο ταυτόχρονης προπέλασης (Optimistic Concurrency Control ή OCC). Σύμφωνα με τον OCC μηχανισμό, μία δοσοληψία τερματίζει ανεπιτυχώς (abort) μόνο μετά το πέρας της εκτέλεσής της και πρίν γίνουν μόνιμες οι αλλαγές που επέφερε στην κατάσταση του συστήματος και όχι λόγω εμπλοκής της σε αδιέξοδο (deadlock). Σκοπός του προτεινόμενου τρόπου είναι να αυξήσει την πιθανότητα, σε μία επόμενη επανεκτέλεση της δοσοληψίας, οι σελίδες να βρίσκονται στον ενταμιευτή.

Στη συνέχεια στην αναφορά [DY93] γίνεται η σύγκριση έξι πολιτικών συνέπειας ενταμιευτών (buffer coherency policies). Αυτές οι πολιτικές μπορούν να χωριστούν σε πολιτικές εντόπισης (detection), ειδοποίησης (notification) σε περίπτωση σελίδων που δεν είναι συνεπείς και βρίσκονται στους ενταμιευτές, καθώς και σε πολιτικές που βασίζονται στην προώθηση (propagation) των αλλαγών. Στην εργασία [DDY94] γίνεται μία εκτενής ανάλυση της συμπεριφοράς των ενταμιευτών σε ένα σύστημα κοινόχρηστων δεδομένων και της επίδρασης που έχει η προσπέλαση όταν αυτή παρουσιάζει ανομοιόμορφη κατανομή της πιθανότητας προσπέλασης (skewed data access).

Τέλος στις αναφορές [YD94a], [YD94b] γίνεται σύγκριση των αρχιτεκτονικών *Shared Nothing* , *Shared Disk* , και *Shared Intermediate Memory* . Στην πρώτη από αυτές, προκειμένου να γίνει η σύγκριση γίνεται μια ομαδοποίηση των δοσοληψιών σε ομάδες που παρουσιάζουν παρόμοιο τρόπο προσπέλασης στη βάση δεδομένων. Γίνεται μελέτη τόσο περιπτώσεων όπου είναι δυνατό να χωριστούν εύκολα σε έναν αριθμό από ομάδες ίσο με τον αριθμό των κόμβων, όσο και σε περιπτώσεις όπου αυτό δεν είναι δυνατό ή ο χωρισμός αυτός καταλήγει σε ομάδες με πολύ διαφορετικό φόρτο. Ιδιαίτερη έμφαση δίνεται στο τρόπο συμπεριφοράς των τριών αρχιτεκτονικών στην μεταβολή παραμέτρων του συστήματος όπως το μέγεθος των ενταμιευτών, αριθμός των κόμβων κ.τ.λ. Αντίθετα με τη πρώτη αναφορά όπου γίνεται μελέτη των αρχιτεκτονικών για την περίπτωση όπου τα χαρακτηριστικά των δοσοληψιών είναι δεδομένα εξαρχής και σταθερά, στην δεύτερη αναφορά οι αρχιτεκτονικές μελετώνται για τις περιπτώσεις όπου αυξάνεται υπερβολικά ο φόρτος μίας κλάσης δοσοληψιών, μεταβάλλονται οι ρυθμοί άφιξης των διαφόρων κλάσεων και τέλος ένας κόμβος παθαίνει βλάβη. Σαν γενικό συμπέρασμα αυτών των συγκρίσεων είναι ότι η *Shared Intermediate Memory* αρχιτεκτονική έχει την πιο ανθεκτική (robust) συμπεριφορά.

Παράλληλα από την ίδια ομάδα του ερευνητικού κέντρου έχουν δημοσιευτεί μία σειρά από αναφορές σε θέματα δρομολόγησης και εξισορρόπησης φόρτου ανάμεσα στους κόμβους του συστήματος για την αρχιτεκτονική *Shared Nothing* [YLL91], [YBL88], [LY91], [LYL88].

Στη συνέχεια θα αναγερθούμε σε μία σειρά από άρθρα, από τον Erhard Rahm κυρίως και το πανεπιστήμιο του Kaiserslauten στη Γερμανία. Στο άρθρο του [Rah86] προτείνεται για την περίπτωση της DB sharing αρχιτεκτονικής ο αλγόριθμος *Primary Copy* ώστε να μειωθεί η ανάγκη για επικοινωνία ανάμεσα στους κόμβους αλλά και να ελαττωθεί το πρόβλημα της ακύρωσης των σελίδων που βρίσκονται αποθηκευμένες στους ενταμιευτές. Σύμφωνα με αυτόν τον αλγόριθμο, κάθε κόμβος αναλαμβάνει ένα τμήμα της κοινόχρηστης βάσης δεδομένων. Μόνο γι' αυτό το τμήμα της βάσης κρατάει πληροφορίες στον τοπικό του lock table. Σε περίπτωση όπου μία δοσοληψία που είναι ενεργή σε έναν κόμβο χρειάζεται δεδομένα που έχουν ανατεθεί σε άλλον κόμβο, τότε η αίτηση αυτή μεταβιβάζεται στον άλλο

κόμβο. Η τεχνική αυτή έχει το πλεονέκτημα (ενισχύεται όταν υπάρχει και η κατάλληλη δρομολόγηση των δοσοληψιών) ότι μπορεί χωρίς επικοινωνία ένας κόμβος να εξυπηρετήσει αιτήσεις προσπέλασης για το δικό του τμήμα της βάσης δεδομένων. Ουσιαστικά δηλαδή αποτελεί μία προσπάθεια μίμησης της *Shared Nothing* αρχιτεκτονικής σε ένα σύστημα όπου όλη η βάση είναι διαθέσιμη σε όλους τους κόμβους.

Στο άρθρο [BHR90] γίνεται λόγος για την προσθήκη μίας γρήγορης, μη-πτητικής μνήμης (Global Extended Memory ή GEM) η οποία μειώνει τον ανταγωνισμό για το κλείδωμα δεδομένων, μειώνει την καθυστέρηση που επιβάλλει η επικοινωνία των κόμβων αλλά και εξισορροπεί το φόρτο εργασίας ανάμεσα στους κόμβους του συστήματος. Βασίζεται και πάλι στην *Shared Disk* αρχιτεκτονική αλλά στο επιπλέον επίπεδο μνήμης που προσθέτουν, σε αντίθεση με την *Shared Intermediate Memory* αρχιτεκτονική, αποθηκεύουν εκτός από τις σελίδες που έχουν μεταβληθεί, το αρχείο μεταβολών, το lock table, αλλά χρησιμεύει και για την ανταλλαγή μηνυμάτων (με την προϋπόθεση ότι οι κόμβοι είναι κοντά ώστε να είναι συμφέρουσα η ανταλλαγή των μηνυμάτων μέσω της GEM). Οι αναφορές [Rah91b], [Rah92b] διαπραγματεύονται τα ίδια θέματα με πολύ μικρές διαφορές ως προς την χρήση της GEM.

Στην αναφορά [Rah91a] προτείνονται κάποιοι νέοι τρόποι για την διατήρηση ενός αρχείου μεταβολών (logging) και για την επαναφορά (recovery) του συστήματος σε συνεπή κατάσταση μετά από μία βλάβη, που όμως προϋποθέτουν την χρήση του αλγόριθμου που βρίσκεται στην αναφορά [Rah86] για έλεγχο συνδρομικότητας και συνέπειας.

Στο [Rah92a] γίνεται μία εκτενής αναφορά στον τρόπο κατανομής του φόρτου εργασίας στα κατανεμημένα συστήματα επεξεργασίας δοσοληψιών. Παρουσιάζει συνοπτικά τις μέχρι τότε υπάρχουσες τεχνικές.

Στα [MR92], [Rah93a] γίνεται μία αξιολόγηση των αρχιτεκτονικών ενώ στις [Rah93c] και [RS95] αναφορές, γίνεται μία ανάλυση της *Shared Disk* αρχιτεκτονικής για την παράλληλη επεξεργασία επερωτήσεων (parallel query processing) αλλά και για την παράλληλη σαρωτική επεξεργασία (parallel scan processing).

Οι [RM93], [RM95] και [Rah96] δημοσιεύσεις διαπραγματεύονται την δυνατότητα για εξισορρόπηση του φόρτου εργασίας ανάμεσα στους κόμβους, τόσο για την *Shared Nothing* αρχιτεκτονική όσο και για την *Shared Disk* . Καταλήγουν στην διαπίστωση ότι γενικά η *Shared Disk* προσέγγιση έχει σαφώς καλύτερες προοπτικές στο θέμα αυτό.

Τέλος στην [Rah93b] μπορεί κανείς να βρει μία ανακεφαλαίωση και κριτική της απόδοσης των πρωτοκόλλων συνδρομικότητας και συνέπειας, τα οποία έχουν προταθεί για τα συστήματα που έχουν μία κοινή βάση δεδομένων.

3.2.1 Άλλες δημοσιεύσεις

Υπάρχει μία πληθώρα από δημοσιεύσεις για θέματα πάνω στις αρχιτεκτονικές που μελετάμε στην παρούσα εργασία. Αν και όχι τόσο συστηματικές και σφαιρικές οι μελέτες από τους περισσότερους συγγραφείς, σε σχέση πάντα με τις προηγούμενες, παρόλα αυτά είναι και αυτές ιδιαίτερα αξιόλογες. Απλά διαπραγματεύονται πολύ πιο περιορισμένα θέματα.

Πιο συγκεκριμένα :

Στην αναφορά [Bhi88] ο συγγραφέας μελετά τις τρεις αρχιτεκτονικές *Shared Nothing* , *Shared Disk* , και *Shared Everything* , συνεχίζοντας μια εργασία που ξεκίνησε με τα άρθρα του [BS87], [BS88]. Επίσης για την αρχιτεκτονική *Shared Disk* μελετά ορισμένες διαφορετικές τεχνικές προκειμένου να ελαχιστοποιήσει τον αριθμό των μηνυμάτων για το κλείδωμα των σελίδων της βάσης.

Ζητήματα πρωτοκόλλων επαναφοράς αλλά και τρόπους απομόνωσης των δεδομένων στην *Shared Everything* αρχιτεκτονική διαπραγματεύονται οι εργασίες [MR94], [MR95].

Στο άρθρο [DG92] γίνεται μία συνολική παρουσίαση της επίδοσης των συστημάτων για την επεξεργασία υψηλού όγκου πληροφοριών με μία αναφορά στα κυριότερα προϊόντα που κυκλοφορούν στην διεθνή αγορά. Παράλληλα γίνεται μία πιο εκτενής αναφορά σε θέματα που άπτονται της *Shared Nothing* αρχιτεκτονικής, όπως είναι ο διαμοιρασμός των δεδομένων στους κόμβους του συστήματος.

Στην εργασία [PMC+90] μελετάται η επίδραση του παραλληλισμού επερωτήσεων στις διάφορες αρχιτεκτονικές. Παράλληλα με βασικό συγγραφέα τον C. Mohan, υπάρχει ένας αρκετά μεγάλος αριθμός από δημοσιεύσεις τόσο για θέματα recovery [MN92] όσο και για θέματα ελέγχου συνδρομικότητας [MN91].

3.3 Συμπεράσματα επισκόπησης βιβλιογραφίας

Επιγραμματικά θα λέγαμε ότι στην διεθνή βιβλιογραφία υπάρχει ικανοποιητικός αριθμός από δημοσιεύσεις οι οποίες αναφέρονται σε συστήματα επεξεργασίας δοσοληψιών καθώς η χρήση αυτών των συστημάτων έχει εξαπλωθεί τις τελευταίες δύο δεκαετίες. Θα λέγαμε ότι οι η λύση για το ποιά αρχιτεκτονική είναι η καλύτερη δεν απαντάται μονολεκτικά. Κάθε αρχιτεκτονική έχει ορισμένα μειονεκτήματα τα οποία σε ορισμένες εφαρμογές είναι δυνατό να οδηγήσουν το σύστημα σε μη αποδεκτούς χρόνους απόκρισης για τόν φόρτο εργασίας του συστήματος. Θυμίζουμε ότι η εγγύηση του χρόνου απόκρισης είναι ένα βασικό χαρακτηριστικό των συστημάτων επεξεργασίας δοσοληψιών.

Έτσι η *Shared Everything* αρχιτεκτονική έχει ως βασικό μειονέκτημα ότι εξαρτάται σε μεγάλο βαθμό από τον δίαυλο επικοινωνίας και την ταχύτητα της μνήμης και έτσι ο αριθμός των επεξεργαστών που μπορούν να αξιοποιηθούν περιορίζεται σε μερικές δεκάδες και μάλιστα, προκειμένου να υπάρχει γρήγορη επικοινωνία αυτοί θα πρέπει να βρίσκονται γεωγραφικά πολύ κοντά.

Αντίθετα η επεκτασιμότητα είναι το μεγαλύτερο πλεονέκτημα της *Shared Nothing* αρχιτεκτονικής αφού επιτρέπει επέκταση του συστήματος σε απεριόριστο αριθμό κόμβων αλλά δύναται να είναι και γεωγραφικά κατανεμημένοι σε μεγάλες αποστάσεις. Το σοβαρότερο μειονέκτημά της είναι η επιλογή του τρόπου διαμοιρασμού της βάσης δεδομένων στους κόμβους του συστήματος. Διότι ο διαμοιρασμός μπορεί να είναι τέτοιος ώστε το σύστημα να οδηγήθεί σε κατάσταση όπου δεν είναι δυνατή η εξισορρόπηση του φόρτου εργασίας ανάμεσα στους κόμβους του συστήματος ή ακόμα χειρότερα η απόδοση του

συστήματος να χειροτερεύσει λόγω αδυναμίας εξυπηρέτησης του φόρτου εργασίας σε έναν κόμβο. Αυτό μπορεί να συμβεί όταν αυτός ο κόμβος έχει στον τοπικό του δίσκο σελίδες της βάσης δεδομένων οι οποίες προσπελάσονται με μεγάλη πιθανότητα.

Οι αρχιτεκτονικές *Shared Disk* και *Shared Intermediate Memory* έχουν θα μπορούσαμε να πούμε, μία μέση συμπεριφορά σε σχέση με τις προηγούμενες δύο αρχιτεκτονικές. Η επεκτασιμότητά τους δεν είναι δυνατό να γίνει σε έναν πολύ μεγάλο αριθμό από υπολογιστικούς κόμβους, αλλά παρόλα αυτά μπορεί εύκολα να περιλαμβάνει αρκετές εκατοντάδες υπολογιστικούς κόμβους. Η προσθήκη και η αφαίρεση ενός κόμβου δεν απαιτεί κάποιου είδους εκ νέου διαμοιρασμό της βάσης (όπως στη *Shared Nothing* αρχιτεκτονική) αλλά είναι πολύ απλή. Επειδή όλη η βάση είναι άμεσα προσβάσιμη από όλους τους κόμβους, η εξισορρόπηση του φόρτου είναι εύκολο να επιτευχθεί. Μειονέκτημά τους είναι ότι εφόσον τα δεδομένα είναι άμεσα προσβάσιμα από όλους τους κόμβους απαιτείται συγχρονισμός στον τρόπο που γίνονται οι προσπελάσεις ώστε να μην παραβιαστούν οι ACID ιδιότητες της δοσοληψίας. Αυτό απαιτεί την ανταλλαγή μεγάλου αριθμού από μηνύματα. Το δίκτυο διασύνδεσης και οι δίσκοι του συστήματος περιορίζουν την απόδοση του συστήματος σε σημαντικό βαθμό. Στην βιβλιογραφία που προαναφέραμε υπάρχουν αρκετές αναφορές με σύγκριση αυτών των αρχιτεκτονικών όπου για μικρούς αριθμούς κόμβων η *Shared Intermediate Memory* αρχιτεκτονική φαίνεται να υπερέχει τόσο της *Shared Disk* όσο και της *Shared Nothing* . Η *Shared Everything* συνίσταται μόνο για εφαρμογές που καλύπτουν γεωγραφικά πολύ περιορισμένη έκταση και σίγουρα δεν μπορούν να αποτελέσουν επιλογή π.χ. για ένα τραπεζικό σύστημα.

Σε γενικές εφαρμογές που περιλαμβάνουν ορισμένες εκατοντάδες υπολογιστικούς κόμβους η καλύτερη επιλογή είναι μάλλον αυτή των υβριδικών συστημάτων όπου σε τοπικό επίπεδο είναι υλοποιημένη μία αρχιτεκτονική *Shared Intermediate Memory* (ή σε ακόμα σε πιο τοπικό μία *Shared Everything* αρχιτεκτονική) ενώ στο ανώτερο επίπεδο είναι υλοποιημένη μία *Shared Nothing* αρχιτεκτονική.

Τέλος από μία προσεκτική ματιά στην βιβλιογραφία θα δούμε ότι έχουν προταθεί αρκετοί αλγόριθμοι για δρομολόγηση δοσοληψιών και εξισορρόπηση φόρτου εργασίας για την *Shared Nothing* αρχιτεκτονική ενώ για τις *Shared Intermediate Memory* και *Shared Disk* αρχιτεκτονικές δεν υπάρχει ανάλογος αριθμός δημοσιεύσεων, ενώ στον τομέα της δρομολόγησης δοσοληψιών δεν υπάρχει κάποια γνωστή δημοσίευση. Η συνεισφορά αυτής της εργασίας είναι η ανάπτυξη και αξιολόγηση της απόδοσης ενός νέου αλγορίθμου δρομολόγησης δοσοληψιών για τις δύο αυτές αρχιτεκτονικές.

Κεφάλαιο 4

Επέκταση προσομοιωτή TPsim

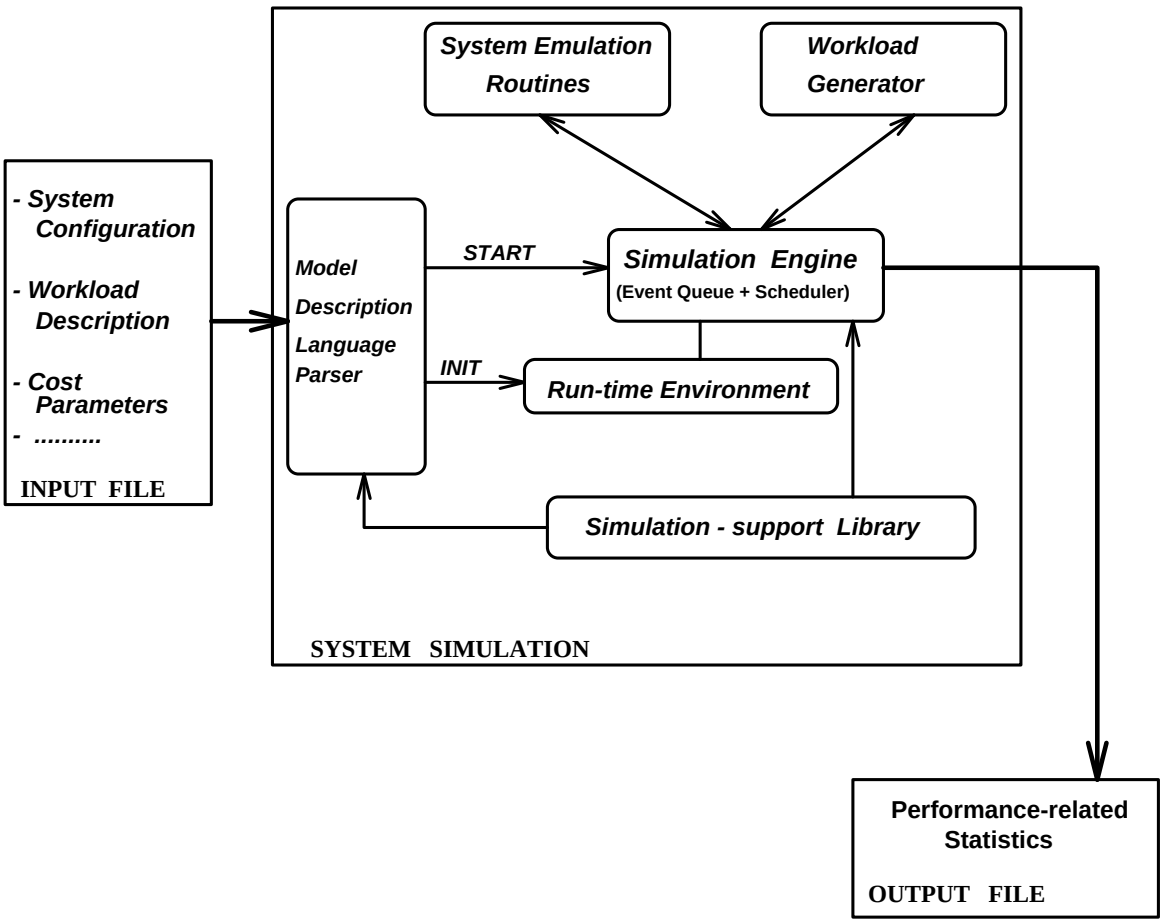
Στα πλαίσια της εργασίας και για την μελέτη των αρχιτεκτονικών σύζευξης υπολογιστικών κόμβων, έγινε επέκταση ενός ήδη υπάρχοντος προσομοιωτή για κατανεμημένα συστήματα επεξεργασίας δοσοληψιών, του **TPsim**. Αρχικά θα δοθούν ορισμένα βασικά χαρακτηριστικά του προσομοιωτή και στην συνέχεια μία λεπτομερής περιγραφή των αλγορίθμων που χρησιμοποιήθηκαν για την υλοποίηση των αρχιτεκτονικών αλλά και ενός νέου αλγορίθμου για δρομολόγηση δοσοληψιών σε αρχιτεκτονικές με κοινόχρηστα δεδομένα.

4.1 Ο προσομοιωτής TPsim

Ο TPsim είναι ένας προσομοιωτής για κατανεμημένα συστήματα διαχείρισης βάσεων δεδομένων (Distributed Database Management Systems ή πιο σύντομα DBMS) και για On-line συστήματα επεξεργασίας δοσοληψιών (Online Transaction Processing Systems ή OLTP). Βασίζεται σε μία βιβλιοθήκη υποστήριξης η οποία παρέχει έναν αριθμό από έτοιμες συναρτήσεις που είναι απαραίτητες για να γίνει μία προσομοίωση και να συλλεγθούν κάποιες μετρήσεις. Έτσι με τη χρήση της βιβλιοθήκης αυτής, γίνεται η κατασκευή του μοντέλου και παράλληλα δεν χρειάζεται να μπει ο προγραμματιστής στην διαδικασία κατασκευής low-level συναρτήσεων. Υπάρχει ένα αρχείο διασύνδεσης του προσομοιωτή με την βιβλιοθήκη υποστήριξης ώστε να είναι δυνατή η χρήση διαφορετικών βιβλιοθηκών χωρίς να πρέπει να αλλαχυεί ο κώδικας του TPsim. Αυτή τη στιγμή υποστηρίζει δύο διαφορετικές βιβλιοθήκες, την PARASOL [Nei91] και την CSIM [Sch94].

Ο προσομοιωτής έχει γραφτεί με χρήση της γλώσσας προγραμματισμού **C** και δίνει την δυνατότητα στους χρήστες να το χρησιμοποιήσουν σε περιβάλλον DEC workstations με το λειτουργικό Digital UNIX, σε Sparc workstations με το λειτουργικό SunOs αλλά και σε συστήματα που υποστηρίζουν το λειτουργικό Solaris.

Ο προσομοιωτής βασίζεται σε νήματα ελέγχου που υλοποιούν τις πολιτικές που έχει επιλέξει ο προγραμματιστής για την διαχείριση των πόρων του συστήματος όπως π.χ. πολιτική χρονοπρογραμματισμού των επεξεργαστών, πολιτική αντικατάστασης των σελίδων στους



Σχήμα 4.1: Βασικά δομικά στοιχεία του προσομοιωτή κατηγορίες

ενταμιευτές κ.α. Αποτελείται σε ένα μεγάλο βαθμό από ανεξάρτητα τμήματα (modules) αφού επιτρέπει εύκολα την αντικατάσταση πολιτικών και αλγορίθμων που χρησιμοποιούνται για την προσομοίωση, με άλλους.

Σε μεγάλο βαθμό, η ευελιξία του οφείλεται στο γεγονός ότι ο χρήστης του προσομοιωτή, προκειμένου να μοντελοποιήσει το σύστημα το οποίο θέλει να προσομοιώσει αρκεί, να δώσει την περιγραφή του συστήματος σε μία γλώσσα προδιαγραφής. Έτσι με αυτόν τον τρόπο δεν είναι ανάγκη να μεταφραστεί και πάλι το πρόγραμμα προκειμένου να γίνουν αλλαγές στο σύστημα. Αυτή η γλώσσα προδιαγραφής δίνει την δυνατότητα στον χρήστη να επιλέξει την τοπολογία του συστήματος, τον τύπο των υπολογιστικών κόμβων μαζί με τα χαρακτηριστικά τους (π.χ. ταχύτητα επεξεργαστή, μέγεθος ιδιωτικού ενταμιευτή κ.τ.λ.), το σχήμα της βάσης δεδομένων καθώς και τον τρόπο που αυτή είναι μοιρασμένη στο σύστημα, τα αποθηκευτικά μέσα (δίσκους) μαζί με τα χαρακτηριστικά τους, καθώς και τον φόρτο εργασίας που επιθυμεί. Στο τμήμα όπου ορίζεται ο φόρτος εργασίας του συστήματος προς προσομοίωση, ο χρήστης μπορεί να ορίσει κλάσεις δοσοληψιών αλλά και στόχους επίδοσης για κάθε κλάση. Αυτοί χρησιμοποιούνται από δρομολογητές (όταν η πολιτική τους υποστηρίζει) προκειμένου να βελτιώσουν τον ρυθμό με τον οποίο οι στόχοι ικανοποιούνται.

Κατά την κλήση του προγράμματος ο χρήστης επιλέγει τον χρόνο προσομοίωσης, καθώς και τις πολιτικές χρονοπρογραμματισμού επεξεργαστών και δρομολόγησης από μία σειρά

πολιτικών που έχουν ενσωματωθεί στον προσομοιωτή.

Στο σχήμα 4.1 φαίνονται τα δομικά στοιχεία του προσομοιωτή τα οποία χρησιμοποιούνται κατά την διαδικασία αξιολόγησης ενός προτεινόμενου συστήματος που περιγράφεται στο αρχείο εισόδου.

Ο προσομοιωτής μπορεί να χρησιμοποιηθεί για την μελέτη DBMS και OLTP συστημάτων και αρχιτεκτονικών κάτω από πολύ διαφορετικές καταστάσεις φόρτου, ώστε να μελετηθεί η συμπεριφορά του συστήματος πριν την αγορά του εξοπλισμού. Ακόμα μπορεί να χρησιμοποιηθεί εύκολα από έναν προγραμματιστή προκειμένου να αξιολογήσει έναν μεγάλο αριθμό από αλγόριθμους : π.χ. αλγόριθμους δρομολόγησης, αλγόριθμους αντικατάστασης σελίδων των ενταμιευτών, πολιτικές χρονοπρογραμματισμού, μηχανισμούς για την διαμοίραση των δεδομένων, αλγόριθμους ομαδοποίησης δοσοληψιών έμμεσα με την αξιολόγηση της ποιότητάς της ομαδοποίησης τους από τον προσομοιωτή και με την χρήση αλγόριθμων δρομολόγησης, μηχανισμούς ελέγχου της συνδρομικότητας, μηχανισμούς επαναφοράς, καθώς και οποιουδήποτε άλλου αλγορίθμου που μπορεί να ενσωματωθεί σε ένα σύστημα επεξεργασίας δοσοληψιών.

Περισσότερες λεπτομέρειες μπορεί κανείς να βρει στο εγχειρίδιο χρήσης του προσομοιωτή [AAMN97], το οποίο υπάρχει και σε online μορφή. Εκεί δίνονται αναλυτικά όλα οι απαραίτητες πληροφορίες για έναν χρήστη αλλά και για έναν που θέλει να τροποποιήσει τον προσομοιωτή ώστε να υποστηρίζει μία δική του υλοποίηση πολιτικής. Στη συνέχεια, αναφορά θα γίνεται μόνο στα σημεία όπου έχουν γίνει οι αλλαγές ή έχει μεταβληθεί κάτι ή όταν απαιτείται να γίνει αναφορά προκειμένου να γίνουν ορισμένα πράγματα κατανοητά.

Ο προσομοιωτής στο ξεκίνημα της παρούσας εργασίας υποστήριζε μία και μόνο αρχιτεκτονική, την *Shared Nothing* . Η υλοποίηση της είχε γίνει από τον Μαραζάκη στα πλαίσια της μεταπτυχιακής του εργασίας [Μαρ95b]. Έτσι στην παρούσα εργασία έγινε η υλοποίηση της *Shared Disk* και *Shared Intermediate Memory* αρχιτεκτονικής και παράλληλα έγινε και η υλοποίηση ενός αλγορίθμου δρομολόγησης για αυτές τις αρχιτεκτονικές. Στη συνέχεια θα αναλυθούν τα διάφορα τμήματα της υλοποίησης ξεκινώντας από την γλώσσα προδιαγραφής, περνώντας στον τρόπο μοντελοποίησης των αρχιτεκτονικών, τους μηχανισμούς που αναπτύχθηκαν και καταλήγοντας στον αλγόριθμο δρομολόγησης.

4.2 Γλώσσα προδιαγραφής

Η πρώτη φάση της παρούσας εργασίας περιλάμβανε την μετατροπή της γλώσσας προδιαγραφής που χρησιμοποιεί ο TPsim προκειμένου να περιγραφεί το μοντέλο του συστήματος που επιθυμεί ο χρήστης ώστε να υποστηρίζει ορισμένες νέες συντακτικές δομές κατάλληλες για την περιγραφή *Shared Disk* και *Shared Intermediate Memory* συστημάτων. Σκοπός της όλης εργασίας ήταν να συνεχίσει ο προσομοιωτής να είναι ενιαίος και να μην αποτελείται από διαφορετικά εκτελέσιμα αρχεία για κάθε αρχιτεκτονική. Οπότε σε πολλά σημεία χρειάστηκε να προσαρμοστεί η τωρινή με την ήδη υπάρχουσα υλοποίηση όταν αυτό ήταν σχετικά εύκολο να γίνει. Ειδικά για την γλώσσα προδιαγραφής υπήρχε ένας επιπλέον λόγος: η ανάγκη για ομοιομορφία στον τρόπο περιγραφής του συστήματος κάτω από όλες

τις αρχιτεκτονικές.

Ο σαρωτής (parser) κατά την διάρκεια επεξεργασίας του αρχείου εισόδου, δημιουργεί το σύστημα, κάνοντας χρήση ρουτίνων της βιβλιοθήκης υποστήριξης και με βάση τις πληροφορίες για τα διάφορα τμήματα (π.χ. κόμβοι, δίκτυο διασύνδεσης κ.τ.λ.) του συστήματος που διαβάζει μέσα από το αρχείο εισόδου. Στη συνέχεια ενεργοποιεί την γεννήτρια συμβάντων που παρέχει η βιβλιοθήκη υποστήριξης και μαζί τις ρουτίνες που έχουν γραφτεί και που κάνουν διαχείριση των πόρων του συστήματος. Επιπλέον ο parser παράγει και μία δομή η οποία χρησιμοποιείται ως κατάλογος του συστήματος και περιλαμβάνει πληροφορίες για τους κόμβους, τις θύρες επικοινωνίας των διεργασιών καθώς και άλλες πληροφορίες για το σύστημα. Αυτή η δομή ουσιαστικά αποτελεί τον πίνακα συμβόλων (symbol table) που δημιουργεί ο parser κατά την επεξεργασία του αρχείου εισόδου και υποθέτουμε ότι υπάρχει ένα αντίγραφό του σε κάθε κόμβο.

Η γλώσσα προδιαγραφής υποστηρίζει τον ορισμό τύπων αποθηκευτικών μέσων, δικτύου επικοινωνίας, υπολογιστικών κόμβων, ομάδων από υπολογιστικούς κόμβους (cluster). Διάκριση γίνεται μεταξύ του ορισμού τύπων ή κλάσεων ενός αντικειμένου και στιγμιοτύπου (instance) αυτού του τύπου. Έτσι, για παράδειγμα, για να ορίσει κανείς έναν κόμβο θα πρέπει αρχικά να ορίσει τον τύπο του. Ο ορισμός του τύπου ενός κόμβου αποτελείται από έναν αριθμό από παραμέτρους που καθορίζουν το κόστος για μία σειρά πράξεις του κόμβου καθώς και τον ορισμό της ταχύτητάς του, των ιδιωτικών αποθηκευτικών μέσων και ενταμιευτών και του βαθμού πολυπρογραμματισμού. Στη συνέχεια ορίζει ο χρήστης έναν κόμβο του τύπου που έχει ορίσει παραπάνω. Αυτό δίνει την δυνατότητα να μην είναι αναγκαίος ο ορισμός κάθε κόμβου ξεχωριστά όταν πρόκειται για ίδιους κόμβους. Παρακάτω θα δούμε τον τρόπο ορισμού για κάθε μια από τις συντακτικές δομές που επιτρέπει η γλώσσα προδιαγραφής. Μαζί δίνονται και κάποια παραδείγματα ενώ τα token της γλώσσας προδιαγραφής φαίνονται με έντονα κεφαλαία γράμματα.

4.2.1 Αποθηκευτικά μέσα

Τα αποθηκευτικά μέσα χρησιμοποιούνται για την αποθήκευση τόσο των δεδομένων αλλά και του αρχείου μεταβολών. Η συντακτική δομή που παρέχεται είναι η DEFINE IO_DEVICE. Δίνεται η δυνατότητα για δύο τρόπους ορισμού ενός αποθηκευτικού μέσου. Ο βασικός τρόπος είναι ο μαγνητικός δίσκος που φαίνεται παρακάτω:

- **DEFINE IO_DEVICE** DataDisk **WITH** {
  IO_COPY_DELAY : 0.001; %time to transfer a block to/from memory
  IO_LOAD_DELAY : 0.00075; %time to ”load” disk arm
  IO_SEEK_DELAY : 0.00075; %time to move disk head from track to track
  IO_ROTATIONAL_DELAY : 0.008333; %time for full disk rotation
  NUM_HEADS : 8; %number of disk heads
  NUM_SECTORS : 250000; %number of blocks (total: around 2 GBytes)
  }

Η καθυστέρηση για τη προσπέλαση στο δίσκο καθορίζεται από τις εξής παραμέτρους: την καθυστέρηση για αναζήτηση (seek delay), την καθυστέρηση για την μεταφορά δεδομένων, για την μεταφορά από και προς την κύρια μνήμη (I/O copy delay), την καθυστέρηση λόγω περιστροφής (rotational delay) και την καθυστέρηση για την κίνηση της κεφαλής (load delay). Η καθυστέρηση για αναφορά ορίζεται ως ο χρόνος για την μεταφορά της κεφαλής πάνω από τον κύλινδρο του δίσκου που περιλαμβάνει το block. Η καθυστέρηση αυτή αποτελεί το ήμισυ της καθυστέρησης για μία πλήρη περιστροφή.

Επίσης παρέχεται η δυνατότητα ορισμού και ενός πιο απλού τύπου όπου λαμβάνεται υπόψη μόνο η ελάχιστη και μέγιστη καθυστέρηση για την προσπέλαση στον δίσκο. Η καθυστέρηση για την προσπέλαση στον δίσκο είναι μια τυχαία μεταβλητή με ομοιόμορφη κατανομή και τιμές ανάμεσα στην μέγιστη και ελάχιστη τιμή. Όλοι οι χρόνοι είναι σε δευτερόλεπτα.

- **DEFINE IO_DEVICE** LogDisk **WITH** {
  IO_DELAY_MIN : 0.015; %minimum I/O delay : 15 msec
  IO_DELAY_MAX : 0.005; %maximum I/O delay : 5 msec

  }

4.2.2 Υπολογιστικοί κόμβοι

Κάθε κόμβος θεωρείται ως στιγμιότυπο μίας κλάσης κόμβων η οποία θα πρέπει να έχει οριστεί προηγουμένως. Γι' αυτό πρώτα θα δούμε τον τρόπο ορισμού μίας κλάσης κόμβων. Ο προσομοιωτής υποστηρίζει κόμβους με έναν μόνο επεξεργαστή αν και συντακτικά δίνεται η δυνατότητα ορισμού οποιουδήποτε αριθμού από επεξεργαστές σε κάθε κόμβο. Οι συντακτικές δομές που χρησιμοποιούνται είναι η DEFINE NODE_CLASS για τους κόμβους που έχουν τοπικούς δίσκους με μέρος της βάσης δεδομένων (*Shared Nothing* αρχιτεκτονική) και η SIMPLE_NODE_CLASS για τους κόμβους που δεν έχουν δεδομένα αποθηκευμένα τοπικά. Με την πρώτη δομή δεν θα ασχοληθούμε καθόλου αφού αποτελεί μέρος της *Shared Nothing* αρχιτεκτονικής. Παρόλα αυτά στο παράρτημα όπου υπάρχει η γραμματική μπορεί κανείς να δει τις διαφορές στον τρόπο ορισμού της σε σχέση με την SIMPLE_NODE_CLASS συντακτική δομή.

Αρχικά ορίζεται ο αριθμός των επεξεργαστών ανά υπολογιστικό κόμβο (πάντα ένα στην παρούσα υλοποίηση του προσομοιωτή). Στη συνέχεια ορίζεται ο βαθμός πολυπρογραμματισμού (MPL) που καθορίζει τον μέγιστο αριθμό από δοσοληψίες που μπορούν να είναι ταυτόχρονα ενεργές σε έναν κόμβο της κλάσης αυτής. Ο βαθμός πολυπρογραμματισμού επηρεάζει τον βαθμό χρησιμοποίησης του επεξεργαστή διότι επιτρέπει στον επεξεργαστή να εξυπηρετήσει μία άλλη δοσοληψία (αν τού το επιτρέπει ο βαθμός πολυπρογραμματισμού) όση ώρα αναμένει την εκτέλεση μίας πράξης εισόδου/εξόδου από κάποιο αποθηκευτικό μέσο. Παράλληλα όμως αυξάνεται και ο ανταγωνισμός για την ταυτόχρονη προσπέλαση στα δεδομένα. Ορίζεται η μέση ταχύτητα του επεξεργαστή (cpu rate) ως ο αριθμός των εντολών που είναι ικανός να εκτελέσει ο επεξεργαστής σε ένα χρονικό διάστημα (μετράται σε εκατομμύρια εντολές το δευτερόλεπτο). Επίσης ορίζεται ο τύπος του

δίσκου που θα χρησιμοποιηθεί για την αποθήκευση του αρχείου μεταβολών (log I/O device) καθώς και το μέγεθος (σε σελίδες) του τοπικού ενταμιευτή (buffer) που έχει στην διάθεση του κάθε κόμβος της κλάσης αυτής. Ο ιδιωτικός ενταμιευτής χρησιμοποιείται για την αποθήκευση σελίδων της βάσης που χρησιμοποίησε ο κόμβος, προκειμένου να αποφευχθεί μια νέα προσπέλαση στον δίσκο η οποία είναι πολύ χρονοβόρα. Στην συνέχεια υπάρχει μία σειρά από παραμέτρους (σε αριθμό εντολών) οι οποίες ρυθμίζουν το κόστος για πράξεις που έχουν να κάνουν με την επεξεργασία μίας δοσοληψίας. Το κόστος αυτό των πράξεων είναι μεταβλητό διότι καθορίζεται ουσιαστικά από την ταχύτητα του επεξεργαστή. Υπάρχει κόστος εκτέλεσης για την σύνδεση και αποσύνδεση μίας δοσοληψίας με ένα νήμα ελέγχου, κόστος εκτέλεσης για την προσπέλαση στα δεδομένα των δίσκων αλλά και του πρωτοκόλλου δέσμευσης καθώς και οι επιβαρύνσεις για την προσπέλαση στα αποθηκευτικά μέσα.

Στη συνέχεια παρατίθεται ένα παράδειγμα ορισμού κλάσης κόμβων χωρίς τοπικά δεδομένα.

- **DEFINE SIMPLE_NODE_CLASS** SDnodeType **WITH** {

```
CPUcnt : 1;                                  %number of CPUs
MPL : 50;                                    %multiprogramming level
CPUrate : 50.0;                              %CPU rate, measured in MIPS
%The following costs are expressed
%as instruction counts
ATTACH_TASK_COST : 15100.0;                  %cost of attaching a transaction
                                             %from a thread of control
DM_INTERFACE_COST : 2000.0;                  %fixed cost for access to the DB
DM_CALL_COST : 4000.0;                       %average cost for executing a DB
                                             % access call
DM_IO_COST : 10000.0;                        %fixed cost for access to an
                                             %I/O device
FUNCTION_SHIP_SEND_COST : 12600.0;           %cost of sending a remote request
FUNCTION_SHIP_RECV_COST : 12600.0;           %cost of receiving a remote response
%The following four parameters define the cost
%of the 2-phase commit protocol
%(coordinator side)
PRIMARY_PREPARE_COST : 70000;
SEND_PREPARE_COST : 12600.0;
SEND_COMMIT_COST : 12600.0;
PRIMARY_COMMIT_COST : 14000.0;
%The following five parameters define the cost
%for the two-phase commit protocol
%and the cost of logging at the participant side
RECV_PREPARE_COST : 12600.0;
RECV_COMMIT_COST : 12600.0;
LOG_IO_COST : 5000.0;                        %cost of logging
SECONDARY_PREPARE_COST : 12000.0;
SECONDARY_COMMIT_COST : 12000.0;
DETACH_TASK_COST : 15100.0;                  %cost of detaching a transaction
                                             %from a thread of control
SEND_MESSAGE_COST : 2000.0;                  %processing cost of sending a msg
                                             %to a Concurrency Manager
RECV_MESSAGE_COST : 2000.0;                  %processing cost of receiving a msg
                                             %from a Concurrency Manager
DM_BUFFER_SIZE : 20000;                      %size of local buffer: 160 MB
                                             %(8 KB pages)
LOG_IO_DEVICE : LogDisk;                     %type of I/O device used for
                                             %logging
BUFFER_INV_COST : 10000.0;                   %buffer invalidation cost
}
```

Επίσης μπορεί κανείς να δει ότι υπάρχει ένα κόστος για την αποστολή και λήψη μηνύματος (send και receive message) τα οποία απαιτούνται για προσπέλαση στα κοινά

δεδομένα και για την αποτροπή ταυτόχρονης μεταβολής των δεδομένων από δύο δοσοληψίες που είναι ενεργές σε δύο διαφορετικούς κόμβους και ένα κόστος για την ακύρωση σελίδων αποθηκευμένων στον ιδιωτικό ενταμιευτή (buffer invalidation cost).

Πλέον ο ορισμός στιγμιοτύπου της παραπάνω κλάσης είναι εύκολος. Για να ορίσουμε έναν κόμβο SDProcessingNode του τύπου SDnodeType αρκεί να ορίσουμε το παρακάτω:

- **DEFINE SIMPLE_NODE** SDProcessingNode **OF CLASS** SDnodeType;

Επίσης ο προσομοιωτής επιτρέπει τον ορισμό κόμβων που έχουν ως αποκλειστική αρμοδιότητα την δρομολόγηση των δοσοληψιών στους άλλους κόμβους. Αυτό είναι ευρύτατα διαδεδομένο στα υπάρχοντα συστήματα επεξεργασίας δοσοληψιών. Αυτοί οι κόμβοι αποκαλούνται front-end κόμβοι σε αντίθεση με τους κόμβους που εξυπηρετούν δοσοληψίες που αποκαλούνται back-end κόμβοι. Ο τρόπος ορισμού είναι περίπου ίδιος.

- **DEFINE SIMPLE_NODE** SDFrontEndNode **OF CLASS** SDnodeType (**FRONT_END**);

Απαραίτητα πριν από τον ορισμό ενός στιγμιοτύπου μίας κλάσης κόμβου θα πρέπει να έχει οριστεί η κλάση αυτή. Επίσης μπορεί να οριστεί ένας οποιοσδήποτε αριθμός από κόμβους ως στιγμιότυπο μίας κλάσης.

4.2.3 Ομάδες κόμβων

Είναι δυνατός και ο ορισμός ομάδων από κόμβους. Με τον αυτόν τρόπο είναι δυνατό τα μέλη μίας ομάδας να επικοινωνούν μεταξύ τους μέσω ενός τοπικού δικτύου, το οποίο αφαιρετικά μπορεί να θεωρηθεί ως ένα κοινόχρηστο κανάλι επικοινωνίας. Επίσης είναι δυνατό να οριστεί μία ιεραρχική δομή στην τοπολογία των κόμβων (περιλαμβάνοντας ως μέλη της ομάδας άλλες ομάδες) επικοινωνώντας διαμέσου ενός δικτύου κορμού (back-bone network). Τελικά πάντως θα πρέπει κάθε κόμβος που ορίζεται στο αρχείο εισόδου να ανήκει σε μία ομάδα και όλες μαζί οι ομάδες να είναι υποομάδες μίας υπερομάδας (super-cluster).

Δύο τρόποι ορισμού ομάδων υποστηρίζονται από την γλώσσα προδιαγραφής του προσομοιωτή:

1. Ο τρόπος ορισμού ομογενών ομάδων δηλ. ομάδων που απαρτίζονται από κόμβους του ιδίου τύπου και επικοινωνούν μέσω ενός κοινόχρηστου καναλιού επικοινωνίας π.χ. Ethernet bus.

2. Με απαρίθμηση των κόμβων όπου είναι δυνατό οι κόμβοι αυτοί να είναι ετερογενείς δημιουργώντας έτσι και ετερογενείς ομάδες.

Στον ορισμό της ομάδας (cluster) επιλέχθηκε να προστεθεί και ο ορισμός των κοινόχρηστων δίσκων. Έτσι όλα τα μέλη μίας ομάδας έχουν άμεση πρόσβαση στους δίσκους. Υπάρχει και ορισμός ομάδας που δεν περιλαμβάνει κοινόχρηστους δίσκους (*Shared Nothing* αρχιτεκτονική) για τον οποίο μπορεί κανείς να δει την γραμματική ή το εγχειρίδιο χρήσης του προσομοιωτή. Η συντακτική δομή που χρησιμοποιείται για τον ορισμό μίας ομάδας με κοινόχρηστους δίσκους είναι η SIMPLE_NODE_CLUSTER. Ας δούμε πρώτα όμως δύο παραδείγματα των παραπάνω τρόπων ορισμού ομάδων.

- **DEFINE SIMPLE_NODE_CLUSTER** SDHomogeneousCluster **OF CLASS** SDnodeType **WITH** {
  numNodes : 8; %number of nodes in cluster
  packetSize : 1024; %packet size (in bytes)
  transferRate : 500000; %measured in bytes per sec (4MBit)
  busArbitration : RANDOM; %RANDOM or FIFO
  DEFINE SHARED_DATABASE Shared_DB1 **WITH** {
  IO_DEVICE : DataDisk; %type of I/O device (disk)
  NUM_DISKS : 3; } %num of disks
  }
- **DEFINE SIMPLE_NODE** Node1 **OF CLASS** SDnodeType1;
- **DEFINE SIMPLE_NODE** Node2 **OF CLASS** SDnodeType2;
- **DEFINE SIMPLE_NODE_CLUSTER** SDHeterogeneousCluster **WITH** {
  numNodes : 3; %number of nodes in cluster
  packetSize : 1024; %packet size (in bytes)
  transferRate : 500000; %measured in bytes per sec (4MBit)
  busArbitration : RANDOM; %RANDOM or FIFO
  MEMBER_NODES : {Node1, Node2, SDHomogeneousCluster}
  DEFINE SHARED_DATABASE Shared_DB2 **WITH** {
  IO_DEVICE : DataDisk; %type of I/O device (disk)
  NUM_DISKS : 3; } %num of disks
  }

Ο προσομοιωτής δίνει την δυνατότητα ορισμού του μεγέθους του πακέτου αποστολής στο δίκτυο καθώς και του ρυθμού μετάδοσης. Στον πρώτο τρόπο ορισμού δεν απαιτείται να έχουν οριστεί τα ονόματα των μελών της ομάδας. Με τον ορισμό της ομάδας τα μέλη της παίρνουν το όνομα της ομάδας με την προσθήκη της κατάληξης _node_i όπου i ο αριθμός του κόμβου (από 1 έως τον αριθμό των κόμβων της ομάδας που έχει ορίσει ο χρήστης). Μέσα στον ορισμό της συντακτικής δομής της ομάδας υπάρχει και ο ορισμός της κοινόχρηστης βάσης όπου δηλώνεται ο αριθμός των δίσκων και ο τύπος τους. Ο τύπος αυτός θα πρέπει να έχει δηλωθεί προηγουμένως ενώ η ομάδα μπορεί να έχει δίσκους μόνο ενός τύπου (απεριόριστο πάντως αριθμό). Στον δεύτερο τρόπο ορισμού ορίζεται μία ετερογενής ομάδα που απαρτίζεται από δύο κόμβους Node1, Node2 δύο διαφορετικών τύπων, για τους οποίους ο τύπος έχει δηλωθεί πιο πριν αλλά και από μία άλλη ομάδα. Ο λόγος που φαίνονται και οι δηλώσεις των δύο κόμβων είναι διότι ο ορισμός με απαρίθμηση προϋποθέτει να έχουν ήδη οριστεί (οπότε

έχουν κάποιο όνομα). Δεν παίρνουν κάποια ονομασία με αυτόματο τρόπο όπως στην πρώτη περίπτωση.

4.2.4 Κοινόχρηστος ενταμιευτής

Για τον ορισμό ενός κοινόχρηστου ενταμιευτή, ώστε να προσομοιάσει κανείς την *Shared Intermediate Memory* αρχιτεκτονική, αρκεί να δοθεί ένας ορισμός μίας ομάδας κόμβων και στη συνέχεια να χρησιμοποιηθεί η συντακτική δομή SHARED_BUFFER που επιτρέπει τον ορισμό ενός ενταμιευτή κοινόχρηστου για την παραπάνω ομάδα. Παράδειγμα τέτοιου ορισμού είναι το :

- **DEFINE SHARED_BUFFER** SimBuf **IN SIMPLE_NODE_CLUSTER**
  SDHomogeneousCluster **WITH** {
  SB_BUFFER_SIZE : 10000; %number of pages of the shared buffer
  IO_COPY_DELAY : 0.0025; %time to transfer a block tofrom the local buffers
  SB_DB_COPY_DELAY : 0.007; %time to transfer a block to the shared database
  %the following are in # of instructions
  SB_CALL_COST : 2000.0; %processing cost for accessing the shared buffer
  SB_IO_COST : 5000.0; %processing cost for I/O between Site and buffer
  }

Ορίζεται το μέγεθος του buffer σε αριθμό σελίδων, ο χρόνος για την μεταφορά μιας σελίδας στην τοπική μνήμη ενός κόμβου και στους δίσκους και τέλος ένα κόστος για τον έλεγχο του buffer προκειμένου να βρεθεί κάποια σελίδα και ένα κόστος για I/O στο buffer.

4.2.5 Βάση Δεδομένων

Ο τρόπος περιγραφής της βάσης δεδομένων στην οποία κάνουν προσπέλαση οι δοσοληψίες του συστήματος δεν έχει μεταβληθεί σε σχέση με την προϋπάρχουσα υλοποίηση. Το σχήμα της κατανεμημένης βάσης δεδομένων ορίζεται με την χρήση της RELATION συντακτικής δομής η οποία καθορίζει το σχήμα της βάσης δεδομένων κατά το σχεσιακό μοντέλο. Επίσης είναι δυνατός ο ορισμός ενός σχήματος **τμηματοποίησης** (fragmentation schema) μιας σχέσης.

Η τμηματοποίηση μπορεί να είναι **οριζόντια** (horizontal fragmentation) οπότε η σχέση χωρίζεται σε έναν αριθμό τμημάτων ανάλογα με το selection statement που χρησιμοποιείται ή **κατακόρυφη** (vertical fragmentation) οπότε τα τμήματα της σχέσης απαρτίζονται από ένα υποσύνολο των πεδίων της σχέσης και του πρωτεύοντος κλειδιού.

Σε κάθε σχέση της βάσης δεδομένων αναγράφεται ο αριθμός των πλειάδων για κάθε σελίδα, το πρωτεύον κλειδί αλλά και το πεδίο ή τα πεδία για τα οποία υπάρχει δεικτοδότηση (indexing). Ακολουθούν και πάλι ορισμένα παραδείγματα για να γίνουν πιο κατανοητά τα παραπάνω:

- **DEFINE RELATION** Account **WITH** { % definition of primary relation schema
  ATTRIBUTES {
  Account_ID: NUMERIC;
  Branch_ID: NUMERIC;
  Account_Balance: NUMERIC;
  Account_Interest_Level: SYMBOLIC;
  }
  tuplesPerPage: 20.0; **KEY** = { Account_ID };
  CLUSTERED_INDEX ON { Account_ID };
  INDEX ON { Account_ID, Branch_ID };
  }

- **DEFINE HORIZONTAL FRAGMENTATION OF** Account **AS** {
  TblH1 **WITH** ((Account_Balance >= 100000) **AND** (Account_Balance < 1000000)),
  tuplesPerPage : 20.0;
  TblH2 **WITH** (**NOT**(Account_Balance >= 100000) **OR** (Account_Balance >= 1000000)),
  tuplesPerPage : 20.0;
  }

- **DEFINE VERTICAL FRAGMENTATION OF** TblH2 **AS** {
  TblH2V1 **WITH COLUMNS** { Account_ID, Account_Balance },
  tuplesPerPage : 40.0;
  TblH2V2 **WITH COLUMNS** { Account_ID, Branch_ID },
  tuplesPerPage : 25.0;
  }

Να σημειωθεί ότι στην τωρινή υλοποίηση δεν υποστηρίζονται το indexing ή ο αριθμός των πλειάδων που χωρούν σε κάθε σελίδα. Η σκέψη ήταν να υποστηρίζονται για μία μελλοντική χρήση του προσομοιωτή ώστε να είναι δυνατός ο ορισμός δηλώσεων (statements) που μοιάζουν με την SQL γλώσσα.

Ακόμα υπάρχει και η συντακτική δομή INSTANCE για τον καθορισμό του σχήματος ανάθεσης των σχέσεων. Καθορίζει σε ποιό δίσκο βρίσκεται αποθηκευμένη η σχέση και μπορεί να οριστεί για σχέση ή για fragment της σχέσης. Επίσης καθορίζει και το πλήθος των αντικειμένων της σχέσης.

- **DEFINE INSTANCE OF RELATION** Account **AT**
  { SDHomogeneousCluster_node_1_disk_1 } **OF SIMPLE_NODE_CLUSTER**
  SDHomogeneousCluster **WITH numItems**: 50000;

- **DEFINE INSTANCE OF RELATION** TblH2 **AT**
  { SDHomogeneousCluster_node_1_disk_1 } **OF SIMPLE_NODE_CLUSTER**
  SDHomogeneousCluster **WITH numItems**: 50000;

4.2.6 Φόρτος εργασίας

Τέλος ορίζεται ο φόρτος εργασίας που καλείται να εξυπηρετήσει το σύστημα. Προκειμένου να γίνει αυτό πρέπει να οριστούν οι κλάσεις δοσοληψιών και οι κλάσεις χρηστών. Κάθε κλάση αντιπροσωπεύει μια κατηγορία από δοσοληψίες ή χρήστες με παρόμοια χαρακτηριστικά.

Ο ορισμός μίας κλάσης δοσοληψιών γίνεται με την συντακτική δομή TRANSACTION_CLASS. Κατά τον ορισμό της κλάσης καθορίζεται ο αριθμός των εντολών που εκτελούνται από το πρόγραμμα ενδιάμεσα από δύο συνεχόμενες προσπελάσεις σε δεδομένα (application burst length) μαζί με τον ελάχιστο και μέγιστο αριθμό από προσπελάσεις στην βάση δεδομένων. Επίσης ορίζονται οι πιθανότητες για προσπέλαση της κλάσης σε μία σχέση ή τμήμα της μαζί με την πιθανότητα η προσπέλαση να γίνει για εγγραφή. Μια κλάση μπορεί να κάνει προσπελάσεις σε οποιοδήποτε αριθμό από σχέσεις αλλά συνολικά η πιθανότητα θα πρέπει να αθροίζει στην μονάδα. Με αυτόν τρόπο καθορίζεται και η τοπικότητα (locality) των κλήσεων μίας κλάσης δοσοληψιών. Υποθέτουμε ότι κάθε σελίδα της σχέσης έχει την ίδια πιθανότητα προσπέλασης σε σχέση με οποιαδήποτε άλλη που βρίσκεται στην ίδια σχέση. Υποστηρίζονται επίσης και αλυσίδες δοσοληψιών (workflows) που ορίζονται ως μία ακολουθία από δοσοληψίες διαφορετικών κλάσεων. Μαζί με κάθε κλάση ή workflow δίνεται η δυνατότητα να οριστεί και ένας στόχος επίδοσης για την περίπτωση όπου στην προσομοίωση γίνει χρήση αλγορίθμων που λαμβάνουν υπόψη στόχους επίδοσης.

- **DEFINE TRANSACTION_CLASS** TxClassA **AS** {
  % definition of a transaction class
  applicationBurst: 100000.0;
  blocksAccessed_min: 1;
  blocksAccessed_max: 16;
  accessProbability OF INSTANCE Account: 0.5;
  writeProbability OF INSTANCE Account: 0.7;
  accessProbability OF INSTANCE TblH2V1: 0.5;
  writeProbability OF INSTANCE TblH2V1: 0.25;
  }
- **DEFINE WORKFLOW_CLASS** Workflow **AS** {
  program: TxChain;
  chain: { TxClassB, TxClassA, TxClassB };
  }
- **DEFINE PERFORMANCE_GOAL OF CLASS** TxClassB: 2.0;
- **DEFINE PERFORMANCE_GOAL OF CLASS** Workflow: 5.0;

Ο ορισμός κλάσεων χρηστών γίνεται με την συντακτική δομή CLIENT_CLASS και ορίζεται η πιθανότητα αποστολής μίας κλάσης ή ενός workflow καθώς και ένα χρονικό διάστημα το οποίο θα αναφέρουμε παρακάτω τον τρόπο που λειτουργεί. Με τον ορισμό τέλος του SYNTHETIC WORKLOAD καθορίζεται το πού βρίσκονται οι χρήστες του συστήματος

(τερματικά) και τι τύπου είναι. Συνήθως όταν υιοθετείται front-end κόμβος τότε επιλέγεται να είναι και ο κόμβος στον οποίο είναι συνδεδεμένα και τα τερματικά.

- **DEFINE CLIENT_CLASS** clientType **AS** {
  % definition of user class
  arrivalRate: 20.0; % think-time ...
  SUBMIT TxClassA **WITH PROBABILITY** 0.5;
  % 50 % the execution of transactions of class 'TxClassA'.
  SUBMIT TxClassB **WITH PROBABILITY** 0.4;
  SUBMIT Workflow **WITH PROBABILITY** 0.1;
  }
- **DEFINE SYNTHETIC WORKLOAD** SystemLoad **AS** {
  % definition of workload
  numClients OF CLASS clientType: 15 **AT ALL_SITES**;
  % At each site, there are 15 active users of class 'clientType'.
  numClients OF CLASS clientType2: 1 **AT** SDHomogeneous_node_1;
  % There is 1 active user of class 'clientType2' at site SDHomogeneous_node_1.
  numClients OF CLASS clientType3: 10 **AT**
  SDHomogeneous_node_1,SDHomogeneous_node_2;
  % There are 10 active users of class 'clientType3' at
  % each of the two nodes.
  }

Υπάρχει η δυνατότητα να οριστούν χρήστες σε όλους του κόμβους (με χρήση του ALL_SITES token), αλλά και χρήστες σε έναν ή περισσότερους κόμβους, διευκολύνοντας έτσι την περιγραφή.

Ο χρόνος που αναφέρθηκε προηγουμένως ότι ορίζεται για τις κλάσεις των χρηστών έχει διαφορετική σημασία ανάλογα με το μοντέλο που χρησιμοποιείται για την μοντελοποίηση του φόρτου εργασίας. Καταρχήν υπάρχουν δύο μοντέλα :το κλειστό και το ανοικτό.

Στο κλειστό μοντέλο φόρτου εργασίας υπάρχει ένας αριθμός από χρήστες που εκτελεί τον εξής κύκλο:

1. Υποβάλει μία αίτηση εξυπηρέτησης για μία δοσοληψία που ανήκει σε μία από τις κλάσεις για τις οποίες η πιθανότητα αποστολής είναι διαφορετική από μηδέν. Η αίτηση μεταβιβάζεται στον κόμβο που έχει αναλάβει την δρομολόγηση και ο οποίος αναλαμβάνει να επιλέξει τον κόμβο που θα εξυπηρετήσει την αίτηση.
2. Ο χρήστης αναμένει την απάντηση στην αίτηση εξυπηρέτησης.
3. Αφού λάβει την απάντηση, ο χρήστης περιμένει για ένα χρονικό διάστημα με μέση διάρκεια ίση με τον προαναφερθέντα χρόνο (think time). Αμέσως μετά αποστέλλει την επόμενη αίτησή του (βήμα 1).

Με αυτό το μοντέλο ο αριθμός των αιτήσεων που βρίσκονται στο σύστημα είναι το πολύ ίσος με τον αριθμό των χρηστών του συστήματος. Εναλλακτικά, όπως συμβαίνει στο ανοικτό μοντέλο, ο χρήστης δεν περιμένει απάντηση. Αντί αυτής, η πηγή φόρτου στέλνει αιτήσεις με ρυθμό που καθορίζεται από την παράμετρο χρόνου της κλάσης χρηστών. Πιο συγκεκριμένα το χρονικό διάστημα μεταξύ δύο διαδοχικών αιτήσεων καθορίζεται ως μια τυχαία μεταβλητή που ακολουθεί την εκθετική κατανομή και έχει μέση τιμή ίση με την δοθείσα χρονική παράμετρο. Όπως είναι φανερό, στο ανοικτό μοντέλο το σύστημα αν δεν είναι ικανό να εξυπηρετήσει τον ρυθμό άφιξης των αιτήσεων, τελικά θα κατακλυστεί από αιτήσεις με αποτέλεσμα να γίνονται πολλά timeout και το πρόβλημα του ανταγωνισμού για τα δεδομένα (data contention) να γίνει πολύ έντονο.

4.3 Προσομοίωση

Έχοντας ορίσει μέσω της γραμματικής το μοντέλο του συστήματος, ο σαρωτής αναλαμβάνει να κατασκευάσει τον πίνακα συμβόλων όπου κρατείται κάθε πληροφορία για το μοντέλο. Κάνοντας χρήση αυτού του πίνακα, ο προσομοιωτής κάνει διαδοχικές κλήσεις σε ρουτίνες της βιβλιοθήκης υποστήριξης ώστε να μοντελοποιήσει το σύστημα. Βασική έννοια - οντότητα αποτελεί ο "κόμβος επεξεργασίας" που μοντελοποιεί έναν πόρο του συστήματος. Όπως είπαμε προηγουμένως, ο προσομοιωτής βασίζεται σε νήματα ελέγχου. Τα νήματα ελέγχου αντιπροσωπεύουν μια σειριακή ροή ελέγχου μέσα σε ένα πρόγραμμα και προκειμένου να επιτευχθεί συγχρονισμός μπορεί να απαιτείται η χρήση σημαφόρων (semaphores) ή άλλων τεχνικών. Τα νήματα ελέγχου σχετίζονται με έναν κόμβο επεξεργασίας, ενώ κάθε κόμβος επεξεργασίας μπορεί να σχετίζεται με περισσότερα από ένα νήματα ελέγχου. Η βιβλιοθήκη παρέχει και μηχανισμούς για την κατανάλωση πόρων. Η κλήση τους από τα νήματα ελέγχου προκειμένου να χρεώσουν έναν πόρο για ορισμένες λειτουργίες (π.χ. αποστολή μηνύματος), "προχωρά" τον ρολόι της προσομοίωσης (simulation clock).

Τα νήματα ελέγχου επικοινωνούν κάνοντας χρήση της θύρας επικοινωνίας. Κάθε νήμα ελέγχου, κατά την δημιουργία, του συνδέεται με μια θύρα επικοινωνίας και στη συνέχεια μπορεί να δημιουργήσει και άλλες με δυναμικό τρόπο. Βασικός τρόπος επικοινωνίας των διαφόρων νημάτων ελέγχου είναι τα μηνύματα τα οποία ανταλλάσσονται. Όλα τα μηνύματα που απευθύνονται σε ένα νήμα, πηγαίνουν στη θύρα επικοινωνίας την οποία γνωρίζουν όλα τα άλλα νήματα. Μαζί με κάθε μήνυμα υπάρχει και μία χρονοσφραγίδα (timestamp).

Ο TPsim υλοποιεί ένα μοντέλο προσομοίωσης διακριτών συμβάντων (discrete event simulation model). Σε αυτό το μοντέλο οι αλλαγές στις καταστάσεις σημειώνονται σε διακριτούς χρόνους του χρόνου προσομοίωσης. Η μετάβαση από μία κατάσταση σε μία άλλη γίνεται όταν εμφανιστεί ένα συμβάν (event).

Η μέθοδος προσομοίωσης διακριτών συμβάντων είναι χρήσιμη για την προσομοίωση ασύγχρονων συστημάτων (όπως ο TPsim) όπου η εμφάνιση συμβάντων δεν είναι συγχρονισμένη με το ρολόι του συστήματος. Υπάρχουν τρεις βασικές δομές: οι μεταβλητές κατάστασης που περιγράφουν την κατάσταση του συστήματος, ο κατάλογος συμβάντων όπου κρατούνται τα συμβάντα τα οποία δεν έχουν ληφθεί υπόψη οπότε και δεν έχουν επιφέρει αλλαγή στο σύστημα και τέλος το ρολόι προσομοίωσης το οποίο ενημερώνεται

βάσει των χρονοσφραγίδων των συμβάντων που λαμβάνει ο προσομοιωτής από τον κατάλογο συμβάντων. Σε κάθε περίπτωση λαμβάνεται το συμβάν με την μικρότερη χρονοσφραγίδα ώστε να εξασφαλιστεί ότι δεν παραβιάζεται η αρχή της αιτιότητας. Η αρχή αυτή ορίζει ότι το αίτιο προηγείται πάντοτε του αποτελέσματος.

Κατά την προσομοίωση, το πρώτο νήμα ελέγχου που ξεκινά είναι ένα νήμα ελέγχου της βιβλιοθήκης ελέγχου (genesis) το οποίο και αναλαμβάνει να ξεκινήσει όλα τα άλλα. Όταν τελειώσει αυτή η διαδικασία ξεκινά και τον μηχανισμό που προσομοιώνει τις αφίξεις αιτήσεων στο σύστημα. Ο συγχρονισμός γίνεται με χρήση δύο σημαφόρων που υλοποιούν μια δομή συγχρονισμού τύπου barrier. Αμέσως μετά το νήμα ελέγχου γίνεται suspend από την ουρά των νημάτων ελέγχου.

4.4 Προσομοίωση υποσυστημάτων υλικού

Για την προσομοίωση του υλικού υποσυστήματος δημιουργούνται στιγμιότυπα του τύπου κόμβου επεξεργασίας με την χρήση ρουτίνων της βιβλιοθήκης υποστήριξης. Δημιουργείται έτσι ένα στιγμιότυπο για κάθε κόμβο του μοντέλου που περιγράφεται στο αρχείο εισόδου αλλά και για κάθε συσκευή αποθήκευσης και κάθε δίκτυο διασύνδεσης.

Για τους κόμβους που αντιστοιχούν σε κόμβους του μοντέλου ορίζεται η πολιτική χρονοπρογραμματισμού τους να είναι η preemptive priority με καθοριζόμενο από τον χρήστη quantum χρονοπρογραμματισμού. Ο βαθμός χρησιμοποίησης του καθορίζεται από την κατανάλωση του πόρου που γίνεται από τα νήματα ελέγχου που έχουν σχετιστεί με αυτόν τον πόρο.

Για τους πόρους που αντιστοιχούν σε συσκευές αποθήκευσης ή δικτύου διασύνδεσης ορίζεται η FCFS (First Come First Served) ως πολιτική χρονοπρογραμματισμού.

4.5 Προσομοίωση υποσυστημάτων λογισμικού

Για κάθε κόμβο επεξεργασίας δοσοληψιών που ορίζεται στο αρχείο εισόδου του προσομοιωτή δημιουργούνται και τα εξής νήματα ελέγχου : διαχειριστής επικοινωνίας (communication manager), διαχειριστής φόρτου (load manager), διαχειριστής ελέγχου ταυτόχρονης πρόσβασης (concurrency control manager), διαχειριστής δοσοληψιών (transaction manager), διαχειριστής δεδομένων (data manager), διαχειριστής του μηχανισμού καταγραφής μεταβολών (log manager) διαχειριστής περιφεριακής μνήμης του μηχανισμού καταγραφής μεταβολών (log storage manager), διαχειριστής για την εύρεση τοπικών αδιεξόδων (deadlock detector manager), διαχειριστής ουράς (queue manager), διαχειριστής τοπικών ενταμιευτών (local buffer manager) και ο διαχειριστής περιφεριακής μνήμης (storage manager).

Ο διαχειριστής επικοινωνίας (communication manager) αποτελεί τον μεσάζοντα στην λήψη και αποστολή μηνυμάτων από και προς το δίκτυο επικοινωνίας. Χρησιμοποιείται για την ανταλλαγή μηνυμάτων όταν αυτά απευθύνονται σε νήματα ελέγχου που βρίσκονται σε διαφορετικούς κόμβους του συστήματος. Όταν το μήνυμα απευθύνεται σε κόμβο του τοπικού

δικτύου τότε μπορεί να διαβιβαστεί απευθείας στον κόμβο αλλιώς πρέπει να δρομολογηθεί το μήνυμα στο backbone δίκτυο.

Ο διαχειριστής φόρτου (load manager) έχει ως σκοπό την παρακολούθηση της κατάστασης του συστήματος. Ενεργοποιείται περιοδικά προκειμένου να μεταβάλει την κατάσταση του κόμβου ώστε αυτή να είναι δυνατό να χρησιμοποιηθεί π.χ. από αλγορίθμους δρομολόγησης που λαμβάνουν υπόψη τους την κατάσταση του συστήματος.

Ο διαχειριστής δοσοληψιών (transaction manager) διατηρεί πληροφορίες για τις δοσοληψίες που είναι ενεργές σε έναν κόμβο και φροντίζει για την ικανοποίηση των ιδιοτήτων της μονιμότητας και ατομικότητας, συντονίζοντας τους διαχειριστές πόρων του συστήματος και του μηχανισμού καταγραφής μεταβολών.

Ο διαχειριστής για την εύρεση τοπικών αδιεξόδων (deadlock detector manager) ενεργοποιείται περιοδικά προκειμένου να ελέγξει εάν οι δοσοληψίες που εκτελούνται σε έναν κόμβο έχουν εμπλακεί σε αδιέξοδο. Εξετάζεται ο lock table και σχηματίζεται ένας γράφος που παριστάνει την σχέση wait-for μεταξύ των δοσοληψιών. Κύκλος σε αυτόν τον γράφο παριστάνει την πιθανή ύπαρξη αδιεξόδου οπότε μία από τις δοσοληψίες του κύκλου τερματίζεται. Ο διαχειριστής υλοποιεί μία πολιτική αποφυγής αδιεξόδων (deadlock avoidance). Ανάλογα με τις πληροφορίες που κρατάει ο προσομοιωτής σε κάθε αρχιτεκτονική στον lock table, χρειάζεται και διαφορετική αντιμετώπιση.

Ο διαχειριστής ουράς (queue manager) υλοποιεί την λειτουργία των ουρών αναμονής στα συστήματα επεξεργασίας δοσοληψιών. Παρέχει μια σειρά από βασικές υπηρεσίες επικοινωνίας.

Ο διαχειριστής περιφεριακής μνήμης (storage manager) προσομοιώνει την λειτουργία διαχείρισης της βάσης δεδομένων που βρίσκεται στα αποθηκευτικά μέσα του κόμβου. Αποφασίζει σε ποιον διαχειριστή χρονοπρογραμματισμού των αιτήσεων προσπέλασης θα στείλει την αίτηση. Κάθε δίσκος έχει έναν τέτοιο διαχειριστή και εφόσον ένας κόμβος μπορεί να έχει πολλούς δίσκους, αποφασίζει σε ποιον από όλους όσους ελέγχει έχει το αποθηκευμένο αρχείο. Για την περίπτωση κόμβων που δεν έχουν τοπικούς δίσκους, όπως η αρχιτεκτονική που μελετάμε, η αίτηση προσπέλασης μεταβιβάζεται στον διαχειριστή κοινόχρηστης περιφεριακής μνήμης (shared storage manager) ο οποίος δημιουργείται για κάθε ομάδα από κόμβους. Με αυτόν τον τρόπο είναι δυνατό μελλοντικά να επεκταθεί ο προσομοιωτής ώστε παράλληλα με τους κοινόχρηστους δίσκους να μπορεί να έχει και δεδομένα αποθηκευμένα σε τοπικούς δίσκους.

Ανάλογος με τον διαχειριστή περιφεριακής μνήμης είναι και ο διαχειριστής περιφεριακής μνήμης του μηχανισμού καταγραφής μεταβολών με την μοναδική διαφορά ότι πάντα υπάρχει ένας τοπικός δίσκος για την αποθήκευση του αρχείου μεταβολών.

Ο διαχειριστής δεδομένων (data manager) διατηρεί μία ουρά αιτήσεων δοσοληψιών για προσπελάσεις σε δεδομένα και μία ουρά απαντήσεων από τους log και storage managers. Παράλληλα υλοποιούν το πρωτόκολλο δέσμευσης της εκάστοτε αρχιτεκτονικής. Εάν μία αίτηση δεν μπορεί να ικανοποιηθεί τότε μπαίνει σε μια λίστα αναμονής ή ακόμα μπορεί και να τερματιστεί αν δοθεί εντολή π.χ. από τον deadlock detector. Αλλιώς ενημερώνεται ο lock table και στη συνέχεια αποστέλλεται μία απάντηση στην δοσοληψία που έκανε την αίτηση.

Ο διαχειριστής του τοπικού ενταμιευτή (buffer manager) ελέγχει τον τοπικό ενταμιευτή υποστηρίζοντας μία σειρά από πράξεις ώστε να είναι σε θέση να μεταφέρει, από και προς τους δίσκους, δεδομένα της βάσης δεδομένων. Υποστηρίζει την LRU (Least Recently Used) πολιτική αντικατάστασης σελίδων που ορίζει ότι σε περίπτωση αντικατάστασης σελίδας επιλέγεται η σελίδα στην οποία έχει γίνει η πιο παλιά πρόσβαση. Προκειμένου να γίνει αυτό, οι σελίδες βρίσκονται σε μία λίστα υλοποιώντας έτσι την lru αλυσίδα. Συνεργάζεται στενά με τον διαχειριστή του ελέγχου ταυτόχρονης πρόσβασης προκειμένου να εντοπίζει τις μη έγκυρες σελίδες που είναι αποθηκευμένες στον buffer. Στην περίπτωση όπου μία σελίδα δεν είναι έγκυρη, γίνεται set το bit που φανερώνει ότι η σελίδα δεν είναι valid. Με αυτόν τον τρόπο η σελίδα ουσιαστικά γίνεται υποψήφια για αντικατάσταση.

Για τους διαχειριστές του μηχανισμού καταγραφής μεταβολών και του ελέγχου ταυτόχρονης πρόσβασης θα μιλήσουμε αναλυτικά παρακάτω. Παράλληλα με αυτά τα νήματα ελέγχου, υπάρχουν και ορισμένα άλλα νήματα ελέγχου που σχετίζονται με ένα cluster ή με το δίκτυο διασύνδεσης.

Το δίκτυο διασύνδεσης (network model) αναπαριστά ένα απλό μοντέλο προσομοίωσης συμβάντων διακριτού χρόνου σε ένα κανάλι επικοινωνίας υιοθετώντας την FCFS πολιτική εξυπηρέτησης (infinite FIFO server).

Ο διαχειριστής για την εύρεση αδιεξόδων ανάμεσα σε ομάδες κόμβων (global deadlock detector) ενεργοποιείται ανά τακτά χρονικά διαστήματα προκειμένου να ανιχνεύσει αδιέξοδα που δημιουργούνται από δοσοληψίες που είναι ενεργές σε διαφορετικούς κόμβους. Έχει υπό την εποπτεία του μία ομάδα (cluster) κόμβων.

Ο διαχειριστής του κοινόχρηστου ενταμιευτή (shared buffer manager) έχει υπό την επίβλεψή του τον κοινόχρηστο ενταμιευτή. Σε αυτόν αποθηκεύονται μόνο σελίδες που έχουν μεταβληθεί σε έναν από τους κόμβους του cluster. Εφόσον είναι μη πτητικός τα αποθηκευμένα δεδομένα επιζούν μετά από μία βλάβη (που δεν έχει σχέση με τον κοινόχρηστο ενταμιευτή). Έτσι πριν το commit μίας δοσοληψίας αρκεί να μεταφερθεί η σελίδα στον κοινόχρηστο ενταμιευτή. Εφόσον αυτός είναι αρκετά πιο γρήγορος σε σχέση με τους δίσκους, βελτιώνει σε μεγάλο βαθμό την απόκριση του συστήματος [YD94b], [YD94a], [BHR90]. Και γι' αυτόν τον ενταμιευτή, η πολιτική αντικατάστασης σελίδων είναι η LRU. Όταν αποφασίσει ότι πρέπει να αντικατασταθεί μία σελίδα τότε βγάζει την σελίδα που βρίσκεται στο τέλος της λίστας και την μεταφέρει στους δίσκους του συστήματος. Στην θέση της στον ενταμιευτή βάζει τη νέα σελίδα και τη τοποθετεί στην αρχή της λίστας που κρατά την lru αλυσίδα.

Ο διαχειριστής χρονοπρογραμματισμού των αιτήσεων προσπέλασης σε δίσκους (I/O scheduler), λαμβάνει αιτήσεις από έναν storage manager και προγραμματίζει την εκτέλεσή τους από τον ελεγχόμενό του δίσκο. Ο δίσκος εξυπηρετεί με FCFS τρόπο τις αιτήσεις αλλά ο I/O scheduler υλοποιεί ένα αλγόριθμο για την ελαχιστοποίηση της κίνησης της κεφαλής του δίσκου (SCAN αλγόριθμος).

Ο διαχειριστής δίσκου προσομοιώνει την λειτουργία ενός μέσου (δίσκου) αποθήκευσης δεδομένων. Η θέση της κεφαλής σε μία δεδομένη χρονική στιγμή μαζί με την θέση που βρίσκεται το block στο οποίο γίνεται η προσπέλαση, καθώς και οι καθυστερήσεις για περιστροφή και κίνηση της κεφαλής, καθορίζουν τον χρόνο που χρειάζεται για να γίνει μία

προσπέλαση.

4.5.1 Έλεγχος ταυτόχρονης πρόσβασης

Εφόσον όλα τα αντικείμενα στη βάση δεδομένων είναι προσβάσιμα από πολλούς κόμβους στις αρχιτεκτονικές *Shared Disk* και *Shared Intermediate Memory* , απαιτείται η χρήση ενός μηχανισμού συγχρονισμού ανάμεσα σε όλους αυτούς τους κόμβους προκειμένου να μην παραβιαστεί η ιδιότητα της σειριοποίησης στην επεξεργασία δοσοληψιών. Ενώ στην αρχιτεκτονική *Shared Nothing* ο έλεγχος ταυτόχρονης πρόσβασης είναι υπόθεση τοπική για κάθε κόμβο, στις data sharing αρχιτεκτονικές απαιτείται η ανταλλαγή μηνυμάτων μεταξύ των κόμβων προκειμένου να γίνει ο απαραίτητος συγχρονισμός. Οπότε για ένα σύστημα που υποστηρίζει υψηλό ρυθμό εξυπηρέτησης δοσοληψιών θα πρέπει να ελαχιστοποιηθεί ο αριθμός των μηνυμάτων που ανταλλάσσουν οι κόμβοι, ώστε να μειωθεί το κόστος επικοινωνίας και συνεπώς να βελτιωθεί απόδοση του συστήματος. Υψηλός αριθμός από μηνύματα οδηγεί σε μεγαλύτερους χρόνους απόκρισης οι οποίοι με την σειρά τους οδηγούν και σε μεγέθυνση του προβλήματος του data contention.

Προκειμένου να ελαχιστοποιηθεί ο αριθμός των μηνυμάτων τα πρωτόκολλα ταυτόχρονης πρόσβασης και συνέπειας καλό είναι να ενοποιηθούν σε ένα. Τα πρωτόκολλα ταυτόχρονης πρόσβασης με την καλύτερη απόδοση είναι αυτά που στηρίζονται στο κλείδωμα σελίδων της βάσης δεδομένων και αυτά που στηρίζονται στο λεγόμενο αισιόδοξο έλεγχος ταυτόχρονης πρόσβασης (optimistic concurrency control), είτε αυτοί οι έλεγχοι γίνονται κεντρικοποιημένα (centralized) είτε κατανεμημένα (distributed).

Ένα βασικό πρωτόκολλο centralized locking βασίζεται στον Central Lock Manager (CLM) [Rah93b], όπου είναι και το ποιο απλό πρωτόκολλο. Υπάρχει ένας κεντρικός lock manager όπου κρατάει το lock table όλης της βάσης και για όλους τους κόμβους. Κάθε lock και unlock αίτηση προωθείται στον CLM και αυτός εξετάζει (εφόσον έχει όλον τον lock table) αν είναι δυνατή η άμεση εξυπηρέτηση της αίτησης. Απαιτούνται δύο μηνύματα για κάθε lock και ένα στο τέλος για την ελευθέρωση όλων των locks. Σε αυτό το πρωτόκολλο μπορούν να γίνουν πολλές αλλαγές προκειμένου να βελτιωθεί η απόδοσή του, π.χ. μπορούν να ομαδοποιηθούν οι αιτήσεις και να μειωθούν έτσι τα μηνύματα που απαιτούνται και μπορεί να δοθεί η δυνατότητα χωρίς επικοινωνία με τον CLM να εξυπηρετούνται read locks.

Ένα άλλο πρωτόκολλο distributed locking είναι το Primary Copy Locking (PCL) [Rah86], όπου η βάση δεδομένων χωρίζεται σε λογικές διαμερίσεις και αναλαμβάνει ένας κόμβος την ευθύνη συγχρονισμού (primary copy authority) για το λογικό αυτό τμήμα της βάσης. Έτσι locks στο τμήμα της βάσης που έχει ευθύνη ο κόμβος δεν χρειάζονται επικοινωνία ενώ για τα lock στα άλλα τμήματα, οι αιτήσεις προωθούνται στον κατάλληλο κόμβο. Και σε αυτό το πρωτόκολλο έχουν προταθεί ορισμένες βελτιώσεις [Rah86], [Rah93b].

Γενικά στους μηχανισμούς που υιοθετούν το optimistic concurrency control μία δοσοληψία δεν κάνει abort ποτέ κατά την εκτέλεσή της παρά μόνο όταν στο τέλος γίνεται μια εξακρίβωση του κατά πόσο τα δεδομένα που προσπέλασε ήταν έγκυρα (up-to-date). Μόνο στη περίπτωση που τα δεδομένα δεν ήταν έγκυρα η δοσοληψία γίνεται abort. Σε αυτά τα πρωτόκολλα βασίζεται πολλές από της εργασίες του Rahm καθώς και η διδακτορική του διατριβή.

Παράδειγμα centralized optimistic concurrency control είναι το πρωτόκολλο Central Validation Scheme (CV-OCC) [Rah93b] όπου ένας κόμβος αναλαμβάνει να πιστοποιήσει την εγκυρότητα των δεδομένων (validation) τα οποία προσπέλασε μία δοσοληψία κατά την εκτέλεσή της. Με αυτόν τον τρόπο μόνο ένα μήνυμα απαιτείται. Συνήθως γίνεται χρήση χρονοσφραγίδων προκειμένου να αποφασιστεί η εγκυρότητα των αντιγράφων σελίδων που έχουν προσπελαστεί από μια δοσοληψία.

Εναλλακτικά, το πιο διαδεδομένο distributed optimistic concurrency control πρωτόκολλο είναι το Optimistic Token-Ring (TR-OCC) [Rah93b] όπου τα validations γίνονται από όλους τους κόμβους αλλά μόνο όταν αυτοί έχουν στην κατοχή τους το token. Το token είναι ένα μήνυμα ειδικό που αποστέλλεται σε όλους τους κόμβους με μία ορισμένη σειρά και στο τέλος επιστρέφει στον αρχικό του κάτοχο (ring). Έτσι με το πέρας μίας δοσοληψίας αυτή περιμένει το token για να μπορέσει να κάνει το local validation. Αφού γίνει αυτό και η δοσοληψία δεν γίνει abort, τότε μαζί με το token προστίθεται και η πληροφορία για το validation σε σχέση με τις δοσοληψίες των άλλων κόμβων. Όταν πάλι επιστρέψει το token μπορεί να αποφασιστεί κατά πόσο η δοσοληψία έχει τερματίσει επιτυχώς ή όχι.

Ο προσομοιωτής χρησιμοποιεί το two phase locking προκειμένου να γίνει μια προσπέλαση σε δεδομένα. Εάν αυτή η προσπέλαση έχει ως στόχο την μεταβολή τους τότε απαιτείται η αποκλειστική τους προσπέλαση, διαφορετικά η μη αποκλειστική προσπέλαση που επιτρέπει σε άλλες δοσοληψίες να προσπελάσουν ταυτόχρονα το ίδιο δεδομένο, αρκεί ο τρόπος προσπέλασης τους να είναι και πάλι μη αποκλειστικός. Με αυτόν τον τρόπο διασφαλίζεται η ιδιότητα της απομόνωσης.

Το two phase locking αποτελείται από δύο φάσεις, όπως φανερώνει και το όνομά του. Στην πρώτη φάση (φάση ανάπτυξης - growing phase) η δοσοληψία κάνει αιτήσεις για νέα lock χωρίς να αφήνει το lock από κανένα από τα δεδομένα στα οποία έχει το lock. Στην δεύτερη φάση (φάση συρρίκνωσης - shrinking phase) η δοσοληψία ελευθερώνει τα lock που έχει στην κατοχή της ενώ δεν αποκτά νέα.

Το κλείδωμα δεδομένων χρησιμοποιείται βασικά για την εγγύηση και της ιδιότητας της σειριοποίησης. Παρόλα αυτά υπάρχει η πιθανότητα δύο οι περισσότερες δοσοληψίες να εμπλακούν σε αδιέξοδο. Αυτό συμβαίνει όταν έχουν στην κατοχή τους δεδομένα κλειδωμένα αποκλειστικά και η μία επιθυμεί να κλειδώσει δεδομένα που έχει η άλλη κλειδωμένα. Επειδή καμιά δεν μπορεί να τερματίσει θα πρέπει κάποια από αυτές τις δοσοληψίες που εμπλέκονται στο αδιέξοδο να γίνει abort. Αυτή την δουλειά κάνει ο deadlock manager που έχει αναπτυχθεί.

Για την παρούσα εργασία επιλέχθηκε να γίνει υλοποίηση ενός κατανεμημένου πρωτοκόλλου locking, το οποίο βασίζεται στην ιδέα της επανάληψης του lock table σε όλους τους κόμβους της ομάδας που έχουν πρόσβαση στην κοινόχρηστη βάση δεδομένων. Αυτή η μέθοδος επιβάλλει την αποστολή ενός πολύ μεγάλου αριθμού από μηνύματα μεταξύ των κόμβων.

Συγκεκριμένα, για κάθε νέο lock ο κόμβος στέλνει μήνυμα σε κάθε κόμβο του cluster και μετά αναμένει για τις απαντήσεις τους. Όμοια για κάθε unlock ο ίδιος αριθμός από μηνύματα ανταλλάσσεται. Συνολικά δηλαδή απαιτείται η ανταλλαγή από $4 * (N - 1)$ μηνύματα (N ο αριθμός των κόμβων του cluster), ώστε κάθε κόμβος να κρατάει αντίγραφο του lock table. Η

αύξηση του αριθμού των μηνυμάτων είναι γραμμική σε σχέση με τον αριθμό των κόμβων, κάτι που κάνει το συγκεκριμένο πρωτόκολλο μη αποδεκτό για συστήματα με μεγάλο αριθμό από κόμβους.

Μπορεί να είναι το πιο απλό πρωτόκολλο και αυτό με το μεγαλύτερο κόστος όμως απαιτεί τον μέγιστο βαθμό συγχρονισμού από τον προγραμματιστή του, προκειμένου να προβλεφθούν όλα τα πιθανά σενάρια άφιξης αιτήσεων στους κόμβους για ένα συγκεκριμένο αντικείμενο της βάσης δεδομένων. Συμβαίνει συχνά δύο κόμβοι να ξεκινούν ταυτόχρονα να κλειδώσουν το ίδιο δεδομένο οπότε πρέπει με κάποιο τρόπο όλοι οι κόμβοι να στείλουν την απάντηση μόνο στον ένα τον κόμβο ενώ ο άλλος θα πρέπει να τοποθετήσει την αίτησή του στην λίστα αναμονής. Αυτό δεν είναι δυνατό να συμβεί στα παραπάνω πρωτόκολλα διότι στην περίπτωση κεντρικοποιημένου ελέγχου, ο πρώτος που θα φτάσει θα αποκτήσει το lock και ο άλλος θα περιμένει ενώ για την περίπτωση του PCL ή TR-OCC ελέγχου, ένας κόμβος θα πρέπει να απευθυνθεί σε έναν συγκεκριμένο κόμβο του συστήματος ή αντίστοιχα να περιμένει το token.

Όταν ένας κόμβος θέλει να κλειδώσει μία σελίδα μη αποκλειστικά και την οποία ένας άλλος κόμβος κρατάει το lock και πάλι μη αποκλειστικά, τότε ζητάει μέσω ειδικού μηνύματος από τον κόμβο ιδιοκτήτη της σελίδας να κλειδώσει και αυτός το ίδιο δεδομένο. Σε περίπτωση που ένας από τους δύο κόμβους ζητά μη αποκλειστική προσπέλαση τότε η αίτηση αυτή θα μπει στην λίστα αναμονής του lock table entry (που βρίσκεται σε κάθε κόμβο ξεχωριστά) και κατά την φάση του unlock ο κόμβος ανταγωνίζεται με τους άλλους προκειμένου να γίνει ο νέος ιδιοκτήτης της σελίδας. Συνοπτικά η διαδικασία locking και unlocking ενός δεδομένου εμφανίζεται στις επόμενες δύο διαδικασίες:

Διαδικασία 1 *Διαδικασία locking δεδομένου*

Βήμα 1: Έλεγξε τον τοπικό lock table.
Βήμα 2: Αν το δεδομένο είναι κλειδωμένο από κάποιον κόμβο τότε :
- Αν είναι κλειδωμένο για πράξη ανάγνωσης και η νέα αίτηση είναι και αυτή για πράξη ανάγνωσης τότε στείλε την αίτηση στον κόμβο - ιδιοκτήτη του δεδομένου.

- Αν οποιαδήποτε από τις δύο πράξεις είναι για αποκλειστική πράξη τότε πρόσθεσε την αίτηση στη λίστα αναμονής για το συγκεκριμένο δεδομένο και περίμενε να ξεκλειδωθεί το δεδομένο.

end
Αλλιώς:
Βήμα 3: Στείλε σε κάθε κόμβο μέλος του cluster μήνυμα για την απόκτηση της ιδιοκτησίας του δεδομένου.
Βήμα 4: Περίμενε να απαντήσουν όλοι οι κόμβοι.
Βήμα 5: Αν όλοι οι κόμβοι απαντήσουν θετικά τότε είναι ο ιδιοκτήτης του δεδομένου. Αλλιώς ανάλογα με το επιστραφέν μήνυμα πρόσθεσε την αίτηση στη λίστα αναμονής για το συγκεκριμένο δεδομένο ή ικανοποίησε την αίτηση για lock. Αρνητική απάντηση μπορεί να πάρει ένας κόμβος όταν κάποιος άλλος κόμβος έχει στον τοπικό του lock table καταχώρηση για αυτό το δεδομένο.

end
end

end

Διαδικασία 2 *Διαδικασία unlocking δεδομένου*

Βήμα 1:] Στείλε σε κάθε κόμβο μέλος του cluster μήνυμα για το unlock του δεδομένου.
Επανέλαβε
Βήμα 2: Περίμενε για απάντηση. Εάν η απάντηση περιλαμβάνει πληροφορία για πρόθεση νέας ιδιοκτησίας εκ μέρους του κόμβου αποστολέα τότε, αν είναι ο πρώτος που το ζητά, γίνεται αυτόματα ο επόμενος ιδιοκτήτης αλλιώς πληροφορείται για το ποιός κόμβος θα είναι ο επόμενος ιδιοκτήτης του δεδομένου.
end
Έως ότου απαντήσουν όλοι οι κόμβοι.
end

end

Επειδή οι πράξεις lock και unlock δεδομένων γίνονται με μεγάλη συχνότητα και επειδή η επεξεργασία που γίνεται ενδιάμεσα του lock και unlock message συχνά είναι πολύ μικρή, θα πρέπει η όλη διαδικασία να μπορεί να γίνεται αρκετά γρήγορα. Έτσι στην παρούσα υλοποίηση (όπως άλλωστε συνηθίζεται και στα περισσότερα συστήματα) έχει γίνει η υλοποίηση του lock table ως πίνακα κατακερματισμού (hash table) με κλειδί το data item. Αυτό εξασφαλίζει ότι η αναζήτηση ενός lock δεν θα είναι χρονοβόρα.

Τα δεδομένα κλειδώνονται σε επίπεδο σελίδας η οποία είναι και η μονάδα μεταφοράς μεταξύ κόμβων και βάσης δεδομένων. Το λεγόμενο και ως locking granularity είναι πολύ σημαντικό για την απόδοση ενός συστήματος. Όσο πιο μικρό είναι το μέγεθος της μονάδας που κλειδώνεται τόσο πιο πολύ μειώνεται το data contention αλλά απαιτείται μεγαλύτερος αριθμός από locks με αποτέλεσμα να αυξάνεται η επιπλέον επιβάρυνση (overhead) αλλά και το μέγεθος του lock table. Η επιλογή μεγάλου locking granularity μειώνει την επικοινωνία και το μέγεθος του lock table αλλά αυξάνει το data contention πρόβλημα. Η επιλογή του σε ένα σύστημα είναι θα λέγαμε μία δύσκολη υπόθεση που εξαρτάται από αρκετούς παράγοντες όπως είναι ο βαθμός πολυπρογραμματισμού, το πρωτόκολλο συνδρομικότητας που χρησιμοποιείται, το είδος των προσπελάσεων και η πιθανότητα που αυτές μεταβάλλουν τα αντικείμενα της βάσης δεδομένων κ.τ.λ.

Παράλληλα με τον έλεγχο συνδρομικότητας γίνεται και ο έλεγχος συνέπειας. Ο έλεγχος συνέπειας είναι επιτακτικός λόγω της δυνατότητας που δίνεται στους κόμβους να κρατάνε σελίδες της βάσης στους τοπικούς ενταμιευτές που αποτελούν μέρος της κύριας μνήμης κάθε κόμβου. Η αποθήκευση σελίδων στην κύρια μνήμη μπορεί να μειώσει τον αριθμό από τις προσπελάσεις στους αργούς δίσκους του συστήματος, ειδικά στην περίπτωση όπου υπάρχει

τοπικότητα στις προσπελάσεις των δοσοληψιών. Δυστυχώς στην περίπτωση των data sharing αρχιτεκτονικών υπάρχει το πρόβλημα της ακύρωσης των σελίδων (buffer invalidation) που βρίσκονται αποθηκευμένες στους τοπικούς ενταμιευτές λόγω της δυνατότητας μίας σελίδας να βρίσκεται σε παραπάνω από έναν ενταμιευτή συγχρόνως. Η μεταβολή μίας σελίδας σε έναν από αυτούς τους κόμβους από μία δοσοληψία κάνει όλα τα αντίγραφα, μαζί και αυτό στον δίσκο, ασυνεπή. Βασικό λοιπόν μέλημα του ελέγχου συνδρομικότητας είναι να εγγυάται ότι οι δοσοληψίες βλέπουν πάντα δεδομένα της βάσης που είναι συνεπή.

Η λύση του προβλήματος του buffer invalidation στην αρχιτεκτονική των κοινόχρηστων δεδομένων έχει άμεση σχέση με το μέγεθος (granularity) των δεδομένων για τα οποία γίνεται ο έλεγχος συνδρομικότητας και η στρατηγική για τον τρόπο που οι αλλαγές αυτές μεταφέρονται στην μόνιμη μνήμη. Η μόνιμη μνήμη μπορεί να είναι ο δίσκος ή μία μη πτητική μνήμη.

Ο έλεγχος της συνδρομικότητας σε επίπεδο record μίας σελίδας μπορεί να βελτιώνει το φαινόμενο του data contention όμως επιβάλλει κάποιους περιορισμούς. Έτσι θα πρέπει να γίνουν οι συγχωνεύσεις των αλλαγών που έχουν στα record μιας σελίδας από όλους τους κόμβους, όταν αυτή θα μεταφερθεί σε μόνιμη μνήμη [MN91] ώστε η νέα σελίδα στην περιφεριακή μνήμη του συστήματος να ενσωματώνει όλες τις αλλαγές που έχουν γίνει. Αυτό μπορεί να απαιτεί έναν μεγάλο αριθμό από μηνύματα που θα πρέπει να ανταλλαγούν μεταξύ των κόμβων. Παράλληλα, σε περιπτώσεις όπου αλλάζει η δομή της σελίδας (π.χ. το μέγεθος του νέου record μπορεί να απαιτεί μεγαλύτερο μέγεθος σελίδας) τα πράγματα περιπλέκονται και χρειάζονται λύσεις που πιθανώς είναι αρκετά χρονοβόρες. Γι' αυτό συνήθως τα συστήματα δεν υποστηρίζουν έλεγχο συνδρομικότητας σε επίπεδο record ή το υποστηρίζουν μόνο στις περιπτώσεις όπου δεν αλλάζει η δομή της σελίδας [Rah91a], [MNS91].

Δύο είναι οι στρατηγικές για την προώθηση των αλλαγών στην μόνιμη μνήμη του συστήματος: η FORCE όπου όλες οι σελίδες που αλλάζουν από μία δοσοληψία μεταφέρονται αναγκαστικά στην μόνιμη μνήμη πριν το commit της δοσοληψίας. Αυτή η στρατηγική είναι μη αποδεκτή για συστήματα που πρέπει να έχουν υψηλό ρυθμό εξυπηρέτησης διότι οδηγεί σε αυξημένο ρυθμό από πράξεις I/O. Παρόλα αυτά διευκολύνει τον έλεγχο συνέπειας διότι η πιο πρόσφατη σελίδα βρίσκεται στην μόνιμη μνήμη. Δεν υπάρχει λόγος για redo recovery μετά από την βλάβη ενός κόμβου. Παρόλα τα προβλήματα που έχει η στρατηγική αυτή, έχει υιοθετηθεί από τα περισσότερα συστήματα και μάλιστα στις περιπτώσεις που συνδυάζεται με την ύπαρξη μίας μη πτητικής μνήμης το κέρδος είναι σαφώς μεγαλύτερο, διότι αποφεύγεται η διαρκής πρόσβαση στους δίσκους [Rah91b], [BHR90], [YD94b], [YD94a].

Η NO-FORCE στρατηγική μειώνει την επιβάρυνση για I/O αλλά πρέπει να κρατάει αρκετή πληροφορία στο αρχείο μεταβολών ώστε να είναι σε θέση να κάνει redo recovery. Παράλληλα θα πρέπει για τον έλεγχο συνέπειας να είναι σε θέση να κρατάει με κάποιο τρόπο πληροφορία για την τοποθεσία (ενταμιευτή και κόμβου) που υπάρχει η έγκυρη σελίδα. Θα πρέπει δηλαδή να υποστηρίζεται η συνεργασία των ενταμιευτών προκειμένου να μεταφέρουν την σελίδα από τον κόμβο που έχει την έγκυρη σελίδα.

Επίσης είναι δυνατό μία σελίδα να μεταφερθεί στη μόνιμη μνήμη όταν ο αλγόριθμος αντικατάστασης το αποφασίσει ανεξάρτητα εάν η δοσοληψία που έχει κάνει την προσπέλαση σε αυτήν την σελίδα είναι ενεργή ή όχι. Αυτός ο κανόνας είναι ο κανόνας STEAL, σε

αντίθεση με τον κανόνα NO-STEAL όπου δεν είναι δυνατό να μεταφερθούν οι σελίδες μίας δοσοληψίας στην μόνιμη μνήμη όσο αυτή είναι ενεργή. Αξιοσημείωτο είναι ότι η FORCE στρατηγική ουσιαστικά προϋποθέτει το κανόνα STEAL εφόσον οι σελίδες που έχουν μεταβληθεί μεταφέρονται πριν το commit στη μόνιμη μνήμη. Ο κανόνας STEAL συνεπάγεται ότι για την ακύρωση των αλλαγών που επέφερε μία δοσοληψία χρειάζεται ο μηχανισμός καταγραφής μεταβολών να κρατά αρκετή πληροφορία ώστε να είναι δυνατή η ακύρωση των αλλαγών (undo) σε κάθε σελίδα που προσπέλασε η δοσοληψία. Ενδεχομένως αυτό μπορεί να απαιτεί και προσπελάσεις στην περιφεριακή μνήμη.

Μια ανασκόπηση των μεθόδων ελέγχου της συνέπειας μπορεί να βρεθεί στις εργασίες [DY93] και [Rah93b]. Γενικά το buffer invalidation μπορεί είτε να εντοπιστεί είτε να αποφευχθεί. Ένα απλό πρωτόκολλο αλλά με μεγάλο κόστος είναι η αποστολή μηνύματος σε όλους τους κόμβους ώστε να γίνει το invalidation τυχών εμφανίσεων της σελίδας στους ενταμιευτές των κόμβων (broadcast invalidation).

Ένας άλλος τρόπος που αποφεύγει την αποστολή μηνυμάτων ορίζει πως σε κάθε προσπέλαση σε δεδομένα που βρίσκονται στον ενταμιευτή, θα πρέπει να ελέγχεται και η εγκυρότητά τους (on request invalidation). Τέλος, ένας άλλος τρόπος είναι να ακυρώνονται οι σελίδες σε έναν κόμβο πριν γίνει η μεταβολή τους σε έναν άλλο κόμβο.

Στην παρούσα υλοποίηση, όταν ένας κόμβος ζητά από τους άλλους κόμβους την άδεια να κλειδώσει μία σελίδα αποκλειστικά (write ή update πράξη) τότε οι άλλοι κόμβοι, πριν του στείλουν την απάντηση, ακυρώνουν τυχόν εμφάνιση της σελίδας στον δικό τους ενταμιευτή. Δεν απαιτείται δηλαδή επιπλέον μήνυμα. Με αυτόν τον τρόπο όμως είναι δυνατό να ακυρωθούν ορισμένες σελίδες που τελικά δεν θα έπρεπε (διότι η δοσοληψία που το ζήτησε μπορεί να έκανε abort). Πάντως αυτή η πιθανότητα είναι πολύ μικρή. Παράλληλα ο προσομοιωτής υλοποιεί την FORCE στρατηγική με την χρήση του κανόνα STEAL για τις σελίδες που βρίσκονται στον ενταμιευτή. Έτσι λοιπόν, κατά την επανεκκίνηση του συστήματος μετά από βλάβη, δεν χρειάζεται να γίνουν redo πράξεις παρά μόνο undo. Ο προσομοιωτής δεν υποστηρίζει την βλάβη κόμβων ή οποιουδήποτε άλλου μέρους του συστήματος. Έτσι κάθε κόμβος κρατάει το δικό του αρχείο μεταβολών υλοποιώντας το πρωτόκολλο write-ahead-logging, όπου κάθε δοσοληψία που επιθυμεί να μεταβάλει ένα αντικείμενο της βάσης θα πρέπει πρώτα να εισάγει στο αρχείο μεταβολών μία εγγραφή με αρκετή πληροφορία ώστε να είναι δυνατή η επαναφορά σε περίπτωση βλάβης του συστήματος ή abort της δοσοληψίας.

Κάθε εγγραφή έχει ένα μοναδικό αριθμό LSN (Log Sequence Number) ενώ κάθε σελίδα έχει το LSN της πιο πρόσφατης εγγραφής. Βασική ιδιότητα του αναγνωριστικού αυτού αριθμού είναι η μονότονη αύξησή του. Δηλαδή μία εγγραφή της ίδιας σελίδας που έπεται χρονικά θα έχει και μεγαλύτερο LSN. Κατά την φάση της επανεκκίνησης, μόνο οι εγγραφές του αρχείου μεταβολών με LSN μεγαλύτερο από αυτόν της σελίδας θα πρέπει να ληφθούν υπόψη (για κάθε σελίδα). Προκειμένου να γίνει η επανεκκίνηση του συστήματος μετά από βλάβη ενός κόμβου απαιτείται η κατασκευή ενός αρχείου μεταβολών που θα περιλαμβάνει τις πληροφορίες όλων των επιμέρους αρχείων μεταβολών που κρατά κάθε κόμβος ξεχωριστά. Αυτή η εργασία γίνεται συνήθως από αλγόριθμους που τρέχουν off-line. Αναφορές στο θέμα της επανεκκίνησης μετά από βλάβη σε συστήματα κοινόχρηστων δεδομένων μπορεί κανείς

να βρει στις αναφορές [MN91], [MN92], [Rah91a], [BHG87], [BG81].

Κεφάλαιο 5

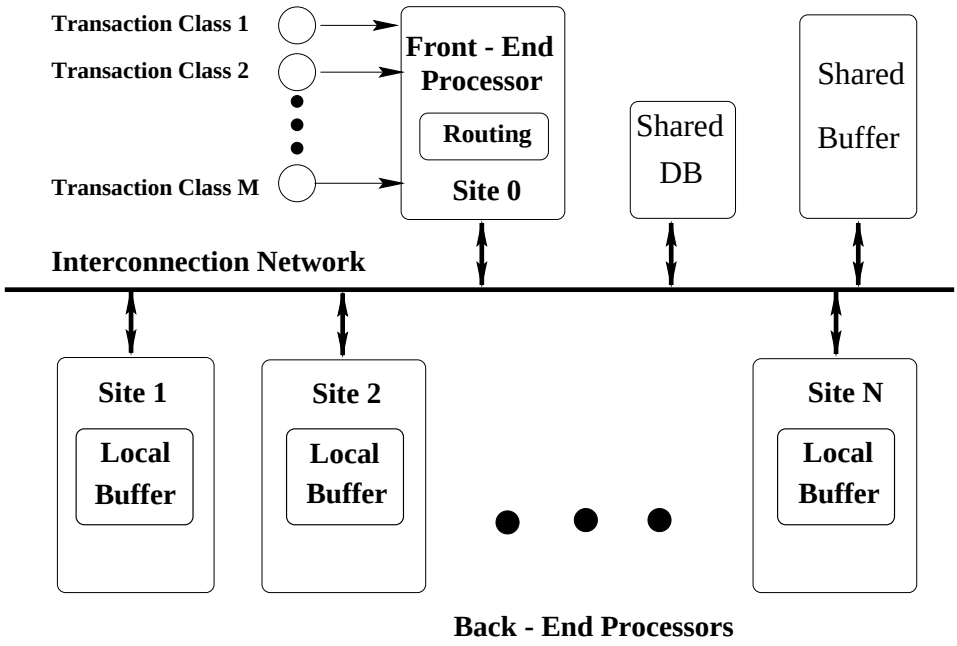
Δρομολόγηση δοσοληψιών σε αρχιτεκτονικές κοινόχρηστων δεδομένων

Στο κεφάλαιο αυτό θα δούμε αναλυτικά έναν νέο αλγόριθμο για δρομολόγηση δοσοληψιών κατάλληλο για αρχιτεκτονικές που έχουν δεδομένα αποθηκευμένα σε κοινόχρηστους δίσκους όπως οι αρχιτεκτονικές *Shared Disk* και *Shared Intermediate Memory* που υλοποιήθηκαν στον προσομοιωτή, στα πλαίσια της παρούσας μεταπτυχιακής εργασίας. Μια περιγραφή του αλγορίθμου αυτού βρίσκεται στην αναφορά [FP95].

Αρχικά θα δοθεί μία λεπτομερής περιγραφή του αλγορίθμου δρομολόγησης και στη συνέχεια θα αξιολογηθεί η επίδοσή του σε συγκεκριμένα μοντέλα συστημάτων με την χρήση των προαναφερομένων αρχιτεκτονικών του TPsim. Η δρομολόγηση σε συστήματα με κοινόχρηστους δίσκους είναι ένας τομέας όπου επικρατεί ένα είδος μυστικότητας καθώς δεν υπάρχει στην διεθνή βιβλιογραφία περιγραφή αλγόριθμων για τέτοιες αρχιτεκτονικές. Οι αλγόριθμοι που υπάρχουν αποτελούν εσωτερικά μυστικά των μεγάλων εταιρειών που ειδικεύονται στην παροχή λογισμικού για τα συστήματα επεξεργασίας δοσοληψιών και ιδιαίτερα για τις προαναφερόμενες αρχιτεκτονικές οι εταιρίες αυτές δεν έχουν δημοσιεύσει τους αλγόριθμους δρομολόγησης που χρησιμοποιούν στα προϊόντα τους.

5.1 Το πρόβλημα

Έχουμε μία κοινόχρηστη βάση δεδομένων υπό την έννοια ότι μία ομάδα από κόμβους έχει άμεση πρόσβαση στα δεδομένα που είναι αποθηκευμένα στην περιφεριακή μνήμη. Πιθανώς υπάρχει και ένα ενδιάμεσο επίπεδο στην ιεραρχία της μνήμης που στην προκειμένη περίπτωση είναι ένας κοινόχρηστος ενταμιευτής. Έχουμε ένα σύστημα όπως αυτό που φαίνεται στο σχήμα 5.1 όπου υπάρχουν N + 1 επεξεργαστές $(F_0, B_1, B_2, ..., B_N)$. Ο επεξεργαστής F_0 είναι ένας επεξεργαστής Front-End και είναι υπεύθυνος για την δρομολόγηση των δοσοληψιών σε έναν από τους υπόλοιπους N κόμβους του συστήματος



Σχήμα 5.1: Παράδειγμα συστήματος με έναν κόμβο Front-End και N κόμβους Back-End

(Back-End processors) οι οποίοι είναι αυτοί που θα εξυπηρετήσουν τις δοσοληψίες.

Υπάρχουν M κλάσεις δοσοληψιών $(C_1, C_2, ..., C_M)$ και κάθε δοσοληψία ανήκει σε μία από αυτές τις κλάσεις. Για κάθε μία κλάση δοσοληψιών ο διαχειριστής του συστήματος έχει ορίσει κάποιους στόχους επίδοσης g_i για $i = 1, ..., M$ που αφορούν τον μέγιστο επιθυμητό χρόνο απόκρισης για κάθε κλάση. Σκοπός και πρόβλημα μαζί είναι το πού θα πρέπει να δρομολογήσουμε τις δοσοληψίες που φτάνουν στον Front-End κόμβο ώστε να το σύστημα να ικανοποιεί τους στόχους που έχουν τεθεί.

5.2 Ο αλγόριθμος δρομολόγησης

Προκειμένου να γίνει αυτό ο αλγόριθμος δρομολόγησης υπολογίζει έναν πίνακα $Nopt(i, j)$ για κάθε κλάση και κόμβο του συστήματος, που δηλώνει τον μέγιστο αριθμό από τις δοσοληψίες της κλάσης i οι οποίες θα πρέπει να είναι ενεργές στον κόμβο j ώστε το σύστημα να βρίσκεται σε σταθερή κατάσταση. Αν υπάρχουν $Nopt(i, j)$ δοσοληψίες της κλάσης i στον κόμβο j τότε για την κλάση i ο χρόνος απόκρισης είναι πολύ κοντά στους στόχους που έχουν τεθεί και λέμε τότε ότι το σύστημα βρίσκεται σε σταθερή κατάσταση. Ο υπολογισμός του πίνακα αυτού απαιτεί την εκτίμηση του μέσου χρόνου απόκρισης για κάθε κλάση η οποία γίνεται βάση ενός πίνακα δρομολόγησης. Διαδοχικά μεταβάλλουμε τον πίνακα δρομολόγησης εκτιμούμε τους νέους μέσους χρόνους απόκρισης για τις κλάσεις του συστήματος και στη συνέχεια αποφασίζουμε για το εάν ο νέος τρόπος δρομολόγησης είναι καλύτερος από αυτόν του προηγούμενου βήματος του αλγορίθμου. Στην επόμενη παράγραφο φαίνεται ο τρόπος εκτίμησης του μέσου χρόνου απόκρισης για κάθε κλάση δοσοληψιών.

Ο υπολογισμός του $Nopt(i, j)$ βασίζεται σε έναν αριθμό από στατιστικές παραμέτρους

όπως: ο ρυθμός άφιξης κάθε κλάσης, οι ανάγκες επεξεργασίας για κάθε μία από αυτές, ο τρόπος προσπέλασης στη βάση (access pattern), η πιθανότητα αυτή η προσπέλαση να είναι για μεταβολή των δεδομένων, ο ρυθμός άφιξης για κάθε κλάση κ.α.

Έχοντας υπολογιστεί αυτός ο πίνακας, ο αλγόριθμος δρομολόγησης είναι πολύ απλός και βασίζεται σε δύο πολύ απλές διαδικασίες. Ο κόμβος δρομολογητής κρατάει έναν πίνακα $N(i,j)$ που δηλώνει τον αριθμό από της δοσοληψίες της κλάσης i που είναι ενεργές στον back-end κόμβο j. Η μία είναι η διαδικασία που εφαρμόζεται κατά την άφιξη μίας δοσοληψίας και η άλλη είναι η διαδικασία που ενεργοποιείται κατά τον τερματισμό μίας δοσοληψίας.

Διαδικασία 3 *Διαδικασία άφιξης δοσοληψίας κλάσης i*

∗ Για κάθε Back End Processor (BEP) υπολόγισε $(N(i,j)+1)/Nopt(i,j)$

∗ Για τον κόμβο που ο λόγος αυτός είναι ελάχιστος (έστω k) αύξησε το $N(i,k)$ κατά 1.

∗ Δρομολόγησε την δοσοληψία της κλάσης i στον κόμβο k.

end

end

Διαδικασία 4 *Διαδικασία αναχώρησης δοσοληψίας κλάσης i από τον κόμβο BEP j*

∗ Μείωσε κατά 1 το $N(i,j)$.

end

end

Δηλαδή το βασικό στην όλη διαδικασία είναι ο υπολογισμός του πίνακας Nopt(i,j). Ο υπολογισμός αυτός χρειάζεται να επαναληφθεί μόνο κατά την περίπτωση όπου αλλάζουν οι στατιστικές παράμετροι του προβλήματος. Έτσι αν και η πολυπλοκότητά του είναι μεγάλη αυτή δεν επηρεάζει καθόλου την απόδοση του συστήματος γιατί δεν γίνεται ο υπολογισμός κατά την διάρκεια της δρομολόγησης.

5.2.1 Υπολογισμός του εκτιμώμενου μέσου χρόνου απόκρισης R

Ο τρόπος υπολογισμού του εκτιμώμενου μέσου χρόνου απόκρισης μίας δοσοληψίας κλάσης i στον κόμβο $j(R(i,j))$ γίνεται με την εφαρμογή της εξίσωσης 5.1 η οποία αποτελείται από δύο σκέλη.

$$R(i,j) := \begin{cases} \frac{S(i,j)}{1.0-Util(j)} + MDasd(i,j) * Delay_IO & ,Util \leq 0.98 \\ S(i,j) * [c1 + c2 * Util(j) + c3 * Util(j)^2] + MDasd(i,j) * Delay_IO & ,\text{αλλιώς} \end{cases} \tag{5.1}$$

Ο όρος $Util(j)$ εκφράζει τον βαθμό χρήσης του κόμβου j, ο $MDasd(i,j)$ όρος εκφράζει τον αριθμό των σελίδων που δεν βρίσκονται στον τοπικό ή στον κοινόχρηστο ενταμιευτή, για μία δοσοληψία κλάσης i που εκτελείται στον κόμβο j. Για αυτόν τον αριθμό από σελίδες θα πρέπει να γίνει προσπέλαση στην περιφεριακή μνήμη του συστήματος. Ο όρος Delay_IO εκφράζει την χρονική καθυστέρηση που επιβάλλει κάθε πρόσβαση στην περιφεριακή μνήμη του συστήματος και είναι κοινή για όλους τους κόμβους και κλάσεις καθώς εξαρτάται από τα χαρακτηριστικά του αποθηκευτικού μέσου που χρησιμοποιείται. Καθορίζεται από τα χαρακτηριστικά της ίδιας της συσκευής και μετράτε κατά την διάρκεια του monitor run. Τέλος ο όρος $S(i,j)$ εκφράζει το μέσο χρονικό κόστος επεξεργασίας για μία δοσοληψία κλάσης i που εκτελείται στον κόμβο j. Ο υπολογισμός αυτών των όρων ακολουθεί στις επόμενες παραγράφους.

Ο λόγος που γίνεται η διάκριση στην εξίσωση 5.1 σε δύο σκέλη, είναι ότι κατά την διάρκεια υπολογισμού του πίνακα πιθανοτήτων, μπορεί να συμβεί ο βαθμός χρήσης ενός κόμβου να είναι μεγαλύτερος από 1 ή πολύ κοντά στο 1 με αποτέλεσμα ο παρονομαστής να τείνει στο μηδέν οπότε ο πρώτος όρος τείνει στο άπειρο. Για αυτό το λόγο προσεγγίζουμε την συνάρτηση $\frac{1}{1-x}$ με την συνάρτηση $c1 + c2 * x + c3 * x^2$ για x>ρ όπου ρ μπορούμε να θέσουμε μία ποσότητα πολύ κοντά στο 1 αλλά πιο μικρή (χρησιμοποιήσαμε το 0.98). Η προσέγγιση της αρχικής συνάρτησης βασίζεται στην απαίτηση ότι η τιμή και οι δύο πρώτες παράγωγοι θα πρέπει να είναι για $x = \rho$, ίδιες.

Οι παράμετροι c1, c2, c3 δίνονται από τους τύπους :

$$c1 := (1.0 - 3.0 * \rho - 3.0 * \rho^2) * (1.0 - \rho)^{-3} \tag{5.2}$$

$$c2 := (1.0 - 3.0 * \rho) * (1.0 - \rho)^{-3} \tag{5.3}$$

$$c3 := (1.0 - \rho)^{-3} \tag{5.4}$$

5.3 Υπολογισμός του πίνακα Nopt

Πριν παρουσιάσουμε τον αλγόριθμο που υπολογίζει τον πίνακα $Nopt$ θα πρέπει να δώσουμε μία σειρά από συμβολισμούς πινάκων και την ερμηνεία τους. Καταρχήν θα πρέπει να πούμε ότι το i θα χρησιμοποιείται για τον συμβολισμό του δείκτη μίας κλάσης δοσοληψιών, το j για το συμβολισμό του δείκτη ενός κόμβου και το k για τον συμβολισμό του δείκτη ενός αρχείου. Έτσι:

- $p(i,j)$: εκφράζει την πιθανότητα της δρομολόγησης μίας δοσοληψίας κλάσεως i στον κόμβο j.

- $Arrival_Rate(i)$: είναι ο πίνακας που δηλώνει τον ρυθμό άφιξης των διαφόρων κλάσεων δοσοληψιών.
- $R(i,j)$: είναι ο εκτιμώμενος μέσος χρόνος απόκρισης μίας δοσοληψίας της κλάσης i στον κόμβο j, όταν οι δοσοληψίες δρομολογούνται βάσει ενός πίνακα πιθανοτήτων δρομολόγησης.
- $DerivR(i,j,i_0,j_0)$: η παράγωγος του $R(i_0,j_0)$ ως προς το $p(i.j)$. Η παραγώγιση έχει γίνει στο μεγαλύτερός της μέρος αναλυτικά με την χρήση της εξίσωσης 5.1 πλην του σημείου όπου υπολογίζεται η παράγωγος του πίνακα $MDasd$ όπου αυτή γίνεται εμπειρικά υπολογίζοντας την τιμή για το $p(i,j)$ και το $p(i,j)+$ δ και διαιρώντας δια του δ, όπου δ μία ποσότητα θετική αλλά πολύ μικρή.
- $AvgR(i)$: εκφράζει τον εκτιμώμενο μέσο χρόνος απόκρισης μίας κλάσης δοσοληψιών i στον σύστημα, όταν αυτό βρίσκεται σε σταθερή κατάσταση.
- $PerfIndex(i)$: εκφράζει τον εκτιμώμενο δείκτη επίδοσης για κάθε κλάση δοσοληψιών.
- $g(i)$: εκφράζει τους στόχους επίδοσης που έχουν τεθεί για κάθε κλάση δοσοληψιών.

Ο πίνακας $AvgR$ υπολογίζεται βάση του πίνακα p και του πίνακα R από την εξίσωση 5.5, ενώ το διάνυσμα με τους δείκτες επίδοσης από την εξίσωση 5.6.

$$AvgR(i) := \sum_{j=1}^{N} p(i,j) * R(i,j) \tag{5.5}$$

$$PerfIndex := (\frac{AvgR(1)}{g(1)}, ..., \frac{AvgR(M)}{g(M)}) \tag{5.6}$$

Η λογική του αλγορίθμου είναι απλή. Ξεκινώντας από έναν αρχικό πίνακα δρομολόγησης όπου όλες οι κλάσεις δρομολογούνται στον κόμβο 1 με πιθανότητα 1.0 προσπαθούμε να υπολογίσουμε έναν νέο πίνακα δρομολόγησης για τον οποίο το διάνυσμα των δεικτών επίδοσης για τις κλάσεις δοσοληψιών του συστήματος να είναι μικρότερο. Στην παρούσα υλοποίηση κρίθηκε ως πιο κατάλληλη επιλογή του τρόπου σύγκρισης δύο διανυσμάτων, η σύγκριση του αθροίσματος των τετραγώνων των τιμών κάθε συνιστώσας του διανύσματος. Αυτό με την πιο μικρή τιμή θεωρείται και ως πιο μικρό διάνυσμα.

Προκειμένου να πάρουμε ένα νέο πίνακα δρομολόγησης εφαρμόζουμε την μέθοδο προβολής της παραγώγου για προβλήματα βελτιστοποίησης σε κυρτά σύνολα (standard gradient projection algorithm for optimization problems over convex sets) [DZ88]. Ξεκινώντας από την κλάση με τη χειρότερη επίδοση στο διάνυσμα των δεικτών επίδοσης εφαρμόζουμε την ιδέα του παραπάνω αλγόριθμου. Αρχικά λαμβάνουμε έναν πίνακα $temp(i,j)$ και υπολογίζουμε τον νέο πίνακα δρομολόγησης από την εξίσωση 5.7. Ο πίνακας $temp(i,j)$ προκύπτει από την διαδοχική εφαρμογή των εξισώσεων 5.8, 5.9, 5.10. Στον αλγόριθμο που ακολουθεί οι συναρτήσεις computeF(), computeLow(), και computeNewProbabilityMatrix() αναλαμβάνουν να υπολογίσουν αυτές τις εξισώσεις καθώς και την 5.7.

$$q(i,j) := (1 - mult2) * p(i,j) + mult2 * temp(i,j) \tag{5.7}$$

$$F(i,j) = \begin{cases} p(high2,j) * derivR(i,j,high2,j) & ,i \neq high2 \\ p(high2,j) * derivR(i,j,high2,j) + R(high2,j) & ,\text{αλλιώς} \end{cases} \tag{5.8}$$

$$low(i) := argmin_{(1 \leq j \leq N)} F(i,j) \tag{5.9}$$

$$temp(i,j) = \begin{cases} max\{0, p(i,j) - mult1 * [F(i,j) - F(i,low(i))]\} & ,j \neq low(i) \\ 1.0 - \sum_{j \neq low(i)} p(i,j) & ,\text{αλλιώς} \end{cases} \tag{5.10}$$

Ο $high2$ δείκτης είναι ο δείκτης της κλάσης για την οποία παραγωγίζουμε σε κάθε βήμα του αλγορίθμου. Αρχικά είναι η κλάση με την χειρότερη επίδοση, στη συνέχεια αυτή με τον δεύτερο χειρότερο δείκτη κ.ο.κ. Αρχικά το $mult2$ είναι ίσο με 0.8 και ρυθμίζει το πόσο θα επηρεαστεί ο πίνακας q από τον πίνακα $temp$. Υπολογίζεται στη συνέχεια το νέο διάνυσμα με τους δείκτες επίδοσης και συγκρίνεται με το παλιό. Αν δεν είναι πιο μικρό τότε θέτουμε $mult2 := \frac{mult2}{2}$, αλλιώς ο νέος πίνακας γίνεται αποδεκτός. Εάν δεν βρούμε πίνακα που να δίνει πιο μικρό διάνυσμα με τους δείκτες επίδοσης προχωράμε στο αμέσως επόμενο μεγαλύτερο δείκτη του διανύσματος.

Ας δούμε πρώτα όμως τον αλγόριθμο μαζί με ορισμένες από τις λεπτομέριές του. Αναλυτικά ο αλγόριθμος βρίσκεται στο παράρτημα C.

Διαδικασία 5 *Υπολογισμός του πίνακα* $Nopt(i,j)$

```
{high1} ← 1; {mult1} = {mult2} ← 0.8;
WHILE (high1 <= NumClasses)
    PerfIndex = computePerformanceIndex(p); {exit} ← FALSE;
    high2 = Findhigh1_th_component();
    F = computeF();
    Low = computeLow();
    WHILE (exit == FALSE)
        q = computeNewProbabilityMatrix();
        NewPerfIndex = computePerformanceIndex(q);
        comparison = comparePerfIndexes(PerfIndex, NewPerfIndex);
        IF (comparison == 2)
            {p} ← {q}; {mult2} ← 0.8;
        end
        ELSE
            IF (mult2 > 0.0002) {mult2} ← {mult2} / 2.0; ELSE
                {mult2} ← 0.8; {exit} ← TRUE; {high1} ← {high1} + 1;
            end
```

```
            end
        end
        Nopt = computeNOptimal();
    end

end
```

Στη συνέχεια παραθέτουμε την διαδικασία υπολογισμού του διανύσματος με τον δείκτη επίδοσης για κάθε κλάση δοσοληψιών και στις επόμενες παραγράφους αναλύουμε το τρόπο υπολογισμού κάθε μιας από τις συναρτήσεις που χρησιμοποιούνται στην διαδικασία υπολογισμού του διανύσματος με τον δείκτη επίδοσης για κάθε κλάση δοσοληψιών.

Διαδικασία 6 *Διαδικασία υπολογισμού* **computePerformanceIndex(p)**

```
**********************************************************************
** Όλα τα παρακάτω υπολογίζονται με βάση τον πίνακα πιθανοτήτων p(i,j)
** όπου το στοιχείο p(i,j) φανερώνει την πιθανότητα δρομολόγησης μίας
** δοσοληψίας της κλάσης i στον κόμβο επεξεργασίας j.
**********************************************************************
/*
* Η συνάρτηση computeAvgNumOfLocalMisses() υπολογίζει τον πίνακα Mloc(i,j)
* που περιλαμβάνει τον αριθμό των σελίδων που δεν βρίσκονται στον τοπικό
* ενταμιευτή, για μία δοσοληψία κλάσης i που εκτελείται στον κόμβο j.
*/
Mloc = computeAvgNumOfLocalMisses();
/*
* Η συνάρτηση computeAvgNumOfDASDMisses() υπολογίζει τον πίνακα
* MDasd(i,j) που περιλαμβάνει τον αριθμό των σελίδων που δεν βρίσκονται
* στον κοινόχρηστο ενταμιευτή, για μία δοσοληψία κλάσης i που εκτελείται
* στον κόμβο j. Αν δεν υπάρχει κοινόχρηστος ενταμιευτής τότε
* MDasd(i,j) = Mloc(i,j) για κάθε i,j.
*/
MDasd = computeAvgNumOfDASDMisses();
/*
* Η συνάρτηση computeProcessingRequirements() υπολογίζει τον πίνακα S(i,j)
* που περιλαμβάνει το μέσο χρονικό κόστος επεξεργασίας για μία
* δοσοληψία κλάσης i
* που εκτελείται στον κόμβο j.
*/
S = computeProcessingRequirements();
/*
* Η συνάρτηση computeProcessorUtilization() υπολογίζει τον πίνακα Util(j)
* που περιλαμβάνει τον βαθμό χρησιμοποίησης κάθε κόμβου j.
```

/
Util = **computeProcessorUtilization();**
/
** Η συνάρτηση* **computeResponseTime()** *υπολογίζει τον πίνακα R(i,j)*
** όπου υπολογίζεται ο εκτιμώμενος χρόνος απόκρισης για μία*
** δοσοληψία κλάσης i*
** που εκτελείται στον κόμβο j.*
/
R = **computeResponseTime();**
/
** Η συνάρτηση* **computeAvgResponseTime()** *υπολογίζει τον πίνακα Avg(i)*
** όπου υπολογίζεται ο εκτιμώμενος μέσος χρόνος απόκρισης*
** για κάθε κλάση δοσοληψιών i.*
/
AvgR = **computeAvgResponseTime();**
/
** Στον πίνακα PerfIndex(i) υπολογίζεται ο εκτιμώμενος δείκτης επίδοσης*
** για κάθε κλάση δοσοληψιών i.*
/

$PerfIndex := (\frac{AvgR(1)}{g(1)}, ..., \frac{AvgR(M)}{g(M)})$

end

Ο υπολογισμός του Nopt πίνακα γίνεται με βάση τον τελικό πίνακα πιθανοτήτων δρομολόγησης $p(i,j)$ και του ρυθμού αφίξεως για κάθε κλάση δοσοληψιών που τίθεται να εξυπηρετήσει το σύστημα. Η εξίσωση 5.11 χρησιμοποιείται για αυτόν τον υπολογισμό και εκτελείται για κάθε δυνατό ζευγάρι από i και j.

$$Nopt(i,j) := p(i,j) * Arrival_Rate(i) \tag{5.11}$$

5.3.1 Υπολογισμός της συνάρτησης computeAvgNumOfLocalMisses()

Υποθέτουμε τα εξής :

- Ότι η βάση δεδομένων αποτελείται από K ($F_1, ..., F_K$) και κάθε αρχείο αποτελείται από έναν αριθμό από σελίδες που βρίσκονται αποθηκευμένες στο διάνυσμα $PagesInFile(k)$ για $k = 1, ...,$ K.
- Ο μέσος αριθμός από προσβάσεις μίας δοσοληψίας της κλάσης i στο αρχείο k είναι A(i,k).
- $w(i,k)$: είναι η πιθανότητα πρόσβασης στη βάση δεδομένων για write ή update πράξη.

- $Valid_pages(k)$: είναι ο αριθμός των έγκυρων σελίδων του αρχείου k που βρίσκονται στον τοπικό ενταμιευτή.
- Ότι η πολιτική αντικατάστασης των σελίδων του ενταμιευτή είναι η Least Recently Used (LRU), σύμφωνα με την οποία, η σελίδα που αντικαταστάται κάθε φορά, είναι αυτή η οποία έχει να προσπελαστεί για χρονικό διάστημα μεγαλύτερο σε σχέση με όλες τις υπόλοιπες σελίδες οι οποίες βρίσκονται και αυτές στον ενταμιευτή.
- Το μέγεθος του ενταμιευτή του κόμβου j, είναι B_j και μετράτε σε σελίδες.

Τότε ο αριθμός από τον μέσο αριθμό της αποτυχίας να βρεθεί μία σελίδα στον ενταμιευτή του κόμβου j μετά από αίτηση δοσοληψίας της κλάσης $i(Mloc(i,j))$, δίνεται από τον ακόλουθο αλγόριθμο:

Διαδικασία 7 *Διαδικασία υπολογισμού Mloc(i,j) για τον κόμβο j*

/
** Υπολόγισε (για κάθε κόμβο και αρχείο) τον ρυθμό με τον οποίο*
** προσπελάσονται οι σελίδες του αρχείου k από τον κόμβο j για ανάγνωση rateR*
** για εγγραφή rateW και συνολικά rate. Επίσης υπολόγισε τον ρυθμό που*
** ακυρώνονται οι σελίδες (invrate)*
\/

$rateR(j,k) \leftarrow \sum_{i=1}^{M} Arrival_Rate(i) * p(i,j) * A(i,k) * [1.0 - w(i,k)]$

$rateW(j,k) \leftarrow \sum_{i=1}^{M} Arrival_Rate(i) * p(i,j) * A(i,k) * w(i,k)$

$rate(j,k) \leftarrow rateR(j,k) + rateW(j,k)$

$invrateR(j,k) \leftarrow \sum_{i=1}^{M} Arrival_Rate(i) * [1.0 - p(i,j)] * A(i,k) * w(i,k)$

$sum \leftarrow \sum_{k=1}^{K} rate(j,k)$

/
** Υπολόγισε (για κάθε αρχείο) το ποσοστό από τις σελίδες που είναι έγκυρες.*
\/

$Valid_pages(k) \leftarrow \frac{rate(j,k)}{sum}$

FOR (n←1) TO B_j DO

Για κάθε αρχείο k υπολόγισε

$valrate(k) \leftarrow rate(j,k) * [1.0 - \frac{Valid_pages(k)}{PagesInFile(k)}] - invrate(j,k) * \frac{Valid_pages(k)}{PagesInFile(k)}$

$sum \leftarrow \sum_{k=1}^{K} max\{0, valrate(k)\}$

Για κάθε αρχείο k θέσε

$Valid_pages(k) \leftarrow min\{PagesInFile(k), Valid_pages(k) + \frac{valrate(k)}{sum}\}$

end
Για κάθε κλάση i υπολόγισε το
$Mloc(i,j) \leftarrow \sum_{k=1}^{K} A(i,k) * [1.0 - \frac{Valid_pages(k)}{PagesInFile(k)}]$
end

end

5.3.2 Υπολογισμός της συνάρτησης computeAvgNumOfDASDMisses()

Υποθέτουμε καταρχήν ότι η κοινόχρηστη μνήμη που αναλαμβάνει να παίξει τον ρόλο του κοινόχρηστου ενταμιευτή αποθηκεύει μόνο σελίδες που έχουν μεταβληθεί σε έναν από τους κόμβους του συστήματος. Θέλουμε να υπολογίσουμε το $MDasd(i,j)$, τον αριθμό δηλαδή από της σελίδες που πραγματοποιεί προσπέλαση μία δοσοληψία της κλάσης i στον κόμβο j και οι οποίες δεν βρίσκονται στον κοινόχρηστο buffer. Θυμίζουμε ότι ένας κόμβος σε περίπτωση που ο ιδιωτικός buffer δεν έχει την ζητούμενη σελίδα τότε εάν υπάρχει κοινόχρηστος buffer τότε ψάχνει και εκεί. Σε περίπτωση που δεν βρεθεί και εκεί τότε αναγκαστικά πηγαίνει στην περιφεριακή μνήμη του συστήματος. Ο πίνακας λοιπόν $MDasd(i,j)$ περιέχει τον αριθμό από τις σελίδες για τις οποίες η δοσοληψία θα πρέπει να προσπελάσει την περιφεριακή μνήμη του συστήματος.

Υποθέτουμε και πάλι ότι η πολιτική αντικατάστασης των σελίδων του κοινόχρηστου ενταμιευτή είναι η LRU. Ο πίνακας $Msmem(i)$ δηλώνει την πιθανότητα για buffer miss από μία δοσοληψία της κλάσης i. Όπως φαίνεται και από τον τρόπο ορισμού του πίνακα $Msmem$ αυτός δεν εξαρτάται από τον εκάστοτε πίνακα πιθανοτήτων δρομολόγησης. Οπότε δεν απαιτείται να επανυπολογιστεί. Το πρόβλημα της ακύρωσης σελίδων δεν υπάρχει στον κοινόχρηστο ενταμιευτή. Ο αλγόριθμος για τον υπολογισμό του $MDasd(i,j)$ δίνεται παρακάτω.

Διαδικασία 8 *Διαδικασία υπολογισμού MDasd(i,j) για τον κόμβο j και την κλάση i*

/
** Υπολόγισε για κάθε αρχείο τον ρυθμό με τον οποίο*
** προσπελάσονται οι σελίδες του αρχείου k.*
**/*
$rate(k) \leftarrow \sum_{i=1}^{M} Arrival_Rate(i) * A(i,k) * w(i,k)$

$sum \leftarrow \sum_{k=1}^{K} rate(j)$

/
** Υπολόγισε για κάθε αρχείο τον ποσοστό από τις σελίδες που βρίσκονται*
** στον κοινόχρηστο ενταμιευτή.*
**/*
$Valid_pages(k) \leftarrow \frac{rate(k)}{sum}$

/*
* *B είναι το μέγεθος του buffer σε σελίδες.*
*/

FOR (n←1) TO B DO
Για κάθε αρχείο k υπολόγισε
$valrate(k) \leftarrow rate(k) * [1.0 - \frac{Valid_pages(k)}{PagesInFile(k)}]$

$sum \leftarrow \sum_{k=1}^{K} max\{0, valrate(k)\}$

Για κάθε αρχείο k θέσε
$Valid_pages(k) \leftarrow min\{PagesInFile(k), Valid_pages(k) + \frac{valrate(k)}{sum}\}$
end
Για κάθε κλάση i και αρχείο k υπολόγισε το
$Freq(i,k) \leftarrow \frac{A(i,k)}{\sum_{k=1}^{K} A(i,k)}$

Για κάθε κλάση i υπολόγισε το
$Msmem(i) \leftarrow \sum_{k=1}^{K} Freq(i,k) * w(i,k) * [1.0 - \frac{Valid_pages(k)}{PagesInFile(k)}]$

Για κάθε κλάση i και κόμβο j υπολόγισε το
$MDasd(i,j) \leftarrow Mloc(i,j) * Msmem(i)$
end

end

Στην περίπτωση όπου δεν υπάρχει κοινόχρηστος ενταμιευτής, το διάνυσμα $Msmem$ είναι το $(1, ..., 1)$ με M στοιχεία έτσι ώστε $MDasd(i,j) = Mloc(i,j)$ για κάθε κόμβο j και για κάθε κλάση δοσοληψιών i.

5.3.3 Υπολογισμός των αναγκών επεξεργασίας

Προκειμένου να γίνει δυνατός ο προσδιορισμός των αναγκών επεξεργασίας μίας δοσοληψίας της κλάσης i που εκτελείται στον κόμβο $j(S(i,j))$, πρέπει να ορίσουμε μία σειρά από στατιστικές παραμέτρους και όχι μόνο, που χρησιμοποιούνται προκειμένου να γίνει ο υπολογισμός του πίνακα $S(i,j)$.

- $A_db(i)$: είναι ο μέσος αριθμός των σελίδων στις οποίες κάνει προσπέλαση μία δοσοληψία της κλάσης i.
- $PL_base(i,j)$: Το μέσο κόστος επεξεργασίας για μία δοσοληψία κλάσης i που εκτελείται στον κόμβο j. Αυτό περιλαμβάνει το κόστος για το lock μίας σελίδας, την αίτηση προσπέλασης, καθώς και την εκτέλεση του προγράμματος.
- $PL_io(j)$: Το μέσο κόστος για μία πράξη εισόδου/εξόδου (I/O) στην περιφερειακή μνήμη του συστήματος από τον κόμβο j.

- $PL_smem(j)$: Το μέσο κόστος για την πρόσβαση στον κοινόχρηστο ενταμιευτή από έναν κόμβο j.

Σε όλες τις περιπτώσεις το κόστος είναι σε μονάδες χρόνου. Προκειμένου να γίνει η μέτρηση αυτών των παραμέτρων, αρχικά γινόταν ένα "τρέξιμο" του μοντέλου με το δοθέν φόρτο εργασίας (monitor run) και στο τέλος αυτού του run, οι τιμές αυτές αποθηκεύονταν σε ένα αρχείο. Ο τρόπος δρομολόγησης ήταν τυχαίος προκειμένου οι τιμές αυτές να είναι όσο το δυνατό πιο ρεαλιστικές.

Ο υπολογισμός των αναγκών επεξεργασίας μίας δοσοληψίας της κλάσης i που εκτελείται στον κόμβο $j(S(i,j))$ (εξίσωση 5.12) δεν λαμβάνει υπόψη του την επιβάρυνση λόγω του data contention προβλήματος. Έτσι η παρουσία πολλών ενεργών δοσοληψιών που είναι παράγοντας που ενισχύει αυτό το πρόβλημα δεν λαμβάνεται υπόψη στον υπολογισμό της εξίσωσης αλλά και πουθενά αλλού στη όλη σχεδίαση του αλγορίθμου.

$$S(i,j) := A_db(i) * PL_base(i,j) + Mloc(i,j) * PL_smem(j) + MDasd(i,j) * PL_io(j) \quad (5.12)$$

Όπως είναι φανερό οι δύο τελευταίοι όροι εξαρτώνται από τον πίνακα δρομολόγησης $p(i,j)$ συνεπώς και οι ανάγκες επεξεργασίας μεταβάλλονται με την αλλαγή του πίνακα δρομολόγησης.

5.3.4 Υπολογισμός του βαθμού χρήσης των κόμβων

Ο βαθμός χρήσης του επεξεργαστή ενός κόμβου j δίνεται απλά από την εξίσωση 5.13.

$$Util(j) := \sum_{i=1}^{M} Arrival_Rate(i) * p(i,j) * S(i,j) \quad (5.13)$$

5.4 Πειραματικά αποτελέσματα χρήσης αλγορίθμου δρομολόγησης

Στα πλαίσια της αξιολόγησης του αλγορίθμου έγιναν μία σειρά από πειράματα. Επειδή από νωρίς φάνηκε η απουσία ενός δυναμικού τρόπου δρομολόγησης, έγινε μία προσπάθεια να τροποποιηθεί ο αλγόριθμος κατά τέτοιο τρόπο ώστε να συμπεριφέρεται καλύτερα σε στιγμιαίες αλλαγές στον φόρτο εργασίας.

Πιο συγκεκριμένα παρατηρήθηκε ότι ενώ ορίζοντας κανείς την ίδια πιθανότητα για κάθε κλάση θα περίμενε ίδιο αριθμό από ενεργές δοσοληψίες για κάθε κλάση, εντούτοις υπήρχαν χρονικές στιγμές κατά τις οποίες ο αριθμός από δοσοληψίες μίας κλάσης που ήταν ενεργές, ήταν πολύ διαφορετικός από αυτόν των υπολοίπων. Έτσι π.χ. στην περίπτωση που είχαμε δύο κλάσεις και δύο κόμβους και αναθέταμε την κάθε κλάση σε έναν κόμβο με πιθανότητα 1.0, τότε ο μέσος χρόνος απόκρισης σε κάποια χρονικά διαστήματα ήταν πολύ μεγαλύτερος

Πείραμα 1 (SD) περίπτωση	
number of Nodes	5 (1 Front-End)
MPL	100
CPU speed	15 MIPS
Private buffer	4000 pages
Communication Network:	
packet size	1024 bytes
transfer rate	80Mbits
Workload	
Transaction Classes	4
Class Submit Prob.	0.25
Tx Appl. Burst	500000.0
blocksAccessed_min	1
blocksAccessed_min	6
Each class with affinity to a different relation	
writeProbability	0.5
Users	100
Think time	1.0
Performance Goal	1.2
Data distribution	
Relations (Files)	4
Pages Per File	10000
Total DB size	40000 pages
I/O delay	10 msec

Πείραμα 1 (SIM) περίπτωση	
Private buffer	2000 pages
Shared Buffer	20000 pages
Shared Buffer I/O delay	around 4msec
Workload	
Performance Goal	1.0

Πίνακας 5.1: Βασικές παράμετροι του πρώτου πειράματος.

από τον μέσο όρο. Αυτό συνέβαινε διότι η πλειοψηφία των ενεργών δοσοληψιών ήταν μίας κλάσης και σύμφωνα με τον τρόπο ανάθεσης των κλάσεων στους κόμβους, ο ένας κόμβος ήταν υπερφορτωμένος ενώ ο άλλος μπορεί να μην είχε και καθόλου ενεργές δοσοληψίες. Έτσι ορίσαμε το δυναμικό κομμάτι του αλγορίθμου το οποίο αφού αποφασίσει ο προηγούμενος αλγόριθμος σε ποιόν κόμβο θέλει να δρομολογήσει την δοσοληψία λαμβάνει υπόψη του και το μέγεθος της ουράς. Παίρνει την διαφορά των ενεργών δοσοληψιών στον κόμβο που είναι να δρομολογηθεί η δοσοληψία και στον κόμβο που έχει τον ελάχιστο αριθμό δοσοληψιών και συνεπώς υποχρησιμοποιείται την δεδομένη χρονική περίοδο. Αν αυτή η διαφορά είναι μεγαλύτερη από ένα ποσοστό A των συνολικών ενεργών δοσοληψιών στο σύστημα τότε μεταβάλλει την απόφαση και δρομολογεί την δοσοληψία στον κόμβο που έχει τον μικρότερο αριθμό από ενεργές δοσοληψίες. Δηλαδή μεταβάλλει την επιλογή του ώστε να είναι σύμφωνη με την επιλογή που κάνει ο αλγόριθμος Join Shortest Queue (JSQ) στην παρούσα κατάσταση. Με αυτόν τον τρόπο πιθανώς να θυσιαστεί ένα ποσοστό από την πιθανότητα να βρεθεί μία σελίδα στον ενταμιευτή, αλλά από την άλλη είναι δυνατό να δώσει καλύτερα αποτελέσματα όταν οι πιθανότητες αποστολής μεταβάλλονται κατά την διάρκεια του πειράματος και με την προϋπόθεση ότι αυτό το ποσοστό είναι αρκετά κοντά στο 1.0 ώστε κατά βάση να γίνεται επιλογή του κόμβου που ορίζει ο στατικός αλγόριθμος.

Ο στατικός αλγόριθμος δρομολόγησης θα ονομάζεται από εδώ και στο εξής SDR (Shared Disks Router) ενώ οι δυναμικοί αλγόριθμοι θα ονομάζονται SDR**A** όπου A είναι το προαναφερθέν ποσοστό. Ουσιαστικά δηλαδή SDR1.0 είναι ο ίδιος ο SDR αλγόριθμος ενώ SDR0.0 είναι ο αλγόριθμος JSQ. Τα πειράματα που έγιναν και για τις δύο αρχιτεκτονικές ώστε να μπορεί να γίνει και μία σύγκριση τους, ενώ ως πολιτική χρονοπρογραμματισμού επιλέχθηκε η Round Robin (RR). Το πρώτο πείραμα είναι ένα απλό παράδειγμα για να φανεί ο τρόπος που επιδρά το ποσοστό αυτό στην απόδοση του συστήματος.

Οι βασικές παράμετροι φαίνονται στο πίνακα 5.1. Στον πίνακα αριστερά φαίνονται οι παράμετροι για το σύστημα με την SD αρχιτεκτονική, ενώ δίπλα φαίνονται όλες οι διαφορές του συστήματος με την αρχιτεκτονική SIM. Οι μόνες διαφορές είναι ότι προστέθηκε ένας κοινόχρηστος ενταμιευτής και συγχρόνως μεταβλήθηκε και το μέγεθος του τοπικού ενταμιευτή, στο μισό. Παράλληλα μειώθηκε και ο στόχος επίδοσης για κάθε κλάση ώστε να είναι πιο κοντά στην επίδοση του συστήματος.

Στο πρώτο απλό παράδειγμα υπάρχουν τέσσερις κλάσεις δοσοληψιών, που εισαγάγουν ίδιο φόρτο στο σύστημα και έχουν συγγένεια με διαφορετικά τμήματα της βάσης δεδομένων. Επίσης η πιθανότητα εμφάνισής τους στο σύστημα είναι ακριβώς η ίδια και οι κόμβοι είναι και αυτοί τέσσερις. Είναι προφανές ότι η καλύτερη δρομολόγηση θα ήταν να ανατεθεί η κάθε κλάση σε έναν από τους τέσσερις κόμβους με πιθανότητα 1.0 (η κάθε κλάση σε διαφορετικό κόμβο). Με αυτόν τον τρόπο το πρόβλημα της ακύρωσης των σελίδων στους ενταμιευτές δεν υφίσταται πλέον στο συγκεκριμένο παράδειγμα διότι κάθε κόμβος αναλαμβάνει ουσιαστικά ένα κομμάτι από την βάση οπότε ο ενταμιευτής αποθηκεύει στον buffer του σελίδες μόνο του συγκεκριμένου τμήματος της βάσης. Ο αριθμός από τις σελίδες που βρίσκονται στον ενταμιευτή και είναι έγκυρες θεωρητικά είναι $\frac{\text{μέγεθοςενταμιευτή}}{\text{μέγεθοςτμήματοςτηςβάσης}}$ δηλαδή 40%. Σε κάθε περίπτωση ο αλγόριθμος αναφοράς είναι ο αλγόριθμος JSQ που υποστηρίζεται από τον προσομοιωτή. Ο αλγόριθμος αυτός είναι πολύ καλός αλγόριθμος δρομολόγησης διότι εξισορροπεί τον φόρτο ανάμεσα στους κόμβους και ιδιαίτερα για την περίπτωση των αρχιτεκτονικών με κοινόχρηστα δεδομένα όπου κάθε κόμβος έχει πρόσβαση σε όλα τα δεδομένα, η εξισορρόπηση του φόρτου κάθε κόμβου είναι βασική προϋπόθεση για την καλή απόδοση ενός αλγορίθμου δρομολόγησης. Μειονέκτημά του είναι ότι δεν λαμβάνει υπόψη του την συγγένεια των κλάσεων με κάποια τμήματα της βάσης ώστε να δρομολογήσει δοσοληψίες μίας κλάσης στον ίδιο κόμβο για να επιτύχει μεγαλύτερο ποσοστό επιτυχίας εύρεσης των σελίδων στο τοπικό ενταμιευτή στις περιπτώσεις κλάσεων που εμφανίζουν συγγένεια με τμήματα της βάσης δεδομένων. Παρόλα αυτά στην περίπτωση των αρχιτεκτονικών με κοινόχρηστα δεδομένα ο αλγόριθμος αποδίδει ιδιαίτερα ικανοποιητικά και είναι ευρύτατα διαδεδομένος.

Προκειμένου να γίνει το πείραμα αρχικά έγινε ένα monitor run ώστε να συλλεγθούν ορισμένα στατιστικά στοιχεία για το πείραμα και τα οποία είναι απαραίτητα για την εφαρμογή του αλγορίθμου. Στην συνέχεια με βάση αυτές τις μετρήσεις, όπου σημειωτέον λόγω της ομοιότητας των κλάσεων ήταν και αυτές περίπου ίδιες για κάθε κλάση, έγινε η εφαρμογή του αλγορίθμου δρομολόγησης ο οποίος κατέληξε σε έναν πίνακα p(i,j) όπου ανέθετε κάθε κλάση σε έναν διαφορετικό κόμβο (καλύτερη δυνατή δρομολόγηση). Ξεκινώντας από την κλάση που στο monitor run συμπεριφέρθηκε ελάχιστα χειρότερα, την ανέθεσε στον κόμβο που εμφάνιζε την καλύτερη επίδοση με βάση αυτό το run. Συνεχίζοντας

Πίνακας 5.2: Διάφορες μετρήσεις για το σύστημα του πειράματος 1 (SD αρχιτεκτονική).

Αλγόριθμος	*Overall Tx Response Time*	*% of Tx Routed according JSQ*
JSQ	1.31762	100.0%
SDR0.02	1.25442	44.4%
SDR0.04	1.19965	34.4%
SDR0.06	1.15524	27.7%
SDR0.08	1.12936	22.9%
SDR0.10	1.09661	19.2%
SDR0.15	1.03896	12.9%
SDR0.20	0.99369	9.1%
SDR0.25	0.96722	6.5%
SDR0.30	0.94904	4.80%
SDR0.40	0.91980	2.33%
SDR0.50	0.90559	0.83%
SDR0.60	0.90438	0.28%
SDR0.70	0.89960	0.06%
SDR0.80	0.90694	0.007%
SDR0.90	0.90012	0.0009%
SDR	0.90312	0.0%

με αυτόν τον τρόπο επέλεξε για την επόμενη κλάση τον κόμβο από τους υπολοίπους όπου είχε η κλάση την καλύτερη επίδοση. Αυτό συμβαίνει διότι ο κόμβος λόγω του ότι είχε βαθμό χρήσης αρκετά μεγάλο και σε συνδυασμό με το ελαττωμένο buffer hit ratio που θα εμφάνιζε αν δρομολογούσε δύο κλάσεις στον ίδιο κόμβο, εμφανίζεται κατά την παραγώγιση ως κόμβος που δεν μπορεί να ανταγωνιστεί σε επίδοση τους άλλους κόμβους του συστήματος. Τελικά δηλαδή καταλήγει σε ένα βέλτιστο πίνακα δρομολόγησης από όπου οποιαδήποτε διαταραχή του πίνακα οδηγεί σε δείκτες επίδοσης οι οποίοι είναι χειρότεροι.

Στους πίνακες 5.2 και 5.3 φαίνονται τα αποτελέσματα των πειραμάτων για την περίπτωση της *Shared Disk* αρχιτεκτονικής ενώ στους πίνακες 5.4 και 5.5 για την περίπτωση της *Shared Intermediate Memory* αρχιτεκτονικής.

Όπως φαίνεται από το πίνακα 5.2 ο αλγόριθμος SDR πετυχαίνει μία βελτίωση σε σχέση με τον JSQ της τάξης του 31.5%. Αντίστοιχα βλέπουμε και τον τρόπο που η επιλογή της δρομολόγησης με βάση την απόφαση του SDR αλγορίθμου μεταβάλλεται, όταν αλλάζει το ποσοστό της μέγιστης διαφοράς στον αριθμό των ενεργών δοσοληψιών που επιτρέπουμε μεταξύ του κόμβου επιλογής του SDR και του κόμβου με το ελάχιστο αριθμό δοσοληψιών. Από τον πίνακα μπορούμε να δούμε ότι την καλύτερη απόδοση έχει για ποσοστό κοντά στο 70%. Σε αυτήν την περίπτωση κάνοντας μία καλύτερη εξισορρόπηση και ελαττώνοντας ελαφρώς την πιθανότητα εύρεσης της σελίδας στον τοπικό ενταμιευτή (λόγω buffer invalidation) πετυχαίνει καλύτερη επίδοση.

Το ποσοστό αυτό διαφέρει από πείραμα σε πείραμα και εξαρτάται από πολλούς

Πίνακας 5.3: Διάφορες μετρήσεις για το σύστημα του πειράματος 1 (SD αρχιτεκτονική)..

Αλγόριθμος	**Διάφορες Μετρήσεις**			
	Buffer hit ratio	*Node Util.*	*Tx completed*	*% of Tx missed goal*
JSQ	8.21%	65.5%	172660	57.0%
SDR0.02	12.23%	66.6%	177211	52.9%
SDR0.04	15.39%	68.3%	181614	48.0%
SDR0.06	18.16%	69.6%	185672	44.3%
SDR0.08	20.29%	70.7%	187912	41.0%
SDR0.10	22.46%	71.8%	190930	39.0%
SDR0.15	26.66%	73.7%	196352	33.9%
SDR0.20	29.91%	75.4%	200798	30.4%
SDR0.25	31.89%	76.2%	203150	28.2%
SDR0.30	33.78%	77.1%	205534	27.3%
SDR0.40	36.57%	78.2%	208219	26.0%
SDR0.50	38.35%	78.8%	209637	24.0%
SDR0.60	39.15%	79.4%	210013	25.7%
SDR0.70	39.37%	79.7%	210355	25.3%
SDR0.80	39.53%	79.4%	210014	25.9%
SDR0.90	39.59%	79.5%	210118	25.5%
SDR	39.53%	78.5%	210106	25.6%

Πίνακας 5.4: Διάφορες μετρήσεις για το σύστημα του πειράματος 1 (SIM αρχιτεκτονική).

Αλγόριθμος	*Overall Tx Response Time*	*% of Tx Routed according JSQ*
JSQ	1.25838	100.0%
SDR0.02	1.22714	39.5%
SDR0.04	1.20028	28.0%
SDR0.06	1.17529	21.4%
SDR0.08	1.16039	17.2%
SDR0.10	1.14810	14.1%
SDR0.15	1.13935	9.64%
SDR0.20	1.12464	6.85%
SDR0.25	1.11179	5.08%
SDR0.30	1.11064	3.66%
SDR0.40	1.11563	1.69%
SDR0.50	1.11447	0.4948%
SDR0.60	1.11579	0.1864%
SDR0.70	1.10987	0.0527%
SDR0.80	1.11508	0.0026%
SDR0.90	1.11223	0.0%
SDR	1.11620	0.0%

παράγοντες όπως το κόστος πρόσβασης στην περιφεριακή μνήμη, το πλήθος των ενεργών δοσοληψιών (κατά μέσο όρο) στο σύστημα σε κάθε χρονική στιγμή κ.α. Για την εύρεση του κατάλληλου ποσοστού προκειμένου να γίνει εφικτή η βελτίωση αυτή θα πρέπει να γίνει μία σειρά δοκιμών. Επίσης το ποσοστό βελτίωσης σε σχέση με τον JSQ εξαρτάται και πάλι από τις παραμέτρους του προβλήματος (μέγεθος ενταμιευτή, τοπικότητα προσπελάσεων, καθυστέρηση για I/O) αλλά σαφώς έχει αρκετά καλύτερη επίδοση από τον JSQ κάτι που φαίνεται και από την διαφορά στο buffer hit ratio μεταξύ του SDR και JSQ (39.5% και 8.2% αντίστοιχα) και με δεδομένο ότι η εύρεση μίας σελίδας στον τοπικό ενταμιευτή είναι επιθυμητή καθώς το κόστος πρόσβασης στην περιφεριακή μνήμη είναι μεγάλο. Παράλληλα όταν ο αριθμός των κόμβων είναι μεγάλος και η βάση είναι κοινόχρηστη τότε είναι πολύ πιθανό να καταλήξει η εξυπηρέτηση αιτήσεων προσπέλασης στον δίσκο να είναι υπερβολικά χρονοβόρα (bottleneck).

Ο βαθμός χρήσης των κόμβων διατηρήθηκε σε υψηλά επίπεδα. Όσο πιο γρήγορα εξυπηρετούνταν οι χρήστες από το σύστημα τόσο πιο γρήγορα έστελναν την επόμενη δοσοληψία για εξυπηρέτηση (κλειστό μοντέλο). Γι' αυτό όταν το σύστημα συμπεριφέρεται καλύτερα εμφανίζει και μεγαλύτερο αριθμό από δοσοληψίες που εξυπηρετούνται στον ίδιο χρόνο προσομοίωσης. Η παρατηρούμενη αύξηση του βαθμού χρήσης οφείλεται στο ότι ο κάθε κόμβος του συστήματος δεν είναι υποχρεωμένος να περιμένει συχνά απαντήσεις στις αιτήσεις προσπέλασης στην περιφεριακή μνήμη. Μπλοκάρεται λοιπόν πιο σπάνια.

Ανάλογα μεγάλη είναι και αύξηση του ρυθμού με τον οποίο οι δοσοληψίες

Πίνακας 5.5: Διάφορες μετρήσεις για το σύστημα του πειράματος 1 (SIM αρχιτεκτονική).

Αλγόριθμος	**Διάφορες Μετρήσεις**			
	Local buffer hit ratio	*Node Util.*	*Shared buffer hit ratio*	*% of Tx missed goal*
JSQ	4.51%	67.7%	46.86%	66.941%
SDR0.02	7.54%	68.4%	46.06%	65.384%
SDR0.04	9.84%	69.1%	45.36%	63.678%
SDR0.06	11.60%	69.7%	44.70%	61.978%
SDR0.08	12.85%	70.5%	44.36%	60.807%
SDR0.10	13.81%	70.8%	44.08%	59.523%
SDR0.15	15.44%	70.9%	43.47%	58.158%
SDR0.20	16.71%	71.4%	43.03%	56.044%
SDR0.25	17.46%	71.7%	42.75%	53.973%
SDR0.30	18.05%	71.8%	42.49%	53.068%
SDR0.40	19.07%	71.9%	42.01%	51.923%
SDR0.50	19.56%	72.1%	41.97%	50.962%
SDR0.60	19.73%	72.1%	41.98%	50.583%
SDR0.70	19.79%	72.2%	41.83%	50.384%
SDR0.80	19.86%	72.2%	41.82%	50.628%
SDR0.90	19.83%	71.8%	41.79%	50.525%
SDR	19.88%	71.9%	41.84%	50.817%

εξυπηρετούνται μέσα στον ζητούμενο χρόνο που καθορίζει ο στόχος επίδοσης που έχει δοθεί από τον χρήστη.

Ανάλογες είναι και οι διαπιστώσεις για την SIM αρχιτεκτονική. Η βελτίωση των μεγεθών δεν φτάνει στα επίπεδα της SD αρχιτεκτονικής διότι παράλληλα μειώσαμε σημαντικά (50%) το μέγεθος του ιδιωτικού ενταμιευτή. Αυτό στο επόμενο πείραμα δεν έγινε προκειμένου να φανεί η διαφορά. Μία βασική παρατήρηση είναι ότι από την ύπαρξη του κοινόχρηστου ενταμιευτή φαίνεται να επωφελείται ιδιαίτερα και ο JSQ αλγόριθμος καθώς οι σελίδες με πιθανότητα περίπου 47% βρίσκονται στον κοινόχρηστο ενταμιευτή με αποτέλεσμα να αποφεύγεται η πρόσβαση στον δίσκο και με συνέπεια να βελτιώνεται και η απόδοση του συστήματος. Από την άλλη μεριά έχει χειροτερεύσει η απόδοσή του SDR καθώς χρειάζεται να κάνει προσπέλαση στον κοινόχρηστο ενταμιευτή για περίπου 20% των προσπελάσεων για τις οποίες στο προηγούμενο πείραμα έφτανε η πρόσβαση στον τοπικό ενταμιευτή.

Στο δεύτερο πείραμα που έγινε το μέγεθος του ενταμιευτή δεν μεταβλήθηκε στη περίπτωση της SIM αρχιτεκτονικής. Το σύστημα αποτελείται από 1 front-end και 6 back-end κόμβους. Υπάρχουν 16 κλάσεις δοσοληψιών όπου στους πίνακες 5.7, και 5.8 φαίνονται όλα τα χαρακτηριστικά των κλάσεων αυτών. Ο αριθμός σε μία θέση του πίνακα 5.7 δηλώνει την πιθανότητα η αίτηση προσπέλασης μίας δοσοληψίας της κλάσης που δηλώνει η γραμμή του πίνακα, να απευθύνεται στη συγκεκριμένη σχέση - αρχείο που υποδεικνύει η στήλη. Η πιθανότητα να είναι για write/update πράξη είναι σταθερά 0.5 για όλες τις κλάσεις. Στον πίνακα 5.8 φαίνονται επίσης και οι διαφορές στους δείκτες επίδοσης που έχουν τεθεί στις κλάσεις δοσοληψιών για τις δύο αρχιτεκτονικές. Υπάρχουν 16 αρχεία - σχέσεις αλλά αυτός είναι ένας τρόπος για να μοντελοποιήσουμε 8 αρχεία όπου η πιθανότητα πρόσβασης σε ένα τμήμα του αρχείου είναι μεγαλύτερη από την πιθανότητα πρόσβασης στο υπόλοιπο (skewed access pattern). Έτσι η σχέση χωρίζεται σε δύο σχέσεις σε ποσοστό 25% και 75% με 5000 και 15000 σελίδες αντίστοιχα. Το αρχείο με το 25% των πλειάδων ονομάζεται HOT κομμάτι του αρχείου και το υπόλοιπο COLD (στον πίνακα τα δύο αυτά τμήματα φαίνονται ως H και C για εξοικονόμηση χώρου). Συγκεκριμένα σε αυτό το πείραμα χρησιμοποιούμε την κατανομή προσπελάσεων 80%-25% όπου 80% των προσπελάσεων σε ένα αρχείο αφορά το 25% των δεδομένων του αρχείου. Με αυτόν τον τρόπο για κάθε σχέση υπάρχει ένα τμήμα για το οποίο υπάρχει μεγαλύτερος ανταγωνισμός για τον κλείδωμα δεδομένων αλλά παράλληλα επειδή οι σελίδες που ανήκουν σε αυτό το τμήμα έχουν μεγαλύτερη πιθανότητα να προσπελαστούν από μία δοσοληψία, γι' αυτό και η πιθανότητα να βρεθούν στον τοπικό ενταμιευτή είναι αυξημένη. Η πρακτική αυτή ακολουθείται στις αναφορές [YD94a], [YD94b].

Οι υπόλοιπες παράμετροι του πειράματος φαίνονται στον πίνακα 5.6 όπου ο πίνακας αριστερά δίνει τις παραμέτρους του πειράματος με την αρχιτεκτονική SD ενώ ο πίνακας δεξιά τις παραμέτρους που μεταβλήθηκαν ή προστέθηκαν στο πείραμα με την αρχιτεκτονική SIM. Για το δεύτερο πείραμα δεν παρατίθενται τα αποτελέσματα από τους αλγορίθμους SDR0.02, SDR0.04, SDR0.06 και SDR0.08 καθώς η παράθεσή τους στο προηγούμενο πείραμα έγινε καθαρά για λόγους κατανόησης της λειτουργίας του αλγορίθμου και της συμπεριφοράς του όταν μεταβάλλεται το ποσοστό αυτό.

Ο πίνακας δρομολόγησης που προκύπτει από την εφαρμογή του αλγορίθμου SDR φαίνεται στον πίνακα 5.9 όπου κάθε θέση του πίνακα δηλώνει την αντίστοιχη πιθανότητα δρομολόγησης μίας κλάσης δοσοληψιών σε έναν κόμβο. Η ιδανική περίπτωση θα ήταν εάν το σύστημα είχε 4 κόμβους αντί των 6 που έχουμε εδώ. Σε μία τέτοια περίπτωση θα μπορούσε να γίνει η διαμοίραση των κλάσεων στους κόμβους με τέτοιο τρόπο ώστε η χρήση των κόμβων να είναι ίδια αλλά και να μην υπάρχει περίπτωση για ακύρωση σελίδων. Ο ιδανικός τρόπος δρομολόγησης σε αυτή την περίπτωση θα δρομολογούσε τις κλάσεις που αναφέρονται σε αρχεία που βρίσκονται στην ίδια περιοχή μέσα από τις περιοχές του σχήματος 5.9 σε έναν κόμβο. Οι περιοχές αυτές χωρίζονται στον πίνακα με μία επιπλέον κάθετη γραμμή. Π.χ. μία περιοχή αποτελείται από τις κλάσεις 1, 2, 9 και 10 που οι προσπελάσεις τους αφορούν τις σχέσεις 1H, 1C, 2H και 2C.

Στην περίπτωση όμως των 6 κόμβων θα πρέπει να γίνει και χρήση των επιπλέον δύο κόμβων προκειμένου να βελτιωθεί η επίδοση του συστήματος. Η εξισορρόπηση του φόρτου ανάμεσα στους κόμβους δεν είναι η πιο ενδεδειγμένη διότι θα εντείνει το πρόβλημα της ακύρωσης των σελίδων με αποτέλεσμα την μείωση της πιθανότητας εύρεσης μίας σελίδας στον ενταμιευτή.

Πείραμα 2 (SD) περίπτωση	
number of Nodes	7 (1 Front-End)
MPL	100
CPU speed	10 MIPS
Private buffer	6000 pages
Communication Network:	
packet size	1024 bytes
transfer rate	80Mbits
Workload	
Transaction Classes	16
Class Submit Prob.	0.0625
Users	100
Think time	0.5
Data distribution	
Relations (Files)	16
Total DB size	160000 pages
I/O delay	9 msec

Πείραμα 2 (SIM) περίπτωση	
Shared Buffer	40000 pages
Shared Buffer I/O delay	around 4msec

Πίνακας 5.6: Βασικές παράμετροι του δεύτερου πειράματος.

Πίνακας 5.7: Access Pattern για τις κλάσεις δοσοληψιών του πειράματος 2.

Κλάση	*Σχέση - Αρχείο της βάσης δεδομένων*															
	1H	1C	2H	2C	3H	3C	4H	4C	5H	5C	6H	6C	7H	7C	8H	8C
CLASS_1	0.4	0.1	0.4	0.1												
CLASS_2	0.4	0.1	0.4	0.1												
CLASS_3					0.4	0.1	0.4	0.1								
CLASS_4					0.4	0.1	0.4	0.1								
CLASS_5									0.4	0.1	0.4	0.1				
CLASS_6									0.4	0.1	0.4	0.1				
CLASS_7													0.4	0.1	0.4	0.1
CLASS_8													0.4	0.1	0.4	0.1
CLASS_9	0.8	0.2														
CLASS_10			0.8	0.2												
CLASS_11					0.8	0.2										
CLASS_12							0.8	0.2								
CLASS_13									0.8	0.2						
CLASS_14											0.8	0.2				
CLASS_15													0.8	0.2		
CLASS_16															0.8	0.2

Πίνακας 5.8: Πίνακας με τα υπόλοιπα χαρακτηριστικά για κάθε κλάση δοσοληψιών που εφαρμόστηκαν στο πείραμα 2.

Κλάση δοσοληψιών	*applicationBurst*	*BlocksAccessed*		*Class Goal*	
		Min	*Max*	*(SD)*	*(SIM)*
CLASS_1	20000	1	4	1.0	0.65
CLASS_2	60000	1	12	2.1	1.3
CLASS_3	20000	1	4	1.0	0.65
CLASS_4	60000	1	12	2.1	1.3
CLASS_5	20000	1	4	1.0	0.65
CLASS_6	60000	1	12	2.1	1.3
CLASS_7	20000	1	4	1.0	0.65
CLASS_8	60000	1	12	2.1	1.3
CLASS_9	60000	1	4	1.0	0.65
CLASS_10	60000	1	4	1.0	0.65
CLASS_11	60000	1	4	1.0	0.65
CLASS_12	60000	1	4	1.0	0.65
CLASS_13	60000	1	4	1.0	0.65
CLASS_14	60000	1	4	1.0	0.65
CLASS_15	60000	1	4	1.0	0.65
CLASS_16	60000	1	4	1.0	0.65

Αυτό θα οδηγούσε τους κόμβους να καταφεύγουν πιο συχνά στην περιφεριακή μνήμη. Από την άλλη η μη χρησιμοποίηση των δύο κόμβων εκτός του ότι θα ήταν σπατάλη πόρων, θα οδηγούσε στη υπερφόρτωση των υπολοίπων 4 κόμβων με αποτέλεσμα και πάλι την μη ικανοποιητική απόδοση του συστήματος.

Από την παράλληλη μελέτη του πίνακα δρομολόγησης και του πίνακα με το access pattern της κάθε κλάσης μπορεί κανείς να δει ότι κλάσεις που έχουν συγγένεια με τα ίδια αρχεία ανατίθενται στους ίδιους κόμβους, εκτός από την περίπτωση όπου η κλάση ανατίθενται σε άλλον κόμβο προκειμένου να εξισορροπηθεί ο φόρτος ανάμεσα στους κόμβους του συστήματος. Έτσι π.χ. η κλάση 1 ανατίθεται στον κόμβο 2 μαζί με την κλάση 2 που έχουν το ίδιο access pattern καθώς και οι κλάσεις 9 και 10 όπου οι αιτήσεις προσπέλασης γίνονται στο ίδιο τμήμα της βάσης. Ο κόμβος αυτός στην περίπτωση της δρομολόγησης με τον SDR δεν επιβαρύνεται από την ακύρωση σελίδων του ενταμιευτή οπότε θα έχει και το μεγαλύτερο δυνατό buffer hit ratio. Στον αντίποδα οι κλάσεις 7, 8 και 16 που ανατίθενται στον κόμβο 4 αντιμετωπίζουν το πρόβλημα της ακύρωσης σελίδων του ενταμιευτή από την κλάση 15 που έχει ανατεθεί στον κόμβο 6 και προσπελαύνει σχέση στην οποία απευθύνονται και οι προηγούμενες κλάσεις.

Παρόμοιες διαπιστώσεις μπορεί να κάνει κανείς και για τον πίνακα δρομολόγησης στην περίπτωση της SIM αρχιτεκτονικής (πίνακας 5.10). Η φιλοσοφία του είναι ακριβώς ίδια. Οι διαφορά στον τρόπο δρομολόγησης οφείλεται στο monitor run και στα μεγέθη που συλλέχθησαν σε αυτό. Έτσι από το monitor run φάνηκε μία ελάχιστη διαφορά στην επίδοση των κλάσεων στους διάφορους κόμβους που ήταν θέμα τυχαίο (π.χ. μεγαλύτερος αριθμός deadlocks, προσωρινή υπερφόρτωση κ.τ.λ.). Θυμίζουμε ότι το monitor run γίνεται με τυχαία δρομολόγηση και χρησιμοποιείται μόνο για τον υπολογισμό ορισμένων μεγεθών σχετικών με το κόστος πράξεων στο προς προσομοίωση σύστημα (κόστος πρόσβασης στην περιφεριακή μνήμη, στον κοινόχρηστο ενταμιευτή κ.τ.λ.). Τα μεγέθη αυτά σε ένα πραγματικό σύστημα είναι γνωστά.

Από τον πίνακα 5.11 βλέπουμε και πάλι ότι η βελτίωση είναι σημαντική σε σχέση με τον αλγόριθμο δρομολόγησης JSQ στην περίπτωση της SD αρχιτεκτονικής (της τάξης του 23% σε σχέση με τον JSQ) για το σύνολο των κλάσεων. Ο λόγος είναι ότι αυξάνεται θεαματικά η πιθανότητα να βρεθεί μία σελίδα στον τοπικό ενταμιευτή (πίνακας 5.12). Παρόλη όμως την βελτίωση υπάρχει και κάτι αρνητικό. Οι τέσσερις κλάσεις που ανατίθενται σε έναν κόμβο (κλάσεις 1, 2, 9, και 10) εμφανίζουν χειρότερη επίδοση όταν υιοθετούμε την απόφαση του στατικού SDR με μεγάλη πιθανότητα. Έτσι στους πίνακες 5.13 και 5.14 οι τέσσερις αυτές κλάσεις, εμφανίζουν ποσοστό μη επίτευξης του στόχου επίδοσης αρκετά μεγαλύτερο από αυτόν του JSQ. Βέβαια όλες οι υπόλοιπες κλάσεις εμφανίζουν πολύ καλύτερα ποσοστά. Ο λόγος αυτού του φαινομένου είναι ότι ο κόμβος που αναλαμβάνει να εξυπηρετήσει αυτές τις κλάσεις είναι υπερφορτωμένος σε σχέση με τους άλλους. Την λύση έρχονται να την δώσουν οι δυναμικές εκδόσεις του SDR όπου και πάλι η επιλογή του καλύτερου ποσοστού εξαρτάται από το πρόβλημα αλλά από το ποιές είναι οι απαιτήσεις (διότι μπορεί να μην είναι απαραίτητο να υπάρχουν τόσο μικρά όρια για τις τέσσερις αυτές κλάσεις και να επιλεχθεί ακόμα και ο SDR ο οποίος δίνει και το πιο μικρό ποσοστό στο σύνολο των δοσοληψιών που δεν καταφέρνουν να εξυπηρετηθούν μέσα στο χρονικό όριο που δίνει ο στόχος επίδοσης που τους έχει δοθεί). Θα λέγαμε ότι στην περίπτωση όπου μας ενδιαφέρει η συνολική απόδοση

Πίνακας 5.9: Πίνακας με το αποτέλεσμα του αλγορίθμου δρομολόγησης (πίνακας πιθανοτήτων) για το πείραμα 2 και την SD αρχιτεκτονική.

Κλάση δοσοληψιών.	*Κόμβος*					
	BEP_1	BEP_2	BEP_3	BEP_4	BEP_5	BEP_6
CLASS_1		1.0				
CLASS_2		1.0				
CLASS_3	1.0					
CLASS_4	0.399				0.601	
CLASS_5			1.0			
CLASS_6			1.0			
CLASS_7				1.0		
CLASS_8				1.0		
CLASS_9		1.0				
CLASS_10		1.0				
CLASS_11					1.0	
CLASS_12	1.0					
CLASS_13			1.0			
CLASS_14						1.0
CLASS_15						1.0
CLASS_16				1.0		

Πίνακας 5.10: Πίνακας με το αποτέλεσμα του αλγορίθμου δρομολόγησης (πίνακας πιθανοτήτων) για το πείραμα 2 και την SIM αρχιτεκτονική.

Κλάση δοσοληψιών.	*Κόμβος*					
	BEP_1	BEP_2	BEP_3	BEP_4	BEP_5	BEP_6
CLASS_1			1.0			
CLASS_2			1.0			
CLASS_3	1.0					
CLASS_4		1.0				
CLASS_5				1.0		
CLASS_6				1.0		
CLASS_7					0.581	0.489
CLASS_8					1.0	
CLASS_9		1.0				
CLASS_10						1.0
CLASS_11	1.0					
CLASS_12	1.0					
CLASS_13				1.0		
CLASS_14				1.0		
CLASS_15					1.0	
CLASS_16						1.0

Πίνακας 5.11: Διάφορες μετρήσεις για το σύστημα του πειράματος 2 (SD αρχιτεκτονική).

Αλγόριθμος	*Overall Tx Response Time*	*% of Tx Routed according JSQ*
JSQ	1.31986	100.0%
SDR0.10	1.14560	19.375%
SDR0.15	1.11648	14.337%
SDR0.20	1.09904	11.654%
SDR0.25	1.08144	9.476%
SDR0.30	1.07009	7.554%
SDR0.40	1.04333	3.802%
SDR0.50	1.02861	1.921%
SDR0.60	1.02620	0.857%
SDR0.70	1.01804	0.266%
SDR0.80	1.01558	0.058%
SDR0.90	1.01912	0.002%
SDR	1.01924	0.0%

Πίνακας 5.12: Μετρήσεις για το buffer hit ratio για το σύστημα του πειράματος 2 (SD αρχιτεκτονική).

Αλγόριθμος	*Κόμβος 1*	*Κόμβος 2*	*Κόμβος 3*	*Κόμβος 4*	*Κόμβος 5*	*Κόμβος 6*
JSQ	7.52%	7.62%	7.50%	7.53%	7.52%	7.63%
SDR0.10	18.54%	27.58%	25.23%	24.86%	17.23%	13.26%
SDR0.15	20.74%	30.10%	28.52%	27.78%	19.02%	14.54%
SDR0.20	22.41%	31.01%	29.99%	29.65%	19.75%	15.02%
SDR0.25	23.14%	32.27%	31.26%	30.71%	21.25%	15.73%
SDR0.30	23.43%	32.89%	32.26%	32.02%	22.71%	16.42%
SDR0.40	26.63%	34.73%	33.32%	32.90%	25.23%	19.06%
SDR0.50	28.15%	35.82%	33.93%	34.13%	27.41%	20.79%
SDR0.60	28.49%	36.91%	34.17%	34.38%	28.41%	21.94%
SDR0.70	29.20%	37.90%	34.46%	34.24%	28.77%	22.40%
SDR0.80	29.80%	37.98%	34.76%	34.49%	28.69%	22.82%
SDR0.90	29.62%	37.88%	34.56%	34.57%	29.41%	23.05%
SDR	29.52%	38.16%	34.42%	34.39%	29.23%	22.42%

Πίνακας 5.13: Μετρήσεις για το ποσοστό των δοσοληψιών που δεν έπιασαν τον στόχο επίδοσής τους (ανά κλάση) για το σύστημα του πειράματος 2 (SD αρχιτεκτονική) I.

Αλγόριθμος	1	2	3	4	5	6	7	8
JSQ	51.1%	48.7%	50.7%	49.5%	50.9%	49.4%	51.1%	49.4%
SDR0.10	42.3%	39.1%	35.7%	31.9%	42.4%	38.3%	42.5%	39.1%
SDR0.15	43.4%	41.0%	30.5%	23.9%	41.1%	36.9%	41.8%	37.6%
SDR0.20	45.2%	42.6%	24.2%	13.9%	42.9%	38.3%	43.0%	38.7%
SDR0.25	48.5%	44.6%	18.6%	9.1%	42.9%	39.4%	43.3%	40.2%
SDR0.30	52.2%	47.5%	12.6%	5.0%	43.5%	40.0%	45.2%	41.4%
SDR0.40	57.8%	52.9%	6.9%	2.0%	41.0%	36.8%	42.3%	40.0%
SDR0.50	64.8%	58.6%	4.0%	1.1%	37.5%	33.0%	34.7%	31.2%
SDR0.60	66.4%	61.2%	3.2%	0.8%	32.9%	29.2%	34.2%	30.8%
SDR0.70	66.5%	62.0%	2.6%	0.6%	32.6%	28.8%	33.8%	29.4%
SDR0.80	69.8%	63.4%	3.1%	0.8%	29.1%	25.3%	31.6%	27.7%
SDR0.90	71.5%	66.1%	2.3%	0.6%	29.9%	26.6%	28.9%	25.3%
SDR	68.0%	62.2%	3.0%	0.5%	32.4%	29.7%	31.1%	26.4%

Πίνακας 5.14: Μετρήσεις για το ποσοστό των δοσοληψιών που δεν έπιασαν τον στόχο επίδοσής τους (ανά κλάση) για το σύστημα του πειράματος 2 (SD αρχιτεκτονική) II.

Αλγόριθμος	9	10	11	12	13	14	15	16
JSQ	52.0%	52.1%	52.5%	52.6%	52.1%	52.3%	51.8%	52.2%
SDR0.10	43.9%	43.5%	36.5%	36.4%	43.3%	31.0%	30.7%	44.9%
SDR0.15	44.6%	45.0%	29.5%	29.7%	41.9%	21.7%	21.5%	44.6%
SDR0.20	46.8%	46.6%	20.4%	23.9%	44.0%	13.7%	14.3%	46.1%
SDR0.25	50.6%	49.0%	14.7%	18.5%	44.1%	8.0%	8.3%	46.7%
SDR0.30	53.5%	53.8%	9.0%	12.4%	45.1%	5.2%	5.3%	47.3%
SDR0.40	59.2%	59.6%	4.7%	6.3%	41.1%	2.5%	2.5%	46.4%
SDR0.50	66.2%	66.3%	3.2%	4.0%	36.8%	1.6%	1.5%	37.9%
SDR0.60	68.1%	68.4%	2.4%	3.3%	33.5%	1.1%	1.1%	38.1%
SDR0.70	69.1%	67.6%	2.0%	2.8%	33.8%	1.4%	1.5%	36.3%
SDR0.80	71.7%	70.8%	2.5%	3.1%	30.2%	1.1%	1.0%	33.7%
SDR0.90	74.1%	73.5%	2.0%	2.0%	30.1%	0.8%	0.8%	32.3%
SDR	69.6%	70.4%	1.6%	3.0%	34.0%	1.3%	1.5%	33.8%

του συστήματος τότε η επιλογή του A θα πρέπει να είναι περίπου 0.80 αλλά αν μας ενδιαφέρει να μην χειροτερεύσει η απόδοση για καμία από τις κλάσεις τότε η καλύτερη επιλογή του A είναι ένα ποσοστό ανάμεσα στο 0.30 και το 0.40.

Στο σύνολο τους σχεδόν οι μεταβολές στην απόφαση δρομολόγησης και η τελική τους δρομολόγηση με βάση την απόφαση του JSQ είναι μεταβολές στην απόφαση δοσοληψιών που ανήκουν σε αυτές ακριβώς τις κλάσεις δοσοληψιών (πίνακας 5.11).

Οι ίδιες και πάλι παρατηρήσεις γίνονται και για την περίπτωση του πειράματος με την προσθήκη του κοινόχρηστου ενταμιευτή. Όπως και στο απλό παράδειγμα η βελτίωση είναι πιο μικρή σε σχέση με αυτήν του SD πειράματος καθώς ο JSQ επωφελείται σημαντικά από την παρουσία του κοινόχρηστου buffer. Είναι χαρακτηριστικό ότι διατηρώντας το μέγεθος του τοπικού ενταμιευτή σταθερό, υπάρχει μία βελτίωση στην απόδοση του συστήματος της τάξης του 15%. Υπάρχει δηλαδή σημαντικό όφελος από την παρουσία του κοινόχρηστου ενταμιευτή καθώς εκτός του ότι μεγάλο ποσοστό από τις σελίδες βρίσκονται εκεί οπότε αποφεύγετε η πρόσβαση στην περιφερειακή μνήμη, οι δοσοληψίες επισπεύδουν το commit καθώς αρκεί να μεταφερθούν οι σελίδες που άλλαξαν, στον κοινόχρηστο ενταμιευτή. Θυμίζουμε ότι είναι μη πτητική μνήμη οπότε σε περίπτωση βλάβης οι σελίδες αυτές που έχουν αλλαχτεί δεν χάνονται. Να σημειωθεί ότι δεν υπάρχει αντιστοιχία κόμβων και κλάσεων σε σχέση με το πείραμα με την SD αρχιτεκτονική διότι ο πίνακας δρομολόγησης 5.10 είναι διαφορετικός. Αυτό οφείλεται στο γεγονός ότι το monitor run έδωσε ελαφρά διαφορετικές μετρήσεις σε κάθε κόμβο για τις παραμέτρους του συστήματος σε σχέση με αυτές που συλλέγθησαν στο αντίστοιχο monitor run για την SD αρχιτεκτονική.

Στον πίνακα 5.15 φαίνεται το overall response time για κάθε πείραμα μαζί με το ποσοστό των δοσοληψιών που η τελική απόφαση του αλγορίθμου βασίστηκε σε αυτήν του JSQ

Πίνακας 5.15: Διάφορες μετρήσεις για το σύστημα του πειράματος 2 (SIM αρχιτεκτονική).

Αλγόριθμος	*Overall Tx Response Time*	*% of Tx Routed according JSQ*
JSQ	1.12522	100.0%
SDR0.10	1.01413	14.461%
SDR0.15	1.00252	10.990%
SDR0.20	1.00148	8.110%
SDR0.25	0.99659	6.950%
SDR0.30	0.99853	5.385%
SDR0.40	1.00079	4.364%
SDR0.50	0.99965	3.093%
SDR0.60	1.00273	1.765%
SDR0.70	1.00609	0.863%
SDR0.80	1.01014	0.206%
SDR0.90	1.01010	0.013%
SDR	1.01167	0.0%

Πίνακας 5.16: Μετρήσεις για το buffer hit ratio για το σύστημα του πειράματος 2 (SIM αρχιτεκτονική).

Αλγόριθμος	*Κόμβος*						**Κοιν. ενταμ.**
	1	2	3	4	5	6	
JSQ	7.54%	7.32%	7.53%	7.42%	7.36%	7.46%	55.00%
SDR0.10	18.12%	15.10%	22.36%	28.33%	25.52%	13.51%	50.35%
SDR0.15	19.68%	15.64%	23.92%	29.54%	27.39%	14.85%	49.94%
SDR0.20	20.67%	17.02%	25.07%	30.48%	28.42%	15.42%	49.39%
SDR0.25	21.59%	17.04%	26.30%	31.02%	29.13%	15.97%	49.08%
SDR0.30	22.41%	17.51%	26.81%	31.24%	30.09%	16.37%	48.72%
SDR0.40	23.21%	17.71%	27.16%	32.42%	30.63%	17.56%	48.29%
SDR0.50	23.59%	18.02%	28.17%	33.42%	30.76%	18.45%	47.95%
SDR0.60	24.79%	18.23%	28.88%	35.15%	30.87%	19.16%	47.49%
SDR0.70	26.03%	18.63%	29.31%	35.43%	31.41%	20.57%	46.88%
SDR0.80	26.21%	18.84%	29.70%	36.05%	31.58%	20.58%	46.83%
SDR0.90	26.53%	18.56%	29.95%	36.67%	31.90%	20.98%	46.74%
SDR	26.40%	18.74%	29.68%	36.36%	31.46%	21.09%	46.71%

τμήματος. Στον πίνακα 5.16 έχουμε το buffer hit ratio για κάθε ενταμιευτή κόμβου αλλά και αυτό του κοινόχρηστου ενταμιευτή ενώ στους πίνακες 5.17 και 5.18 μπορεί κανείς να δει το ποσοστό των δοσοληψιών κάθε κλάσης που δεν κατάφερε να πιάσει τον στόχο επίδοσης που είχε η αντίστοιχη κλάση.

Τα σχήματα 5.2 και 5.3 παριστάνουν γραφικά την κατανομή των 16 κλάσεων δοσοληψιών στους 6 κόμβους του συστήματος για δύο από τα πειράματα που έγιναν με την SIM αρχιτεκτονική. Στο πρώτο σχήμα 5.2 (η κλίμακα στον z άξονα αντιπροσωπεύει πλήθος δοσοληψιών και έχει περιοριστεί τεχνητά στις 2000 ώστε να φαίνονται καλύτερα και οι πιο μικρές τιμές, ενώ το σύνολο των δοσοληψιών κάθε κλάσης για τις οποίες έγινε η δρομολόγησή τους στο σύστημα ήταν περίπου 12500) μπορεί κανείς να δει καθαρά ότι οι δοσοληψίες του κόμβου 4 σε μεγάλο ποσοστό δρομολογούνται κάπου αλλού διότι ο ”κόμβος” τους έχει πολύ μεγάλο αριθμό από ενεργές δοσοληψίες. Το μεγαλύτερο ποσοστό από αυτές (όπως και για κάθε άλλη κλάση) πηγαίνει στους κόμβους 1 και 6 οι οποίοι είναι αυτοί που έχουν ως επί το πλείστον μικρότερο αριθμό από δοσοληψίες. Αντίθετα ο κόμβος 5 που είναι και αυτός υπερφορτωμένος παίρνει ένα πολύ μικρό ποσοστό από αυτές. Οι κλάσεις 6, 7, 13 και 14 είναι αυτές που κατά κύριο λόγο ευθύνονται για το ποσοστό των δοσοληψιών που την δρομολόγηση έκανε ο JSQ. Αυτό φαίνεται κυρίως από το σχήμα 5.3 όπου ο αλγόριθμος που χρησιμοποιήθηκε ήταν ο SDR0.50 και μόνο δοσοληψίες αυτών των κλάσεων δρομολογούνται σε κόμβο διαφορετικό από αυτό που ορίζει ο στατικός SDR. Αν και στο ασπρόμαυρο σχήμα δεν φαίνεται καλά (μαύρη κουκίδα στο επίπεδο xy του σχήματος) καμιά δοσοληψία άλλης κλάσης δεν δρομολογείται βάση της απόφασης του JSQ.

Η πιθανότητα εύρεσης μίας σελίδας στον κοινόχρηστο ενταμιευτή μειώνεται όταν οι αποφάσεις πλησιάζουν ολοένα την απόφαση του στατικού SDR, αλλά αυτό δεν πρέπει να παραξενεύει διότι με μεγάλη πιθανότητα βρίσκονται στον τοπικό ενταμιευτή οπότε το ποσοστό των αιτήσεων για εύρεση στον κοινόχρηστο είναι πιο μικρό και συνήθως αφορά σελίδες του COLD αρχείου (είναι και μεγαλύτερο σε μέγεθος) που η προσπέλαση σε αυτό έχει μικρή πιθανότητα.

5.5 Κριτική αλγορίθμου δρομολόγησης

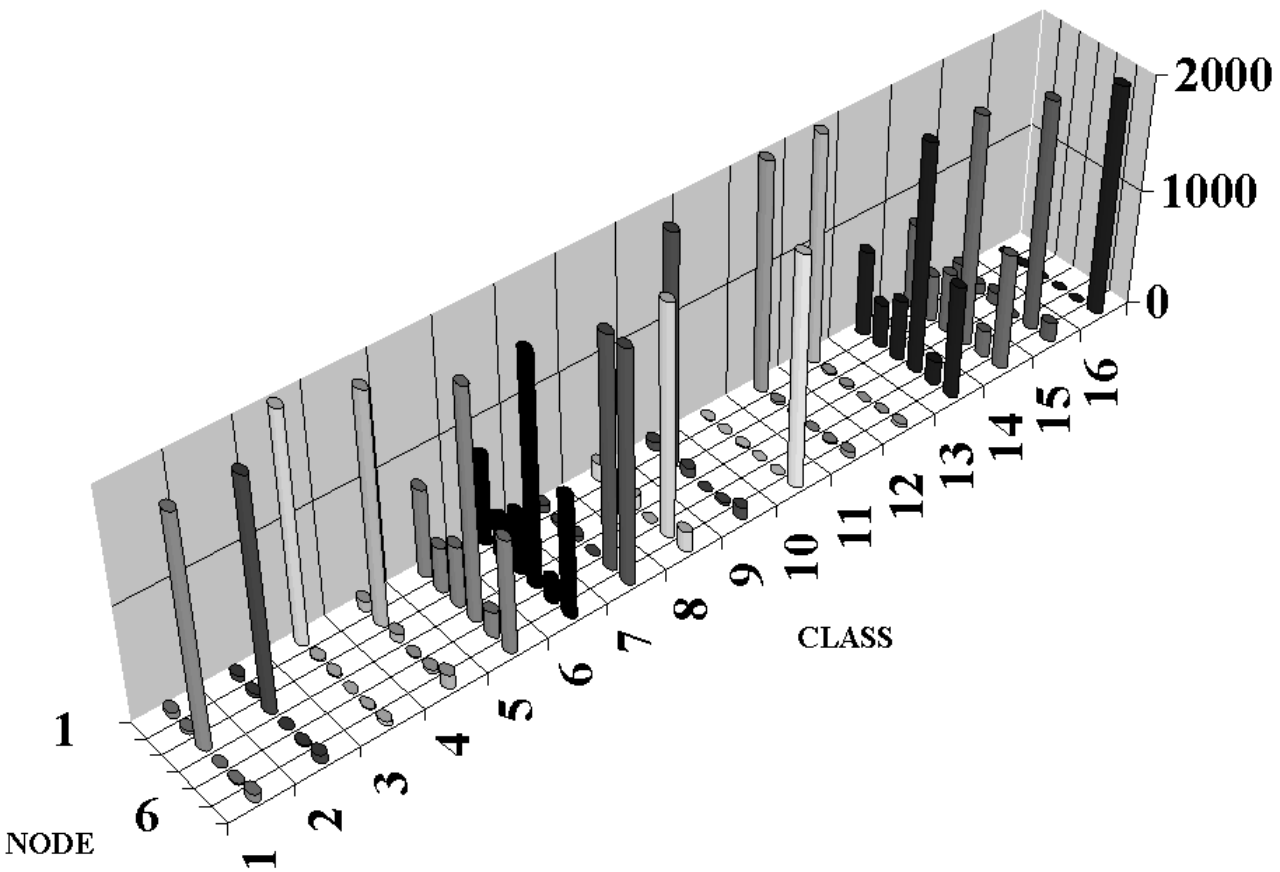
Η εφαρμογή του αλγορίθμου που προτείνουμε έχει πολύ καλά αποτελέσματα. Όσο πιο πολύ μεγάλο είναι το κόστος πρόσβασης στην περιφεριακή μνήμη τόσο καλύτερα συμπεριφέρεται το σύστημα με την χρήση του SDR. Αν και το κόστος για να αποφασιστεί ο πίνακας δρομολόγησης είναι αρκετά μεγάλο εντούτοις, η εφαρμογή του αρκεί να γίνει μόνο μία φορά. Παράλληλα προτείναμε έναν τρόπο μετατροπής του σε δυναμικό αλγόριθμο ώστε να μπορεί ο αλγόριθμος να αντιδρά καλύτερα σε περιπτώσεις όπου ένας κόμβος υπερφορτώνεται από μία παροδική αύξηση του φόρτου που επιφέρει στο σύστημα μία κλάση δοσοληψιών. Αυτή η λύση οδήγησε και στη μερική αντιμετώπιση του προβλήματος που παρουσιάζει ο αλγόριθμος δρομολόγησης. Συγκεκριμένα επιτρέπει μεγαλύτερες διαφορές στην χρήση των κόμβων από αυτές που θα ήταν ιδανικές διότι δεν λαμβάνει υπόψη του ορισμένες από τις παραμέτρους που καθυστερούν την εκτέλεση μίας δοσοληψίας, όπως είναι το data contention το οποίο εντείνεται όταν υπάρχουν πολλές ενεργές δοσοληψίες

Πίνακας 5.17: Μετρήσεις για το ποσοστό των δοσοληψιών που δεν έπιασαν τον στόχο επίδοσής τους (ανά κλάση) για το σύστημα του πειράματος 2 (SIM αρχιτεκτονική) I.

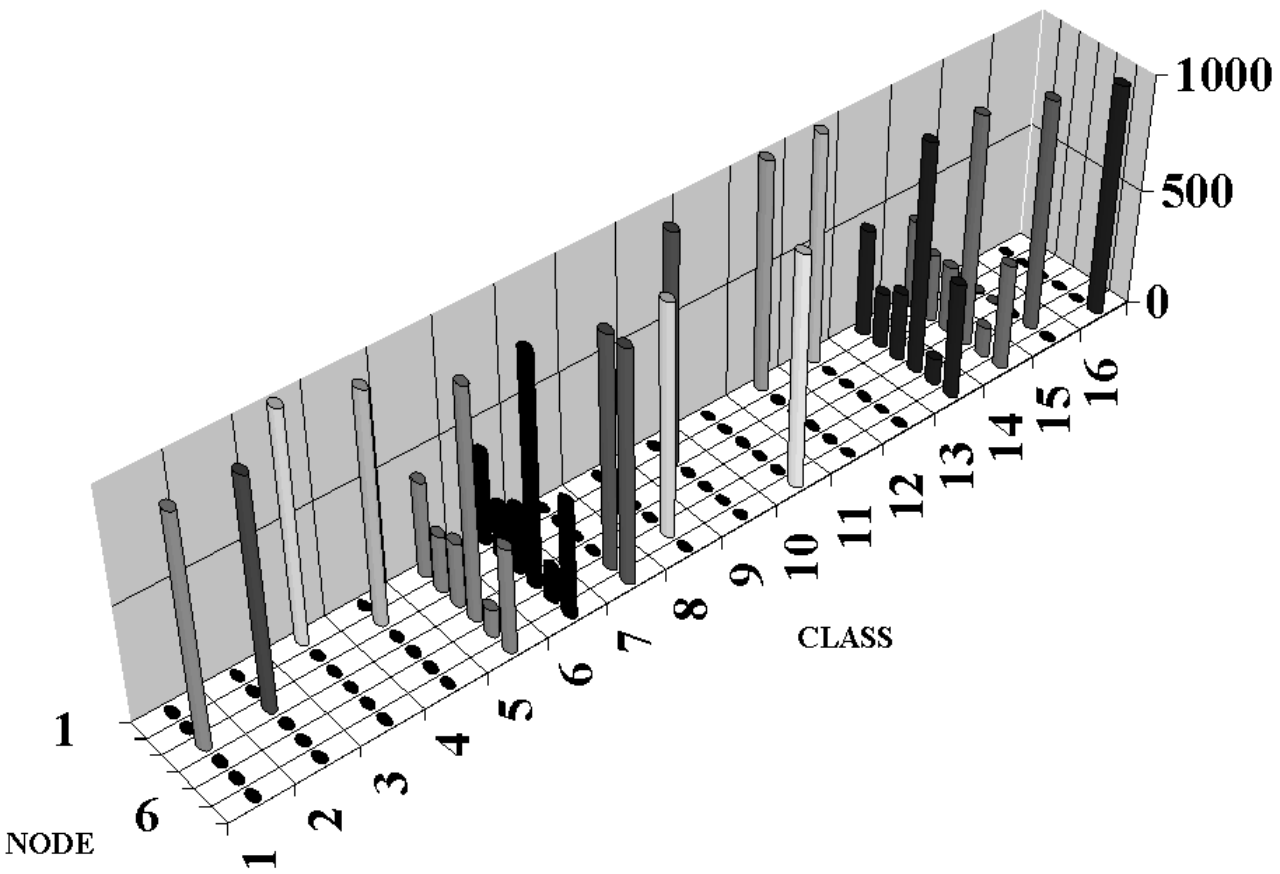
Αλγόριθμος	1	2	3	4	5	6	7	8
JSQ	68.5%	67.3%	68.8%	67.1%	68.3%	66.7%	68.6%	67.2%
SDR0.10	60.7%	59.5%	56.7%	64.4%	65.0%	63.2%	58.0%	62.1%
SDR0.15	66.1%	57.1%	50.0%	62.0%	65.2%	64.1%	53.1%	59.7%
SDR0.20	52.3%	52.8%	45.3%	60.5%	66.0%	66.6%	46.7%	59.5%
SDR0.25	48.0%	47.9%	38.6%	57.8%	68.4%	67.4%	43.0%	59.2%
SDR0.30	46.3%	46.2%	34.1%	55.7%	70.5%	69.9%	37.8%	58.2%
SDR0.40	37.3%	37.1%	27.0%	49.3%	77.0%	75.1%	31.3%	53.8%
SDR0.50	31.4%	30.9%	21.1%	42.5%	83.0%	79.6%	26.0%	47.9%
SDR0.60	25.6%	25.7%	17.5%	36.6%	88.3%	83.6%	22.2%	41.4%
SDR0.70	21.3%	21.3%	13.2%	31.7%	91.6%	87.2%	18.8%	35.4%
SDR0.80	19.2%	19.8%	11.8%	28.3%	94.9%	90.6%	17.2%	30.7%
SDR0.90	20.4%	21.2%	12.6%	28.3%	95.0%	90.4%	17.0%	32.3%
SDR	19.6%	20.0%	12.0%	28.8%	95.4%	90.4%	17.1%	33.3%

Πίνακας 5.18: Μετρήσεις για το ποσοστό των δοσοληψιών που δεν έπιασαν τον στόχο επίδοσής τους (ανά κλάση) για το σύστημα του πειράματος 2 (SIM αρχιτεκτονική) II.

Αλγόριθμος	9	10	11	12	13	14	15	16
JSQ	70.3%	70.0%	70.4%	70.8%	70.7%	69.7%	69.1%	75.2%
SDR0.10	66.8%	54.8%	57.3%	57.8%	65.9%	65.2%	64.7%	61.0%
SDR0.15	64.1%	48.0%	52.1%	52.2%	66.6%	66.8%	62.5%	51.7%
SDR0.20	63.0%	39.0%	46.6%	46.2%	67.9%	68.0%	61.3%	43.6%
SDR0.25	59.9%	31.9%	40.6%	39.7%	70.3%	70.5%	62.9%	36.3%
SDR0.30	58.2%	25.5%	35.6%	35.4%	71.9%	72.4%	61.2%	29.2%
SDR0.40	51.6%	18.9%	27.7%	28.3%	77.8%	77.6%	56.3%	21.0%
SDR0.50	43.1%	13.5%	21.6%	21.9%	82.7%	83.8%	49.8%	15.6%
SDR0.60	38.8%	11.9%	17.9%	18.1%	88.9%	88.7%	42.3%	13.2%
SDR0.70	33.4%	8.4%	14.3%	14.4%	93.0%	93.1%	38.0%	9.9%
SDR0.80	29.9%	8.1%	12.1%	12.7%	95.5%	95.2%	33.4%	9.2%
SDR0.90	28.4%	7.6%	13.5%	12.9%	95.9%	96.1%	33.4%	8.9%
SDR	31.3%	7.8%	12.8%	12.8%	95.8%	95.9%	33.8%	9.0%



Σχήμα 5.2: Η δρομολόγηση των δοσοληψιών για το πείραμα με τον SDR0.25 και την SIM αρχιτεκτονική.



Σχήμα 5.3: Η δρομολόγησης των δοσοληψιών για το πείραμα με τον SDR0.50 και την SIM αρχιτεκτονική.

στο σύστημα, η καθυστέρηση στην ουρά η οποία παρουσία πολλών δοσοληψιών, γίνεται μεγαλύτερη και τέλος δεν λαμβάνεται υπόψη ο ανταγωνισμός για τους κοινόχρηστους πόρους του συστήματος. Αυτό το τελευταίο μπορεί να αποτελέσει σημαντικό παράγοντα καθυστέρησης του όλου συστήματος.

Σε γενικές γραμμές ο αλγόριθμος έχει πολύ καλή συμπεριφορά και οδηγεί σε αποτελέσματα πάντα καλύτερα από αυτά του JSQ αλγορίθμου δρομολόγησης. Ιδιαίτερα οι δυναμικές εκδόσεις του μπορούν να συνδυαστούν με πολύ καλή βελτίωση της συμπεριφοράς του συστήματος και παράλληλη βελτίωση του ρυθμού που οι δοσοληψίες ικανοποιούν τον στόχο επίδοσης που έχει τεθεί για την κάθε κλάση. Η επιλογή του ποσοστού στον δυναμικό αλγόριθμο δεν είναι κάτι που μπορεί να γίνει εκ των προτέρων. Απαιτεί από κάποιον να γνωρίζει κατά πόσο η απόφαση του αλγορίθμου δρομολόγησης οδηγεί σε κόμβους με περίπου ίση χρήση. Αν αυτό συμβαίνει τότε η παράμετρος είναι κοντά στο 0.90 αλλιώς η παράμετρος μπορεί να ποικίλει αρκετά. Επίσης η επιλογή της παραμέτρου μπορεί να γίνει και βάση της απαίτησης ο ρυθμός που οι δοσοληψίες των διαφόρων κλάσεων ικανοποιούν τους στόχους επίδοσης, να μην είναι χειρότερος από αυτόν στην περίπτωση του JSQ. Σε ένα πραγματικό σύστημα ξεκινώντας από ένα ποσοστό κοντά στο 0.70 μπορεί να μετρηθεί για ένα χρονικό διάστημα (π.χ. μίας εβδομάδας) η επίδοση του συστήματος και στη συνέχεια με βάσει τις πραγματικές μετρήσεις που θα συλλεγθούν, να γίνει και πάλι ένας επανυπολογισμός του πίνακα δρομολόγησης.

Ο αλγόριθμος μπορεί να μετατραπεί ώστε να λαμβάνει υπόψη του και βάρη για τις αντίστοιχες κλάσεις δοσοληψιών ώστε με αυτόν τον τρόπο να δίνεται προτεραιότητα σε ορισμένες κλάσεις. Η χρήση αυτών των βαρών είναι δυνατό να γίνει κατά την σύγκριση των δύο διανυσμάτων με τους δείκτες επίδοσης, ώστε η επίδοση ορισμένων κλάσεων να θεωρηθεί πιο ' κρίσιμη ' . Τέλος θα μπορούσαμε να πούμε ότι είναι δυνατό να χρησιμοποιηθεί και για την επιλογή του τρόπου επέκτασης του συστήματος. Έτσι με δεδομένη την πολύ μεγάλη πιθανότητα εύρεσης μίας σελίδας στον τοπικό ενταμιευτή (σε σχέση με τον JSQ), θα μπορούσε ο διαχειριστής του συστήματος να προτείνει την αύξηση της ταχύτητας του επεξεργαστή στους κόμβους που στον SDR εμφανίζονται υπερφορτωμένοι, ώστε να μειωθεί ο αριθμός των δοσοληψιών που δρομολογούνται βάση του JSQ και συνεπώς να επιτευχθεί το μέγιστο δυνατό κέρδος από την επέκταση αυτή.

Κεφάλαιο 6

Επίδραση της ομαδοποίησης δοσοληψιών στην δρομολόγηση δοσοληψιών

Σε αυτό το κεφάλαιο θα δούμε τα αποτελέσματα της επίδρασης που έχει η ομαδοποίηση δοσοληψιών στην απόδοση ορισμένων αλγορίθμων δρομολόγησης. Πριν όμως περάσουμε στα αποτελέσματα θα πρέπει να αναφερθούμε σε βασικές έννοιες αλλά και στο εργαλείο που χρησιμοποιήθηκε προκειμένου να γίνει η ομαδοποίηση των δοσοληψιών. Καταρχήν η αρχιτεκτονική η οποία μελετάται σε αυτό το κεφάλαιο είναι η αρχιτεκτονική *Shared Nothing* [Μαρ95b], [YD94a]. Θυμίζουμε ότι σε αυτήν την αρχιτεκτονική κάθε κόμβος έχει στον τοπικό του δίσκο ένα τμήμα της βάσης δεδομένων. Συνεπώς ο αλγόριθμος του προηγούμενου κεφαλαίου δεν είναι δυνατό να εφαρμοστεί, διότι προϋποθέτει ότι ένας κόμβος έχει πρόσβαση σε ολόκληρη την βάση δεδομένων.

6.1 Ομαδοποίηση φόρτου εργασίας

Η γνώση των ιδιαίτερων (intrinsic) χαρακτηριστικών του φόρτου εργασίας που καλείται να εξυπηρετήσει ένα σύστημα, είναι απαραίτητη για τους αλγορίθμους ελέγχου του φόρτου εργασίας που είναι προσανατολισμένοι στους στόχους επίδοσης, ώστε η χρήση τους στα συστήματα επεξεργασίας δοσοληψιών να συντελέσει στη βελτίωση της απόδοσης των εν λόγω συστημάτων. Τα ιδιαίτερα αυτά χαρακτηριστικά σχετίζονται με τον μέσο αριθμό από προσπελάσεις στην βάση δεδομένων, με τα αρχεία στα οποία γίνονται αυτές οι προσπελάσεις, με την πιθανότητα η προσπέλαση αυτή να είναι για εγγραφή, με τις ανάγκες για CPU και πάντως δεν σχετίζονται με ρυθμούς άφιξης για τις μονάδες φόρτου.

Τα μεγάλα συστήματα επεξεργασίας δοσοληψιών χρησιμοποιούν έναν αριθμό από εκ των προτέρων ορισμένα προγράμματα, τα οποία αναφέρονται στη διεθνή βιβλιογραφία και ως "canned" δοσοληψίες. Με αυτές μοντελοποιούνται συγκεκριμένες ενέργειες του συστήματος όπως π.χ. στο τραπεζικό σύστημα η δοσοληψία - πρόγραμμα πίστωσης / χρέωσης (debit/credit).

Είναι δηλαδή συγκεκριμένες οι δυνατότητες αλληλεπίδρασης χρήστη και συστήματος.

Στις περισσότερες μεγάλες εφαρμογές, η γεωγραφική και οργανωτική δομή της εταιρείας επηρεάζει σημαντικά τον τρόπο και τόπο που αποθηκεύονται τα δεδομένα. Έτσι οι χρήστες π.χ. πελάτες μίας τράπεζας, χρησιμοποιούν ως επί το πλείστον συγκεκριμένα τερματικά π.χ. του υποκαταστήματος της τράπεζάς τους που βρίσκεται στην γειτονιά τους, προκειμένου να έχουν πρόσβαση στα δεδομένα (τραπεζικό λογαριασμό τους).

Ο συνδυασμός του ονόματος του προγράμματος της δοσοληψίας με την ταυτότητα του χρήστη (μοναδικό ID) καθώς και το όνομα του τερματικού, αποτελούν ένα μοναδικό αναγνωριστικό μίας δοσοληψίας, που ονομάζεται triplet ID (ταυτότητα τριπλέτας). Στις περισσότερες περιπτώσεις ο συνδυασμός αυτός αντιπροσωπεύει κοινά χαρακτηριστικά ως προς τις απαιτήσεις επεξεργασίας. Συνδυάζοντας τα στατιστικά δεδομένα που σχετίζονται με τα ιδιαίτερα χαρακτηριστικά και το triplet ID έχουμε την λεγόμενη απλά triplet (τριπλέτα).

Συχνά ο αριθμός των τριπλετών ξεπερνά τις 10000 στα σύγχρονα συστήματα. Αν και τα τερματικά ενός συστήματος είναι συχνά περισσότερα από 100000 και τα canned προγράμματα δοσοληψιών μπορεί να ξεπερνούν τα 1000, ο αριθμός των τριπλετών που εμφανίζονται δεν φτάνει το γινόμενό τους καθώς πολλοί από τους συνδυασμούς δεν εμφανίζονται στο σύστημα, π.χ δύσκολα ένας πελάτης μίας τράπεζας θα εμφανιστεί στα ATM της τράπεζας που βρίσκονται σε άλλες πόλεις της Ελλάδας από αυτήν που κατοικεί.

Πάντως ακόμα και αυτή η τάξη μεγέθους αυτού του αριθμού από τριπλέτες είναι απαγορευτική για τους αλγορίθμους δρομολόγησης που χρησιμοποιούνται για την βελτίωση της επίδοσης των συστημάτων επεξεργασίας δοσοληψιών.

Θα πρέπει λοιπόν δοθεισών αυτών των τριπλετών, να μειωθεί η ποσότητα της πληροφορίας, χωρίς όμως να χαθούν αυτά τα ιδιαίτερα χαρακτηριστικά τους. Γι' αυτό το λόγο θα πρέπει οι τριπλέτες με παρόμοια χαρακτηριστικά τόσο από άποψη δεδομένων στα οποία κάνουν προσπέλαση, όσο και από άποψη απαιτούμενων πόρων, να ανατεθούν στην ίδια ομάδα - κλάση. Η διαδικασία αυτή ονομάζεται ομαδοποίηση του φόρτου εργασίας (workload clustering) [Λαμ95a], [Κλο97].

Συνήθως γίνεται από κάποιον ειδικό καθώς βασίζεται αρκετά σε ευρηματικές μεθόδους αλλά και στην εμπειρία του ατόμου που κάνει αυτή την εργασία. Ενώ όμως για μικρά συστήματα μπορεί η διαδικασία αυτή να γίνει από έναν ειδικό, στα μεγάλα συστήματα αυτό δεν είναι εφικτό και λόγω πλήθους δεδομένων για τα οποία πρέπει να γίνει η ομαδοποίηση, αλλά και λόγω μη προφανών αλληλεξαρτήσεων που υπάρχουν σε ένα τόσο μεγάλο σύστημα. Γι' αυτό σε τέτοια συστήματα απαιτείται η χρήση ενός αυτοματοποιημένου τρόπου ομαδοποίησης.

6.2 Ομαδοποίηση φόρτου εργασίας με χρήση του CLUE

Το **CLUE** [Λαμ95a], [Κλο97] είναι ένα περιβάλλον ομαδοποίησης για τον φόρτο εργασίας OnLine συστημάτων επεξεργασίας δοσοληψιών (OLTP). Περιλαμβάνει έναν αρκετά μεγάλο αριθμό από αλγόριθμους ομαδοποίησης δοσοληψιών. Για τα πειράματα που έγιναν

χρησιμοποιήθηκαν τρεις αλγόριθμοι ομαδοποίησης.

1. **Random** : Σύμφωνα με αυτόν τον αλγόριθμο, κάθε δεδομένο ανατίθεται τυχαία σε μία ομάδα. Έτσι τελικά κάθε ομάδα απαρτίζεται από τον ίδιο περίπου αριθμό από δεδομένα.

2. **K-Means** : Είναι ίσως ο πιο ευρύτατα διαδεδομένος αλγόριθμος ομαδοποίησης καθώς είναι αρκετά απλός και γρήγορος. Ξεκινώντας από τον ζητούμενο αριθμό κλάσεων - ομάδων ο αλγόριθμος κάνει μία σειρά από επαναλήψεις, όπου για κάθε δεδομένο υπολογίζεται η απόστασή του (π.χ. Ευκλείδια απόσταση) από το κέντρο κάθε ομάδας. Αφού υπολογιστούν αυτές οι αποστάσεις το δεδομένο ανατίθεται στην ομάδα για την οποία η απόσταση του κέντρου της ομάδας από το δεδομένο, είναι η ελάχιστη. Μετά από κάθε επανάληψη επανυπολογίζονται τα κέντρα των ομάδων. Όταν δεν υπάρχουν άλλες ανακατατάξεις τότε ο αλγόριθμος σταματά.

3. **HALC** : Ο HALC (Heuristic ALgorithm for Clustering) είναι ένας αλγόριθμος ομαδοποίησης που βασίζεται σε ευρηματικές μεθόδους και είναι κατάλληλος για την επεξεργασία και ομαδοποίηση μεγάλου πλήθους δεδομένων καθώς δίνει πολύ καλά αποτελέσματα και είναι γρήγορος. Αρχικά ο αλγόριθμος αναθέτει κάθε τριπλέτα σε διαφορετική ομάδα και στη συνέχεια βάση των σελίδων που κάνει προσπέλαση κάθε τριπλέτα αποφασίζει ποιες ομάδες θα πρέπει να ενώσει. Η διαδικασία αυτή συνεχίζεται μέχρι ο αριθμός από τις ομάδες να γίνει ίσος ή μικρότερος του ζητούμενου αριθμού ομάδων που έχει θέσει ο χρήστης.

Σε κάθε τριπλέτα αντιστοιχεί ένα διάνυσμα στο n-διάστατο χώρο, με κάθε διάσταση να αντιστοιχεί σε μία διαφορετική σελίδα της βάσης δεδομένων. Η απόσταση μίας τριπλέτας x από μία άλλη y δίνεται από την εξίσωση 6.1:

$$\alpha(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{6.1}$$

Ο αριθμός των προσπελάσεων της τριπλέτας x στην i σελίδα της βάσης που συμβολίζεται ως x_i στην εξίσωση, είναι το σταθμισμένο άθροισμα τον προσπελάσεων για ανάγνωση και για εγγραφή για την συγκεκριμένη τριπλέτα και για την συγκεκριμένη σελίδα της βάσης δεδομένων, δηλ. $x_i = \frac{w_{\text{read}} * \#\text{read accesses} + w_{\text{write}} * \#\text{write accesses}}{w_{\text{read}} + w_{\text{write}}}$. Τα βάρη w_{read} και w_{write} καθορίζονται από τον χρήστη και η προκαθορισμένη τιμή τους είναι 1 (όμοια και για το y_i).

Σε κάθε περίπτωση ο αναγνώστης που ενδιαφέρεται για περισσότερες πληροφορίες για αυτούς τους αλγορίθμους καλείται να μελετήσει τις μεταπτυχιακές εργασίες [Λαμ95a] και [Κλο97]. Το CLUE λαμβάνει υπόψη του μόνο τις σελίδες στις οποίες αναφέρεται μία τριπλέτα.

6.3 Αξιολόγηση ομαδοποίησης

Σε αυτή την παράγραφο θα δώσουμε τα αποτελέσματα της ομαδοποίησης για τους τρεις διαφορετικούς αλγορίθμους. Ως δεδομένα χρησιμοποιήθηκαν ίχνη ενός προσομοιωτή

Χαρακτηριστικά	*DOA*	*PULS*
IDs δοσοληψιών	103	53
IDs χρηστών	474	103
IDs τερματικών	474	110
Αριθμός από τριπλέτες	1984	280
Σελίδες βάσης δεδομένων	41003	66846
μη μηδενικά στοιχεία του πίνακα προσπελάσεων	180017	178749
πυκνότητα προσπελάσεων	0.22123%	0.95501%

Πίνακας 6.1: Χαρακτηριστικά των αρχείων ιχνών *DOA* και *PULS* .

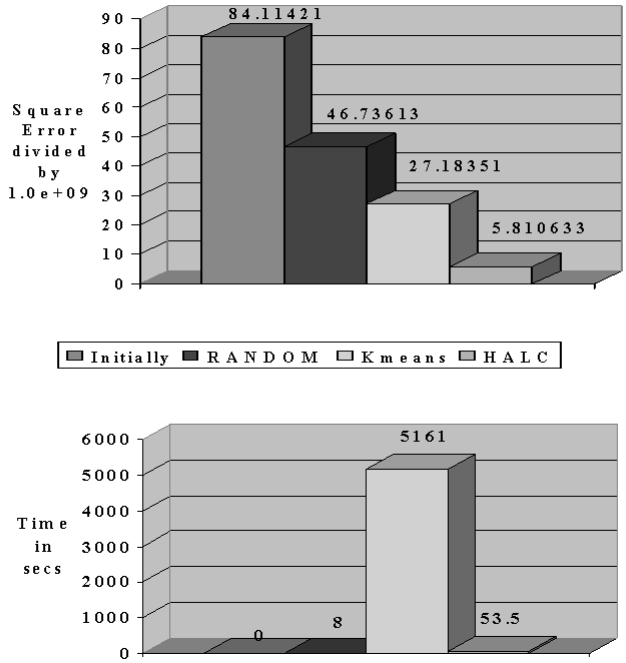
συστήματος σχεσιακής βάσης δεδομένων ο οποίος προσομοίωνε έναν και μόνο κόμβο. Τα δεδομένα αυτά μας παραχωρήθηκαν από την Siemens Nixdorf Informationssysteme AG Germany και αποτελούνταν από δύο αρχεία από ίχνη το DOA και το PULS. Τα βασικά χαρακτηριστικά αυτών των ιχνών συνοψίζονται στον πίνακα 6.1.

Ως πυκνότητα προσπελάσεων ορίζεται το κλάσμα :

$$\text{πυκνότητα προσπελάσεων} = \frac{\text{πλήθος μη μηδενικών προσπελάσεων}}{\text{πλήθος τριπλετών} \times \text{πλήθος σελίδων δεδομένων}}$$

Εφόσον δεν υπήρχε πληροφορία για την πιθανότητα άφιξης κάθε τριπλέτας υποθέσαμε ότι αυτή δίνεται από την πιθανότητα εμφάνισής της στο αρχείο με τα ίχνη. Κάτι πολύ σημαντικό για αυτά τα αρχεία ιχνών είναι οι μετρήσεις που έγιναν και αφορούν την τοπικότητα στις προσπελάσεις των τριπλετών. Έτσι οι μετρήσεις έδειξαν ότι σε ένα πολύ μικρό αριθμό από σελίδες γίνεται ένα μεγάλο ποσοστό από τις συνολικές προσπελάσεις :

- Μετρήσεις για την τοπικότητα των κλήσεων για προσπέλαση στη βάση δεδομένων στο αρχείο ιχνών *DOA*
  - -- Οι 10 πιο "δημοφιλείς" σελίδες (ποσοστό 0.0243% επί του συνόλου των σελίδων) είναι αποδέκτες του 16.76% όλων των προσπελάσεων στη βάση δεδομένων.
  - -- Το 80% όλων των προσπελάσεων γίνεται στο 13.14% του συνόλου των σελίδων της βάσης δεδομένων.
  - -- Το 90% όλων των προσπελάσεων γίνεται στο 25.78% του συνόλου των σελίδων της βάσης δεδομένων.
- Μετρήσεις για την τοπικότητα των κλήσεων για προσπέλαση στη βάση δεδομένων στο αρχείο ιχνών *PULS*
  - -- Οι 10 πιο "δημοφιλείς" σελίδες (ποσοστό 0.015% επί του συνόλου των σελίδων) είναι αποδέκτες του 10.02% όλων των προσπελάσεων στη βάση δεδομένων..
  - -- Το 80% όλων των προσπελάσεων γίνεται στο 26.96% του συνόλου των σελίδων της βάσης δεδομένων.

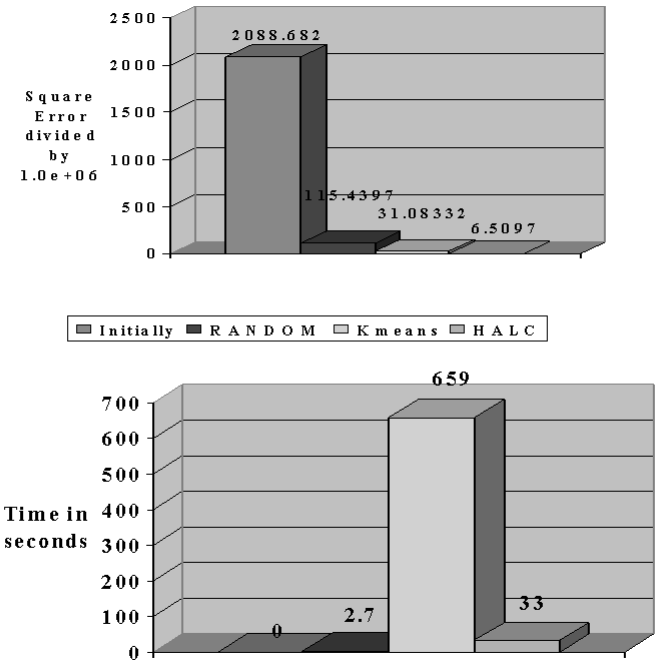


Σχήμα 6.1: Συγκριτικά αποτελέσματα μαζί με τους χρόνους εκτέλεσης για τους τρεις αλγορίθμους ομαδοποίησης και για το αρχείο DOA

-- Το 90% όλων των προσπελάσεων γίνεται στο 45.05% του συνόλου των σελίδων της βάσης δεδομένων.

Οι αλγόριθμοι κλήθηκαν να δημιουργήσουν 30 ομάδες. Τα αποτελέσματα της ομαδοποίησης φαίνονται στα σχήματα 6.1 και 6.2, καθώς και οι αντίστοιχοι χρόνοι εκτέλεσης για κάθε αλγόριθμο. Ως κριτήριο για την αξιολόγηση της ποιότητας ομαδοποίησης χρησιμοποιήθηκε το κριτήριο του τετραγωνικού σφάλματος. Έτσι το συνολικό σφάλμα προκύπτει από το άθροισμα πάνω σε όλες τις ομάδες, του τετραγώνου της απόστασης κάθε δεδομένου από το κέντρο της ομάδας στην οποία έχει ανατεθεί από τον αλγόριθμο ομαδοποίησης.

Βλέπουμε λοιπόν ότι ο Random αλγοριθμος ομαδοποίησης αν και είναι πολύ γρήγορος (αφού κάνει τυχαία ομαδοποίηση) επιτυγχάνει το χειρότερο τετραγωνικό σφάλμα. Καλύτερος και στις δύο περιπτώσεις και μάλιστα αρκετά γρήγορος είναι ο αλγόριθμος HALC. Ο K-Means παραπλανάται από αυτήν την τοπικότητα που έχουν οι τριπλέτες στις προσπελάσεις τους στη βάση δεδομένων και δημιουργεί μία τεράστια κλάση που περιλαμβάνει πάνω από το 90% των τριπλετών. Πολλές ομάδες στην περίπτωση του K-Means αποτελούνταν από μία και μόνο τριπλέτα.



Σχήμα 6.2: Συγκριτικά αποτελέσματα μαζί με τους χρόνους εκτέλεσης για τους τρεις αλγορίθμους ομαδοποίησης και για το αρχείο PULS

6.4 Επίπτωση της ομαδοποίησης του φόρτου εργασίας στην δρομολόγηση δοσοληψιών

Προκειμένου να αξιολογηθεί η επίπτωση της ομαδοποίησης του φόρτου εργασίας στην δρομολόγηση δοσοληψιών χρησιμοποιήθηκε ο προσομοιωτής συστημάτων επεξεργασίας δοσοληψιών **TPsim** και συγκεκριμένα η *Shared Nothing* αρχιτεκτονική [Μαρ95b].

Ένα βασικό ζήτημα που χρειάστηκε να αντιμετωπιστεί ήταν η ανάθεση των σελίδων της βάσης δεδομένων στους κόμβους του συστήματος. Αυτό συνέβει διότι τα ίχνη προέρχονταν από ένα σύστημα με έναν κόμβο οπότε όλα τα δεδομένα ήταν τοπικά. Προκειμένου να λυθεί αυτό το ζήτημα επιλέχθηκε να ανατεθούν οι σελίδες στους κόμβους με τέτοιο τρόπο, ώστε ο αριθμός προσπελάσεων σε κάθε κόμβο να είναι για κάθε κόμβο περίπου ο ίδιος.

Από τα αρχικά πειράματα φάνηκε ότι τα αποτελέσματα επηρεάζονταν σε ένα πολύ μεγάλο βαθμό από την τύχη ορισμένων δοσοληψιών που έκαναν έναν αριθμό από χιλιάδες προσβάσεις στην βάση δεδομένων. Πιο συγκεκριμένα υπήρχε ένα ποσοστό από δοσοληψίες (τριπλέτες στα αρχεία ιχνών) όπου έκαναν έναν δυσανάλογο αριθμό από κλήσεις στη βάση δεδομένων σε σχέση με τον μέσο όρο κλήσεων του κάθε αρχείου ιχνών. Η παρουσία τους οδηγούσε το σύστημα σε έναν υπερβολικά μεγάλο αριθμό από timeouts και αδιέξοδα ανάμεσα στις δοσοληψίες που οδηγούσαν με την σειρά τους σε μικρή χρήση των επεξεργαστών του συστήματος. Το πότε θα γινόταν kill αυτές οι (long-lived) δοσοληψίες μετά από αδιέξοδο (επιλέγεται τυχαία μία δοσοληψία από αυτές που εμπλέκονται στο αδιέξοδο και γίνεται kill), επηρέαζε σημαντικά την όλη απόδοση του συστήματος. Έτσι προκειμένου να λυθεί

αυτό το πρόβλημα δημιουργήθηκαν δύο νέα αρχεία ιχνών που περιείχαν δοσοληψίες που έκαναν προσπέλαση το πολύ σε 30 διαφορετικές σελίδες. Αυτό δεν περιόριζε τον αριθμό από προσπελάσεις κάθε δοσοληψίας στη βάση δεδομένων, καθώς ακόμα και με αυτόν τον τρόπο οι δοσοληψίες εμφάνιζαν αριθμό από προσπελάσεις άνω των 100 πάντα όμως σε 30 το πολύ διαφορετικές σελίδες της βάσης.

Τα νέα αυτά αρχεία θα ονομάζονται από εδώ και στο εξής *Reduced DOA* και *Reduced PULS* και αποτελούνται από 1836 τριπλέτες δηλ. 92.5% του αρχικού αρχείου ιχνών DOA για το *Reduced DOA* , και από 144 τριπλέτες δηλ. 51.4% σε σχέση με το αρχικό αρχείο ιχνών PULS, για το *Reduced PULS* . Στο *Reduced DOA* αρχείο ιχνών ο μέσος αριθμός από τις κλήσεις στη βάση δεδομένων για όλες τις δοσοληψίες είναι περίπου 27, ενώ για το *Reduced PULS* αυτός ο αριθμός είναι περίπου 55 κλήσεις κατά μέσο όρο.

Τρεις αλγόριθμοι δρομολόγησης χρησιμοποιήθηκαν. Ο Join Shortest Queue (JSQ) ο οποίος δεν λαμβάνει υπόψη του την συγγένεια μίας δοσοληψίας με κάποια δεδομένα, ο WFW (Wait-For-Work) ο οποίος βασίζεται στον JSQ αλλά λαμβάνει υπόψη του την συγγένεια αυτή και τέλος ο WFWC ο οποίος παράλληλα λαμβάνει υπόψη του την προτεραιότητα της νέας δοσοληψίας αλλά και αυτή των ήδη υπαρχόντων δοσοληψιών στο σύστημα. Περισσότερες πληροφορίες για αυτούς μπορεί κανείς να βρει στα [ESP], [Μαρ95b]. Το μόνο που θα πρέπει να προσθέσουμε είναι ότι οι δύο αλγόριθμοι WFW και WFWC, βασίζονται σε έναν αριθμό από παραμέτρους του συστήματος που καθορίζουν τις ανάγκες επεξεργασίας για κάθε κόμβο και κάθε κλάση και οι οποίες λαμβάνονται μετά από ένα monitor run όπως και στην περίπτωση του αλγορίθμου που περιγράφεται στο κεφάλαιο 5. Ο WFW και ο WFWC έχουν ως σκοπό την ελαχιστοποίηση του χρόνου απόκρισης για κάθε δοσοληψία που έρχεται στο σύστημα. Οι αλγόριθμοι αυτοί κλήθησαν να δρομολογήσουν τις δοσοληψίες που οι αλγόριθμοι ομαδοποίησης κατέταξαν σε μία από τις 30 κλάσεις-ομάδες. Με αυτόν τον τρόπο η ποιότητα της ομαδοποίησης αναμένουμε να έχει κάποια ορατή επίδραση στην απόδοση του συστήματος που μετράται με την χρήση του TPsim, ώστε να φανεί η επίδραση ενός καλού ή κακού αλγορίθμου ομαδοποίησης στις αποφάσεις που παίρνουν οι αλγόριθμοι δρομολόγησης.

Δύο διαφορετικά μοντέλα συστήματος χρησιμοποιήθηκαν. Για το μεν *Reduced DOA* αρχείο ιχνών, το σύστημα αποτελείτο από 7 κόμβους ενώ για το *Reduced PULS* από 5 κόμβους. Ένας κόμβος (front-end) σε κάθε περίπτωση αναλαμβάνει να κάνει την δρομολόγηση των δοσοληψιών στους υπόλοιπους κόμβους. Υπάρχουν 100 χρήστες στο κάθε σύστημα οι οποίοι στέλνουν μία αίτηση για εξυπηρέτηση και στη συνέχεια περιμένουν για ένα χρονικό διάστημα (think time) προτού στείλουν την επόμενη αίτηση τους, χωρίς προηγουμένως να πρέπει να λάβουν απάντηση στην προηγούμενη αίτησή τους (ανοικτό μοντέλο). Με αυτόν τον τρόπο οι ενεργές δοσοληψίες στο σύστημα δεν περιορίζονται από τον αριθμό των χρηστών αλλά είναι δυνατό το σύστημα να "κατακλυστεί" από αιτήσεις εάν δεν είναι σε θέση να μπορεί να εξυπηρετήσει τον ρυθμό άφιξης των νέων αιτήσεων στο σύστημα. Όλοι οι κόμβοι είναι όμοιοι αν και αρχικά έγιναν πειράματα και με συστήματα που αποτελούνταν από διαφορετικούς κόμβους. Οι βασικοί παράμετροι των πειραμάτων φαίνονται στον πίνακα 6.2. Προκειμένου να ληφθούν μετρήσεις για διαφορετικά υψηλά ποσοστά χρήσης των κόμβων, επιλέχθηκε να γίνεται μεταβολή της ταχύτητας των επεξεργαστών. Εξαιτίας της τοπικότητας στις προσπελάσεις των δοσοληψιών στη βάση δεδομένων, το ποσοστό των σελίδων για τις

Reduced DOA	
number of CPUs	1
MPL	400
number of disks	1
CPU speed	20 - 24 - 28
Private buffer	30% of local pages
Communication Network:	
packet size	1024 bytes
transfer rate	80Mbits
Workload	
Transaction Classes	30
Tx Appl. Burst	500000.0
Users	100
Think time	0.014
Performance Goal	0.5

Reduced PULS	
number of CPUs	1
MPL	400
number of disks	1
CPU speed	20 - 25 - 28
Private buffer	50% of local pages
Communication Network:	
packet size	1024 bytes
transfer rate	80Mbits
Workload	
Transaction Classes	30
Tx Appl.Burst	1000000.0
Users	100
Think time	0.038
Performance Goal	1.0

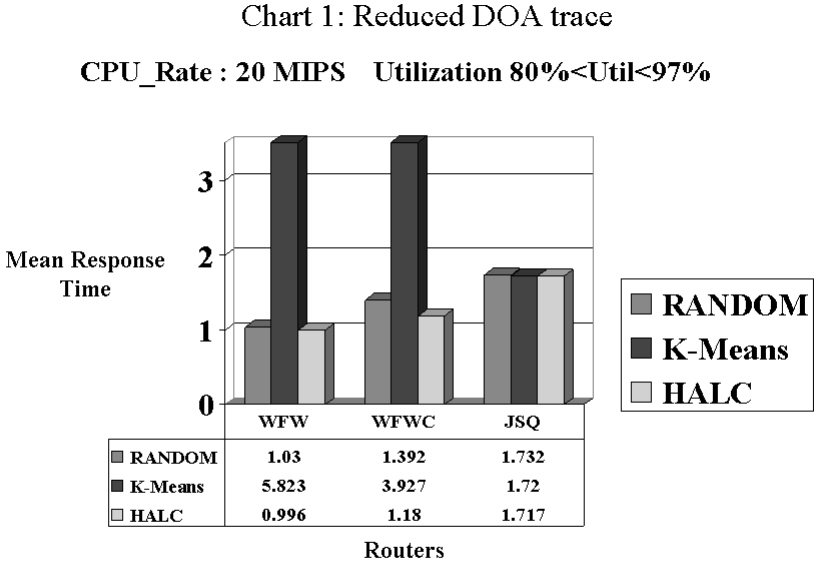
Πίνακας 6.2: Βασικές παράμετροι της προσομοίωσης που έγινε με τον προσομοιωτή TPsim.

οποίες αρκεί η προσπέλαση στους ενταμιευτές είναι πολύ μεγάλο και συνεπώς ο βαθμός χρήσης των μέσων της περιφεριακής μνήμης είναι πολύ μικρός (< 15%). Τέλος λόγω της μη εκ των προτέρων γνώσης της σύστασης κάθε κλάσεις δοσοληψιών, ο στόχος επίδοσης καθορίστηκε να είναι ο ίδιος για όλες τις κλάσεις.

6.5 Πειραματικά αποτελέσματα για το *Reduced DOA* αρχείο ιχνών

Τα αποτελέσματα με την επίδοση του συστήματος για κάθε αλγόριθμο δρομολόγησης και για κάθε αλγόριθμο ομαδοποίησης που χρησιμοποιήθηκε και για το αρχείο ιχνών *Reduced DOA* , φαίνονται στα σχήματα 6.3, 6.4, και 6.5. Μέτρο σύγκρισης είναι ο μέσος χρόνος απόκρισης για το σύνολο των κλάσεων του συστήματος. Στο σχήμα 6.3 η ταχύτητα του επεξεργαστή κάθε κόμβου είναι 20 MIPS (Million Instructions Per Second), για το σχήμα 6.4 24 MIPS, ενώ για το σχήμα 6.5 28 MIPS. Με την αύξηση της ταχύτητας του επεξεργαστή μειώνεται η χρήση του και συνεπώς βελτιώνεται η απόκριση του συστήματος καθώς ο πόρος είναι πολύ πιο συχνά διαθέσιμος για επεξεργασία.

Η απόδοση του JSQ αλγόριθμου δρομολόγησης αποτελεί και την βάση για την σύγκριση. Εφόσον ο αλγόριθμος αγνοεί την συγγένεια μίας κλάσης με τα δεδομένα ενός κόμβου (data affinity) και αγνοεί οποιαδήποτε ομαδοποίηση έχει γίνει (καθώς δεν μεταχειρίζεται καμία δοσοληψία βάση της κλάσης της), μας δίνει την επίδοση του συστήματος όταν δεν χρησιμοποιούμε τεχνικές ομαδοποίησης αλλά και δεν κάνουμε χρήση αλγορίθμων δρομολόγησης που λαμβάνουν υπόψη τους το data affinity. Θυμίζουμε ότι σύμφωνα με τον αλγόριθμο αυτό μία νέα δοσοληψία δρομολογείται στον κόμβο που έχει τον ελάχιστο αριθμό από ενεργές δοσοληψίες και άρα πιο μικρή ουρά. Έτσι από την μελέτη όλων των σχημάτων μπορεί κανείς να δει ότι με την χρήση του JSQ η απόδοση του συστήματος είναι η ίδια,

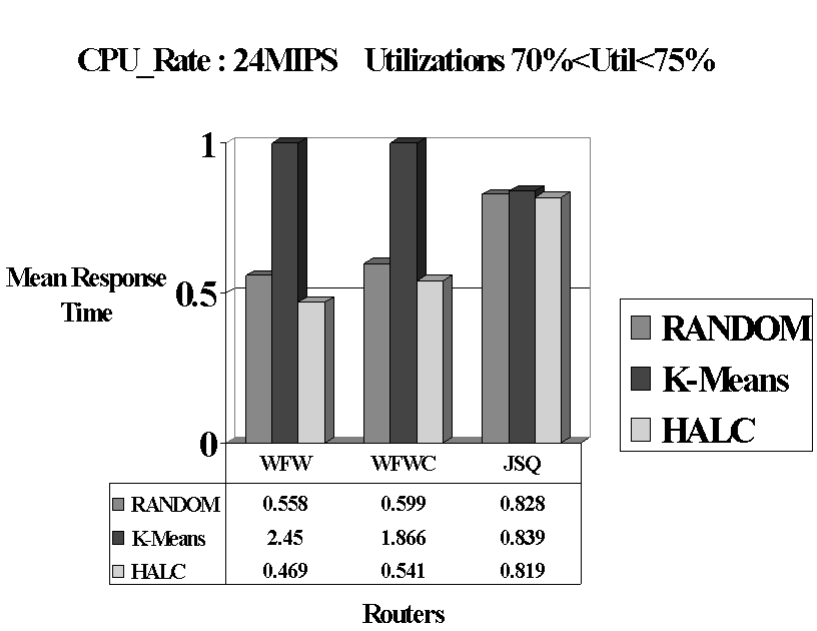


Σχήμα 6.3: Μέσος χρόνος απόκρισης για όλες τις κλάσεις στην περίπτωση του *Reduced DOA* αρχείο ιχνών και για ταχύτητα επεξεργαστών 20 MIPS.

ανεξάρτητα του αλγορίθμου ομαδοποίησης που έχει χρησιμοποιηθεί.

Αν και κάποιος θα περίμενε ότι ο βαθμός χρήσης όλων των κόμβων με την χρήση του JSQ θα είναι ο ίδιος εντούτοις τα πειράματα έδειξαν ότι ένας κόμβος (κόμβος 3) έχει αρκετά πιο μικρό βαθμό χρήσης σε σχέση με τους υπόλοιπους κόμβους. Αυτό συμβαίνει γιατί ο κόμβος αυτός συγκεντρώνει έναν μεγάλο αριθμό από εικονικές δοσοληψίες (mirror transactions). Στην αρχιτεκτονική *Shared Nothing* όταν μία δοσοληψία θέλει να κάνει προσπέλαση σε δεδομένα που βρίσκονται αποθηκευμένα σε έναν άλλο κόμβο τότε η αίτηση μεταβιβάζεται στον κόμβο αυτό και μία εικονική δοσοληψία δημιουργείται προκειμένου να προσπελάσει το ζητούμενο δεδομένο. Η δοσοληψία αυτή επιφέρει πολύ πιο μικρή επιβάρυνση στον κόμβο σε σχέση με την αρχική (primary) δοσοληψία, καθώς δεν επιβαρύνεται από το πρωτόκολλο δέσμευσης δύο φάσεων αλλά ούτε και από το σύνολο των μηνυμάτων που πρέπει να ανταλλάξει ο κόμβος που έχει την αρχική δοσοληψία. Αυτό έχει ως αποτέλεσμα ο κόμβος αυτός να έχει μεγάλο αριθμό από δοσοληψίες και άρα να μην επιλέγεται από τον JSQ για την δρομολόγηση μίας νέας δοσοληψίας που φτάνει στο σύστημα, αλλά παράλληλα και μικρό utilization διότι οι δοσοληψίες που είναι ενεργές στον κόμβο αυτό δεν επιβαρύνουν πολύ τον κόμβο.

Ο απόδοση του JSQ αλγορίθμου δρομολόγησης ήταν η χειρότερη, εκτός από τη περίπτωση όπου χρησιμοποιήσαμε ως αλγόριθμο ομαδοποίησης τον K-Means. Στην περίπτωση αυτή και αφού η ομαδοποίηση κατέληξε σε μία ομάδα με σχεδόν το 93,6% των triplets η δρομολόγησή τους σύμφωνα με τους αλγορίθμους WFW και WFWC κατέληγε σε χειρότερα αποτελέσματα από τα αποτελέσματα που είχε η δρομολόγηση αγνοώντας ουσιαστικά όλες τις πληροφορίες (JSQ αλγόριθμος). Εφόσον σχεδόν όλες οι δοσοληψίες ανήκαν σε μία

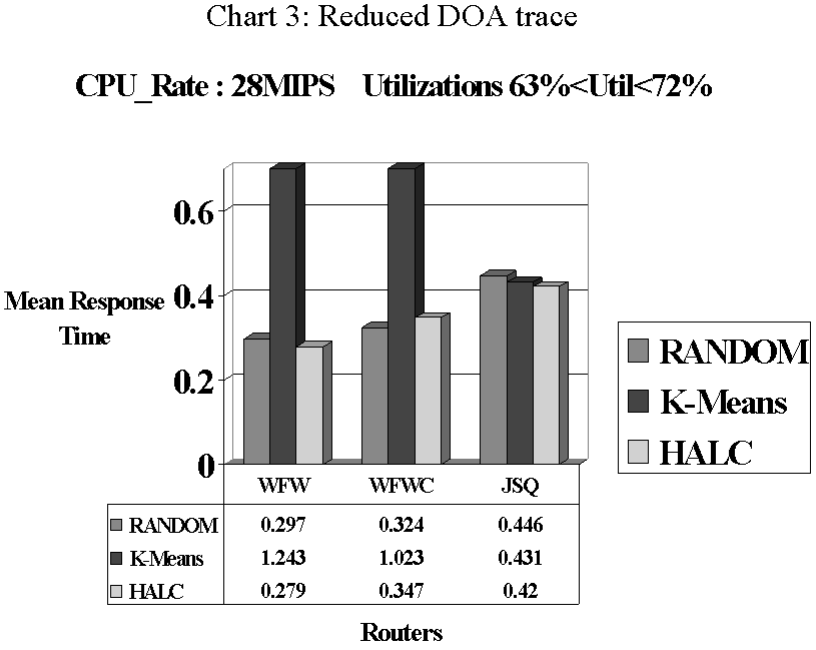


Σχήμα 6.4: Μέσος χρόνος απόκρισης για όλες τις κλάσεις στην περίπτωση του *Reduced DOA* αρχείο ιχνών και για ταχύτητα επεξεργαστών 24 MIPS.

κλάση το στατικό τμήμα του αλγορίθμου επέλεγε σταθερά για αυτές έναν κόμβο για να δρομολογήσει μία νέα δοσοληψία της κλάσης αυτής και μόνο το δυναμικό τμήμα του προσπαθούσε να δρομολογήσει κάπου αλλού την δοσοληψία προκειμένου να κάνει μία καλύτερη εξισορρόπηση του φόρτου ανάμεσα στους κόμβους. Αποτέλεσμα αυτού ήταν η αδυναμία του συστήματος να εξυπηρετήσει τον ρυθμό άφιξης νέων αιτήσεων στο σύστημα και να καταλήξει σε μεγάλο αριθμό από δοσοληψίες που έκαναν timeout ή ενεπλάκησαν σε αδιέξοδο, με αποτέλεσμα να τερματιστούν από το σύστημα. Έτσι ουσιαστικά η κακή ποιότητα της ομαδοποίησης είχε πολύ μεγάλη αρνητική επίπτωση στην απόδοση των αλγορίθμων δρομολόγησης.

Σε γενικές γραμμές ο καλύτερος αλγόριθμος δρομολόγησης ήταν ο WFW ακολουθούμενος από τον WFWC και τελευταίο τον JSQ (εκτός της περίπτωσης του K-Means).

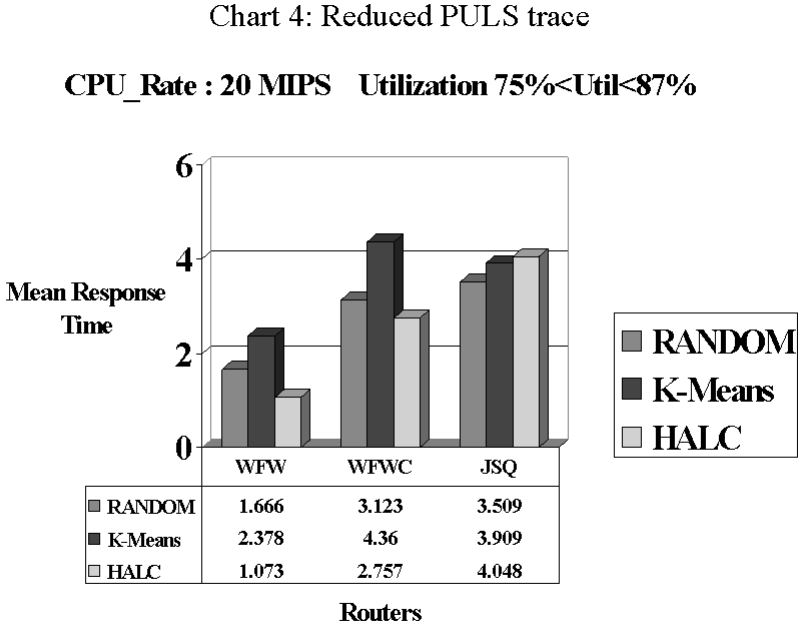
Ο αλγόριθμος ομαδοποίησης HALC με την ποιότητα της ομαδοποίησης που έκανε συνέτεινε στην επίτευξη των καλύτερων επιδόσεων τόσο με τον WFW όσο και με WFWC δρομολογητή δοσοληψιών. Παράλληλα πολύ καλή συμπεριφορά εμφανίζει και η χρήση του αποτελέσματος της ομαδοποίησης με τον αλγόριθμο Random αν και αυτό δεν αντανακλάται από το κριτήριο του τετραγωνικού σφάλματος. Ο λόγος αυτής της απρόσμενα καλής συμπεριφοράς είναι ο εξής: αρχικά θεωρήσαμε ως πιθανότητα εμφάνισης μίας τριπλέτας την πιθανότητα με την οποία εμφανίζεται στο αρχείο ιχνών. Οι πιθανότητες αυτές ήταν σε πολύ μεγάλο βαθμό διαφορετικές. Υπήρχαν δηλαδή τριπλέτες που εμφανίζονταν με πολύ μεγάλη πιθανότητα (περίπου 15 τριπλέτες) σε σχέση με την πιθανότητα εμφάνισης των υπολοίπων. Κατά την τυχαία ανάθεση των τριπλετών σε μία από τις 30 κλάσεις, οι



Σχήμα 6.5: Μέσος χρόνος απόκρισης για όλες τις κλάσεις στην περίπτωση του *Reduced DOA* αρχείο ιχνών και για ταχύτητα επεξεργαστών 28 MIPS.

τριπλέτες αυτές έχουν την πιο μεγάλη πιθανότητα εμφάνισης μέσα στην κάθε κλάση. Πριν από την πραγματοποίηση των πειραμάτων, όπως προείπαμε, γινόταν ένα monitor run για να σημειωθούν οι μέσες ανάγκες κάθε κλάσης για επεξεργασία, "επισκέψεις" σε κάθε κόμβο, καθυστέρηση για I/O και για επικοινωνία. Εάν για μία κλάση υπήρχε μία τριπλέτα με πολύ μεγαλύτερη πιθανότητα εμφάνισης σε σχέση με τις υπόλοιπες τότε κατά οι μετρήσεις του monitor run επηρεάζονταν σημαντικά από τις απαιτήσεις αυτής της τριπλέτας. Αποτέλεσμα όλων αυτών είναι οι WFW και WFWC να δρομολογούν μία δοσοληψία αυτής της κλάσης στον κόμβο που από το monitor run φάνηκε ότι η κλάση αυτή έχει καλύτερη επίδοση. Αυτός ο κόμβος όμως είναι ο κόμβος που είναι ο ιδανικός για την δρομολόγηση της τριπλέτας που έχει την μεγαλύτερη πιθανότητα εμφάνισης σε αυτήν την κλάση. Δηλαδή ακόμα και με την χρήση του Random αλγορίθμου ομαδοποίησης το σύστημα (με την βοήθεια του monitor run) έκανε την ιδανική δρομολόγηση για τις περισσότερες από τις τριπλέτες που έχουν μεγάλη πιθανότητα εμφάνισης στο αρχείο ιχνών. Έτσι λοιπόν η μετρήσιμη διαφορά μεταξύ του HALC και του Random ήταν αποτέλεσμα της καλύτερης ομαδοποίησης για τις υπόλοιπες τριπλέτες. Είναι χαρακτηριστικό ότι εάν είχαμε αριθμό κλάσεων ίσο με τον αριθμό τριπλετών τότε μετά από το monitor run και με αλγόριθμο ομαδοποίησης τον Random, οι δρομολογητές WFW και WFWC θα επέλεγαν πάντα τον ιδανικό κόμβο.

Έτσι εξηγείται γιατί ο HALC αν και κάνει σημαντικά καλύτερη ομαδοποίηση δεν μπορεί να φανεί αυτό με την χρήση του προσομοιωτή. Θα μπορούσαμε να ισχυριστούμε ότι στην περίπτωση που δεν κάναμε ομαδοποίηση των τριπλετών η απόδοση θα ήταν αυτή του JSQ καθώς μην έχοντας κάποιες πληροφορίες για τις διάφορες ομάδες το καλύτερο που θα μπορούσε να κάνει είναι να εξισορροπήσει τον φόρτο εργασίας ανάμεσα στους κόμβους του



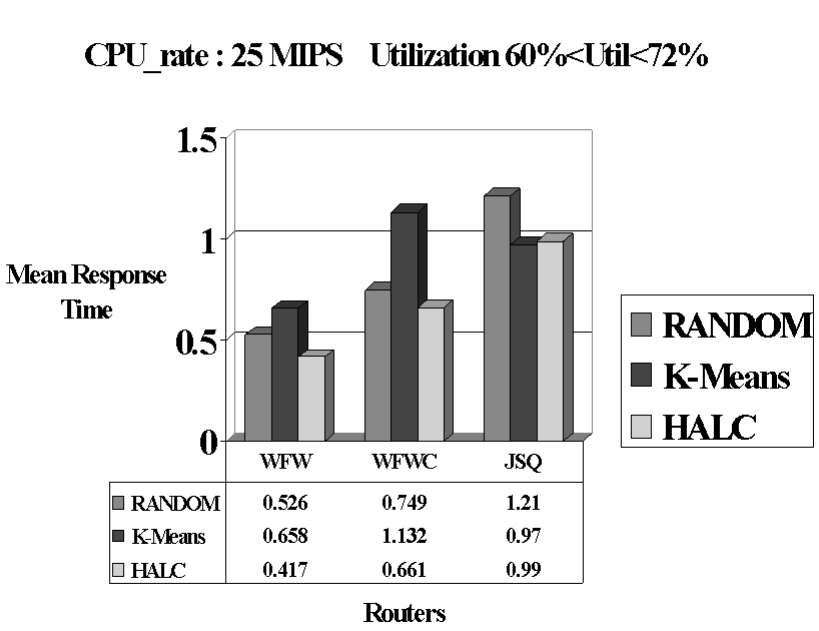
Σχήμα 6.6: Μέσος χρόνος απόκρισης για όλες τις κλάσεις στην περίπτωση του *Reduced PULS* αρχείο ιχνών και για ταχύτητα επεξεργαστών 20 MIPS.

συστήματος. Η διαφορά και μόνο στη απόδοση του WFW δρομολογητή για την περίπτωση της ομαδοποίησης με τον HALC αλγόριθμος σε σχέση με την απόδοση του συστήματος με την χρήση του JSQ δρομολογητή (με οποιοδήποτε αλγόριθμο ομαδοποίησης) φανερώνει το κέρδος που μπορεί να έχουμε από τον συνδυασμό ενός καλού δρομολογητή με έναν καλό αλγόριθμο ομαδοποίησης.

6.6 Πειραματικά αποτελέσματα για το *Reduced PULS* αρχείο ιχνών

Τα αποτελέσματα με την επίδοση του συστήματος για κάθε αλγόριθμο δρομολόγησης και για κάθε αλγόριθμο ομαδοποίησης που χρησιμοποιήθηκε και για το αρχείο ιχνών *Reduced PULS* , φαίνονται στα σχήματα 6.6, 6.7, και 6.8. Μέτρο σύγκρισης είναι και πάλι ο μέσος χρόνος απόκρισης για το σύνολο των κλάσεων του συστήματος. Στο σχήμα 6.6 η ταχύτητα του επεξεργαστή κάθε κόμβου είναι 20 MIPS (Million Instructions Per Second), για το σχήμα 6.7 25 MIPS, ενώ για το σχήμα 6.8 28 MIPS.

Οι ίδιες διαπιστώσεις ισχύουν και το δεύτερο αρχείο ιχνών. Εδώ δεν υπάρχει τόσο μεγάλη διαφορά στις πιθανότητες εμφάνισης των τριπλετών. Αποτέλεσμα αυτού είναι ότι η διαφορά της χρήσης του HALC ως αλγορίθμου ομαδοποίησης σε σχέση με τον Random να είναι πολύ μεγαλύτερη. Παράλληλα επειδή ο αριθμός από τριπλέτες είναι αρκετά μικρός (144) και οι ομάδες που ζητάμε να φτιαχθούν από τους αλγόριθμους ομαδοποίησης είναι 30, αυτό αναγκάζει ακόμα και τον K-Means να μοιράσει σε 30 ομάδες τις τριπλέτες με αποτέλεσμα



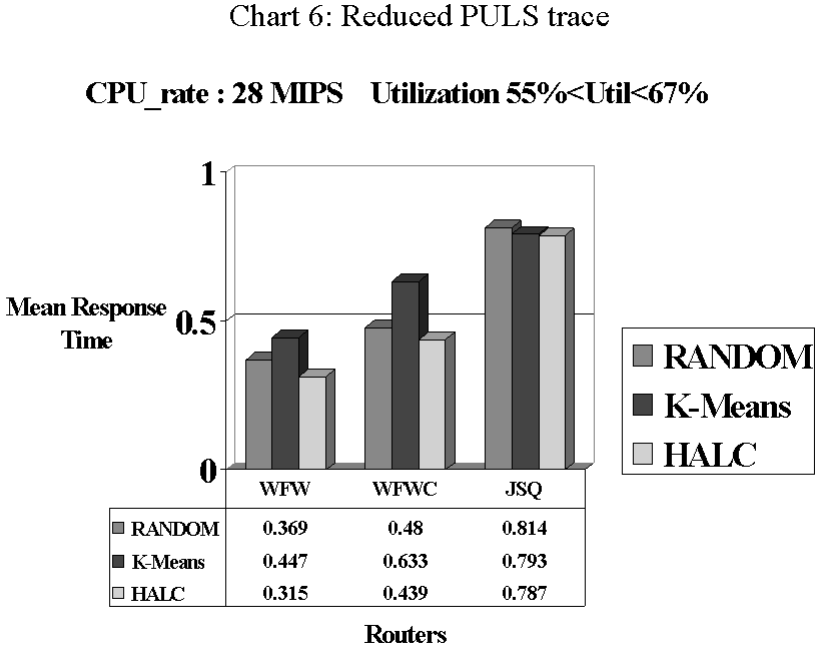
Σχήμα 6.7: Μέσος χρόνος απόκρισης για όλες τις κλάσεις στην περίπτωση του *Reduced PULS* αρχείο ιχνών και για ταχύτητα επεξεργαστών 25 MIPS.

η μέγιστη δυνατή ομάδα που να μπορεί να φτιάξει να αποτελείται από περίπου το 80% των τριπλετών. Πράγματι σχηματίζει και πάλι μία μεγάλη ομάδα που αποτελείται όμως από 84 τριπλέτες δηλ 58.3% του συνόλου των τριπλετών. Το ότι γίνεται καλύτερη ομαδοποίηση σε σχέση με την ομαδοποίηση που είχε κάνει στη περίπτωση του *Reduced DOA* φαίνεται και από το γεγονός ότι ο προσομοιωτής μετρά την απόδοση του K-Means αρκετά πιο κοντά στην απόδοση των άλλων αλγόριθμων δρομολόγησης και πάντως πολύ καλύτερα από την αντίστοιχη του πειράματος με το *Reduced DOA* αρχείο ιχνών.

Επίσης ο μέσος χρόνος απόκρισης είναι πολύ πιο μεγάλος σε σχέση με το *Reduced DOA* καθώς κατά μέσο όρο γίνονται παραπάνω κλήσεις προσπέλασεις προς την βάση δεδομένων σε σχέση με το *Reduced DOA* αρχείο ιχνών με άμεση συνέπεια αυξημένο ποσοστό δοσοληψιών που δεν έχουν επιτυχή τερματισμό.

6.7 Συμπεράσματα

Από τα αποτελέσματα της χρήσης των τριών αλγόριθμων ομαδοποίησης στη απόδοση των τριών αλγόριθμων δρομολόγησης φάνηκε καθαρά η επίπτωση που έχει η ποιότητα της ομαδοποίησης στην απόδοση των δρομολογητών. Πιο συγκεκριμένα η κακή ποιότητα ομαδοποίησης του K-Means επηρέασε αρνητικά σε πολύ μεγάλο βαθμό το σύστημα ειδικά με το *Reduced DOA* αρχείο ιχνών. Αντίθετα ο HALC αλγόριθμος ομαδοποίησης φάνηκε για άλλη μια φορά ότι εκτός του ότι είναι γρήγορος, κάνει και πολύ καλή ομαδοποίηση



Σχήμα 6.8: Μέσος χρόνος απόκρισης για όλες τις κλάσεις στην περίπτωση του *Reduced PULS* αρχείο ιχνών και για ταχύτητα επεξεργαστών 28 MIPS.

ακόμα και σε πραγματικά δεδομένα κάτι που επηρεάζει θετικά την απόδοση των αλγορίθμων δρομολόγησης. Ακόμα και ο Random αγλόριθμος ομαδοποίησης επιδρά θετικά στην απόδοση των αλγορίθμων δρομολόγησης λόγω του ότι όπως προείπαμε δρομολογεί στον σωστό κόμβο τις τριπλέτες που έχουν την μεγαλύτερη πιθανότητα εμφάνισης στο σύστημα. Έτσι παρόλο που οι ομάδες δημιουργούνται τυχαία, τελικά εμφανίζει απόδοση αρκετά καλύτερη από αυτήν που αναμέναμε.

Σίγουρα αυτή η μελέτη θα πρέπει να συνεχιστεί καθώς τα συμπεράσματα αυτά που προκύπτουν δείχνουν ότι αξίζει στα μεγάλα συστήματα επεξεργασίας δεδομένων να γίνει αρχικά μία ομαδοποίηση των δοσοληψιών ώστε η εφαρμογή αλγόριθμων δρομολόγησης να αποδώσει. Θα πρέπει όμως τα ίχνη να παρθούν από ένα πραγματικά κατανεμημένο σύστημα με πολλούς κόμβους ώστε να υπάρχουν κάποιες πληροφορίες που δεν ήταν διαθέσιμες σε αυτά τα αρχεία, όπως το σχήμα ανάθεσης των σελίδων της βάσης δεδομένων στους κόμβους.

Κεφάλαιο 7

Συμπεράσματα και μελλοντικές κατευθύνσεις

Στην ενότητα αυτή θα δούμε συνοπτικά τα αποτελέσματα της εργασίας αυτής καθώς και ορισμένες προτάσεις για επεκτάσεις και μελλοντική έρευνα.

7.1 Αποτελέσματα εργασίας

Στα πλαίσια αυτής της εργασίας έγινε η υλοποίηση των αρχιτεκτονικών κοινόχρηστων δίσκων (*Shared Disk*) και κοινόχρηστης ενδιάμεσης μνήμης (*Shared Intermediate Memory*). Η υλοποίησή τους έγινε στον προσομοιωτή συστημάτων επεξεργασίας δοσοληψιών TPsim που μέχρι τώρα υποστήριζε μόνο την *Shared Nothing* αρχιτεκτονική σύζευξης υπολογιστικών κόμβων. Προκειμένου να αποτελέσει ένα ολοκληρωμένο εργαλείο για την προσομοίωση τέτοιων συστημάτων, η επέκταση του TPsim έγινε με προσεκτικό τρόπο ώστε να μην επηρεαστεί η ήδη υπάρχουσα υλοποίηση. Αρχικά έγινε η μετατροπή της γλώσσας προδιαγραφής που χρησιμοποιεί ο προσομοιωτής για την περιγραφή του υπό μελέτη συστήματος. Παράλληλα έγιναν μία σειρά από μετατροπές - προσθήκες σε έναν μεγάλο αριθμό από συναρτήσεις ώστε να είναι σε θέση αναγνωρίζοντας την αρχιτεκτονική να εκτελούν τμήματα κώδικα που δεν είναι κοινά για τις διάφορες αρχιτεκτονικές.

Στη συνέχεια έγινε η υλοποίηση ενός πρωτοκόλλου για τον έλεγχο της συνδρομικότητας (concurrency control) το οποίο βασίζεται στην ύπαρξη ενός κοινού lock table για όλους τους κόμβους που μοιράζονται την βάση δεδομένων, κάτι που όμως απαιτεί την ανταλλαγή ενός πολύ μεγάλου αριθμού από μηνύματα. Η επιλογή αυτή στηρίχθηκε στην ιδέα ότι μπορεί να αποτελέσει βάση συγκρίσεων για πολλούς αλγόριθμους διότι μία μείωση του αριθμού των μηνυμάτων θα έχει άμεσο αντίκτυπο στην απόδοση του συστήματος. Τέλος αντιμετωπίστηκε και το ζήτημα του ελέγχου της συνέπειας των ενταμιευτών (coherency control) με την απαίτηση οι μεταβολές να προωθούνται σε μόνιμη μνήμη κατά τον τερματισμό μίας δοσοληψίας. Η επιλογή αυτή, αν και ευρύτατα διαδεδομένη σε συστήματα επεξεργασίας δοσοληψιών που βασίζονται στις λεγόμενες data sharing αρχιτεκτονικές, ήταν δεδομένη

καθώς λόγω της επιλογής του πρωτοκόλλου συνδρομικότητας δεν υπήρχε τρόπος να κρατάται η θέση (ενταμιευτής) της έγκυρης σελίδας μέσα στο σύστημα. Ο τρόπος πάντως της ενσωμάτωσης τους στον προσομοιωτή επιτρέπει την εύκολη αντικατάστασή τους από άλλα, πιο αποδοτικά πρωτόκολλα.

Χρησιμοποιώντας ως βάση την υλοποίηση αυτών των αρχιτεκτονικών έγινε η υλοποίηση ενός νέου αλγορίθμου δρομολόγησης κατάλληλου για τέτοιου είδους αρχιτεκτονικές. Τα αποτελέσματα που έδωσε είναι πολύ ικανοποιητικά ως προς την βελτίωση του χρόνου απόκρισης για τις μονάδες φόρτου που καλείται να εξυπηρετήσει το σύστημα και σίγουρα η απουσία από την διεθνή βιβλιογραφία αλγορίθμων δρομολόγησης κατάλληλων για αυτές τις αρχιτεκτονικές, τον καθιστά ως κάτι το σημαντικό σε αυτήν την εργασία.

Τέλος έγιναν μία σειρά από πειράματα για την επίδραση που έχει η ομαδοποίηση δοσοληψιών στη απόδοση αλγορίθμων δρομολόγησης. Η μελέτη αυτή έγινε με την βοήθεια ενός εργαλείου ομαδοποίησης δοσοληψιών, του CLUE και την διασύνδεσή του με το προσομοιωτή TPsim. Τα αποτελέσματα έδειξαν ότι η επίδραση μίας ομαδοποίησης δοσοληψιών στην απόδοση αλγορίθμων δρομολόγησης για συστήματα επεξεργασίας δοσοληψιών είναι πολύ μεγάλη καθώς η ποιότητα της ομαδοποίησης έχει άμεσο αντίκτυπο στις αποφάσεις των δρομολογητών και εν τέλει στην απόδοση του συστήματος.

7.2 Μελλοντικές επεκτάσεις και κατευθύνσεις

Οι προτάσεις για μελλοντικές υλοποιήσεις και μελέτες αφορούν δύο ξεχωριστά πεδία. Το πρώτο είναι επεκτάσεις στον ίδιο τον προσομοιωτή προκειμένου να του δοθεί η δυνατότητα να διευρύνει το πεδίο εφαρμογής του. Το δεύτερο αφορά συνέχιση της εργασίας που παρουσιάστηκε στο κεφάλαιο 6 την οποία θεωρούμε πολύ σημαντική αρχική προσπάθεια.

7.2.1 Επεκτάσεις TPsim

Η όλη φιλοσοφία του TPsim είναι να αποτελέσει μία βιβλιοθήκη από ρουτίνες που υλοποιούν αλγορίθμους : δρομολόγησης, χρονοπρογραμματισμού, αντικατάστασης σελίδων καθώς και πρωτοκόλλων ελέγχου της συνδρομικότητας, και της συνέπειας ενταμιευτών. Προς αυτήν την κατεύθυνση θα πρέπει να προσανατολιστούν οι μελλοντικές επεκτάσεις του TPsim. Μία βασική επέκταση θα μπορούσε να είναι η ανάπτυξη ενός πρωτοκόλλου ελέγχου της ταυτόχρονης προσπέλασης στα κοινόχρηστα δεδομένα για τις δύο αρχιτεκτονικές (*Shared Disk* και *Shared Intermediate Memory*) το οποίο να μειώνει τον αριθμό από τα μηνύματα που απαιτείται να ανταλλαγούν προκειμένου να κλειδωθούν δεδομένα της βάσης. Μία τέτοια βελτίωση θα μπορούσε να είναι η ανάθεση του ελέγχου τμήματος της βάσης σε κάθε κόμβο όπου ο κάθε κόμβος δεν απαιτείται να στείλει μήνυμα για να κλειδώσει ένα δεδομένο για το οποίο έχει τον έλεγχο. Σε διαφορετική περίπτωση θα μπορούσε να απαιτείται ο τρόπος κλειδώματος που υλοποιήθηκε στην παρούσα εργασία. Ένα τέτοιο πρωτόκολλο προτείνεται στην αναφορά [Rah86] το οποίο σε συνδυασμό με τον αλγόριθμο δρομολόγησης που αναπτύχθηκε μπορεί να βελτιώσει σε μεγάλο βαθμό την απόδοση του συστήματος καθώς

η δρομολόγηση εκτός του ότι θα κατέληγε σε αυξημένη πιθανότητα εύρεσης μίας σελίδας στον τοπικό ενταμιευτή, θα είχε και ως αποτέλεσμα την δραστική μείωση των μηνυμάτων που ανταλλάσσονται.

Επίσης είναι ενδιαφέρον να μελετηθεί η επίδοση συστημάτων με φόρτο εργασίας που περιλαμβάνει δοσοληψίες αλλά και επερωτήσεις. Οι επερωτήσεις αυτές στη βάση δεδομένων μπορεί να είναι το αποτέλεσμα ενός βελτιστοποιητή επερωτήσεων και να περιλαμβάνει πράξεις σάρωσης (scan) αλλά και πράξεις σύνδεσης (join) ώστε να γίνει δυνατή η μελέτη της αλληλεπίδρασης δοσοληψιών και επερωτήσεων όπως στην αναφορά [BCD+92]

Μία ακόμα ενδιαφέρουσα επέκταση είναι η μελέτη αλγορίθμων προσανατολισμένων στους στόχους επίδοσης για τον δυναμικό έλεγχο παραμέτρων του συστήματος ώστε να μπορεί να αντιμετωπιστεί η συμφόρηση στα συστήματα επεξεργασίας δοσοληψιών λόγω μίας στιγμιαίας αλλαγής του φόρτου εργασίας στο σύστημα [Rah97].

Τέλος άλλη μία επέκταση που μπορεί να γίνει είναι η ανάπτυξη ενός γραφικού τρόπου αναπαράστασης των τελικών αλλά και ενδιάμεσων αποτελεσμάτων του προσομοιωτή. Προς την ίδια κατεύθυνση είναι και η ανάπτυξη ενός ημιαυτοματοποιημένου τρόπου περιγραφής του υπό μελέτη μοντέλου, όπου αντί η γλώσσα να ορίζεται σε ένα αρχείο με βάση την γλώσσα προδιαγραφής, να υπάρχει ένα περιβάλλον που να επιτρέπει την εύκολη εισαγωγή παραμέτρων όπου για αρκετά πεδία θα υπάρχει ήδη μια προτεινόμενη τιμή από τον προσομοιωτή και το μόνο που θα χρειάζεται να κάνει ένας χρήστης είναι να ορίσει λίγες βασικές παραμέτρους, απλοποιώντας κατά πολύ την δημιουργία τέτοιων αρχείων.

7.2.2 Άλλες μελλοντικές κατευθύνσεις

Η επίδραση της ομαδοποίησης στην επίδοση των αλγόριθμων δρομολόγησης θα πρέπει να γίνει με πιο συστηματικό τρόπο αλλά κυρίως με την χρήση πραγματικών ιχνών από κατανεμημένο σύστημα όπου θα υπάρχουν οι πληροφορίες τόσο για τον τόπο αποθήκευσης των δεδομένων, όσο και για τον χρόνο άφιξης κάθε δοσοληψίας στο σύστημα ώστε να μην χρειάζονται τόσες παραδοχές προκειμένου να γίνει δυνατή η χρήση αυτών των ιχνών. Επίσης χρήσιμο θα ήταν να γίνει η αξιολόγηση και άλλων αλγόριθμων ομαδοποίησης που βασίζονται στην "εν πτήσει" (on the fly) ομαδοποίηση [Κλο97]. Σε αυτόν τον τομέα έχει αρχίσει ήδη να γίνεται κάποια δουλειά.

Παράρτημα Α

Token γλώσσας προδιαγραφής

Ακολουθεί η περιγραφή των token της γλώσσας προδιαγραφής που υποστηρίζει ο προσομοιωτής.

```
"**SYSTEM CONFIGURATION**"  { return(SYS_CONFIG_TOKEN); }
"DEFINE"          { return(DEFINE); }
"OF"            { return(OF); }
"WITH"           { return(WITH); }
"AS"            { return(AS); }
"AT"            { return(AT); }
"AND"            { return(AND_T); }
"IN"            { return(IN); }
"OR"            { return(OR); }
"NOT"            { return(NOT); }
"<>"            { return(NEQ); }
"<="            { return(LEQ); }
">="            { return(GEQ); }
"="            { return(EQ); }
"<"            { return(LT); }
">"            { return(GT); }
"CLASS"           { return(CLASS); }
"NODE_CLASS"         { return(NODE_CLASS_TOKEN); }
"SIMPLE_NODE_CLASS"      { return(SIMPLE_NODE_CLASS_TOKEN); }
"SHARED_DATABASE"       { return(SHARED_DATABASE); }
"COMPLEX_NODE_CLUSTER"    { return(COMPLEX_NODE_CLUSTER); }
"CPUcnt"          { return(CPU_CNT); }
"numDisks"          { return(NUM_DISKS); }
"DM_BUFFER_SIZE"       { return(DM_BUFFER_SIZE_TOKEN); }
"SB_BUFFER_SIZE"       { return(SB_BUFFER_SIZE_TOKEN); }
"SB_IO_COST"         { return(SB_IO_COST_TOKEN); }
"SB_DB_COPY_DELAY"      { return(SB_DB_COPY_DELAY_TOKEN); }
"MPL"           { return(MPL_TOKEN); }
"CPUrate"          { return(CPU_RATE_T); }
"ATTACH_TASK_COST"      { return(ATTACH_TASK_COST_TOKEN); }
"DM_INTERFACE_COST"      { return(DM_INTERFACE_COST_TOKEN); }
```

```
"DM_CALL_COST"       { return(DM_CALL_COST_TOKEN); }
"DM_IO_COST"        { return(DM_IO_COST_TOKEN); }
"FUNCTION_SHIP_SEND_COST"  { return(FUNCTION_SHIP_SEND_COST_TOKEN); }
"FUNCTION_SHIP_RECV_COST"  { return(FUNCTION_SHIP_RECV_COST_TOKEN); }
"PRIMARY_PREPARE_COST"    { return(PRIMARY_PREPARE_COST_TOKEN); }
"SEND_PREPARE_COST"     { return(SEND_PREPARE_COST_TOKEN); }
"SEND_COMMIT_COST"      { return(SEND_COMMIT_COST_TOKEN); }
"PRIMARY_COMMIT_COST"    { return(PRIMARY_COMMIT_COST_TOKEN); }
"RECV_PREPARE_COST"     { return(RECV_PREPARE_COST_TOKEN); }
"RECV_COMMIT_COST"      { return(RECV_COMMIT_COST_TOKEN); }
"LOG_IO_COST"        { return(LOG_IO_COST_TOKEN); }
"SECONDARY_PREPARE_COST"   { return(SECONDARY_PREPARE_COST_TOKEN); }
"SECONDARY_COMMIT_COST"   { return(SECONDARY_COMMIT_COST_TOKEN); }
"DETACH_TASK_COST"      { return(DETACH_TASK_COST_TOKEN); }
"SEND_MESSAGE_COST"     { return(SEND_MESSAGE_COST_TOKEN); }
"RECV_MESSAGE_COST"     { return(RECV_MESSAGE_COST_TOKEN); }
"BUFFER_INV_COST"      { return(BUFFER_INV_COST_TOKEN); }
"LOG_IO_DEVICE"       { return(LOG_IO_DEVICE_TOKEN); }
"DM_IO_DEVICE"       { return(DM_IO_DEVICE_TOKEN); }
"NODE_CLUSTER"       { return(NODE_CLUSTER); }
"SIMPLE_NODE_CLUSTER"    { return(SIMPLE_NODE_CLUSTER); }
"SHARED_BUFFER"       { return(SHARED_BUFFER_TOKEN); }
"FRONT_END"         { return(FRONT_END_TOKEN); }
"QUEUE_MANAGER"       { return(QUEUE_MANAGER_TOKEN); }
"GATEWAY"          { return(GATEWAY_TOKEN); }
"MEMBER_NODES"        { return(MEMBER_NODES); }
"numNodes"         { return(NUM_NODES); }
"IPC_DELAY"         { return(IPC_DELAY); }
"NODE"           { return(NODE); }
"SIMPLE_NODE"        { return(SIMPLE_NODE); }
"LINK_CLASS"        { return(LINK_CLASS_TOKEN); }
"LINK"           { return(LINK_T); }
"IO_DEVICE"         { return(IO_DEVICE); }
"IO_DELAY_MIN"       { return(IO_DELAY_MIN_TOKEN); }
"IO_DELAY_MAX"       { return(IO_DELAY_MAX_TOKEN); }
"IO_COPY_DELAY"       { return(IO_COPY_DELAY_TOKEN); }
"IO_ROTATIONAL_DELAY"    { return(IO_ROT_DELAY_TOKEN); }
"IO_SEEK_DELAY"       { return(IO_SEEK_DELAY_TOKEN); }
"IO_LOAD_DELAY"       { return(IO_LOAD_DELAY_TOKEN); }
"NUM_HEADS"         { return(NUM_HEADS); }
"NUM_SECTORS"        { return(NUM_SECTORS); }
"packetSize"        { return(PACKET_SIZE); }
"transferRate"       { return(TRANSFER_RATE); }
"from"           { return(SRC); }
"to"            { return(DST); }
"busArbitration"      { return(BUS_DISCIPLINE); }
"**DATA DISTRIBUTION**"   { return(DATA_DISTRIBUTION_TOKEN); }
"RELATION"         { return(RELATION); }
"ATTRIBUTES"        { return(ATTRIBUTES); }
"NUMERIC"          { return(NUMERIC); }
"SYMBOLIC"         { return(SYMBOLIC); }
```

```
"KEY"           { return(KEY); }
"INDEX"          { return(INDEX); }
"CLUSTERED_INDEX"      { return(CLUSTERED_INDEX); }
"ON"           { return(ON_T); }
"ALIAS"          { return(ALIAS); }
"FRAGMENTATION"       { return(FRAGMENTATION); }
"HORIZONTAL"        { return(HORIZONTAL); }
"VERTICAL"         { return(VERTICAL); }
"COLUMNS"         { return(COLUMNS); }
"numItems"         { return(NUM_ITEMS); }
"fromTuple"         { return(FROM_TUPLE); }
"toTuple"         { return(TO_TUPLE); }
"INSTANCE"         { return(INSTANCE); }
"**WORKLOAD DESCRIPTION**"  { return(WORKLOAD_DESCRIPTION_TOKEN); }
"TRANSACTION_CLASS"     { return(TRANSACTION_CLASS); }
"WORKFLOW_CLASS"      { return(WORKFLOW_CLASS); }
"program"         { return(PROGRAM_TOKEN); }
"chain"          { return(CHAIN_TOKEN); }
"PERFORMANCE_GOAL"      { return(PERFORMANCE_GOAL_TOKEN); }
"SELECT"          { return(SELECT_TOKEN); }
"SELECTIVITY"        { return(SELECTIVITY_TOKEN); }
"ACCESS_METHOD"       { return(ACCESS_METHOD_TOKEN); }
"SCAN_SEQUENTIAL"      { return(SCAN_SEQUENTIAL_TOKEN); }
"SCAN_CLUSTERED_INDEX"    { return(SCAN_CLUSTERED_INDEX_TOKEN); }
"SCAN_UNCLUSTERED_INDEX"   { return(SCAN_UNCLUSTERED_INDEX_TOKEN); }
"MERGE_COST"        { return(MERGE_COST_TOKEN); }
"FROM"          { return(FROM_TOKEN); }
"WHERE"          { return(WHERE_TOKEN); }
"arrivalRate"        { return(ARRIVAL_RATE); }
"numClients"        { return(NUM_CLIENTS); }
"tuplesPerPage"       { return(TUPLES_PER_PAGE); }
"applicationBurst"      { return(APPLICATION_BURST); }
"blocksAccessed_min"     { return(MIN_BLOCKS_ACCESSED); }
"blocksAccessed_max"     { return(MAX_BLOCKS_ACCESSED); }
"writeProbability"      { return(WRITE_PROBABILITY); }
"accessProbability"     { return(ACCESS_PROBABILITY); }
"PROBABILITY_IO"      { return(PROBABILITY_IO_TOKEN); }
"PROBABILITY"       { return(PROBABILITY_TOKEN); }
"WORKLOAD"         { return(WORKLOAD_TOKEN); }
"ALL_SITES"         { return(ALL_SITES_TOKEN); }
"ALL_DISKS"         { return(ALL_DISKS_TOKEN); }
"CLIENT_CLASS"       { return(CLIENT_CLASS_TOKEN); }
"SUBMIT"          { return(SUBMIT); }
"SYNTHETIC"         { return(SYNTHETIC); }
"BUFFER_HIT_RATIO"      { return(BUFFER_HIT_RATIO); }
"TRACE_BASED"        { return(TRACE_BASED); }
"read"          { return(READ); }
"write"          { return(WRITE); }
"read_and_update"      { return(READ_AND_UPDATE); }
"compute"         { return(COMPUTE); }
"Begin_Transaction"     { return(BEGIN_TRANSACTION); }
```

```
"Commit"          { return(COMMIT); }
"Abort"           { return(ABORT); }
```

Παράρτημα Β

Γραμματική γλώσσας προδιαγραφής

Εδώ μπορεί κανείς να δει την περιγραφή της context-free γραμματικής που χρησιμοποιεί ο προσομειωτής προκειμένου να δεχθεί την περιγραφή ενός μοντέλου . Τα token εμφανίζονται με κεφαλαία γράμματα.

```
SysConfigFile: SYS_CONFIG_TOKEN
       SystemConfiguration
       DATA_DISTRIBUTION_TOKEN
       DataDistribution
       WORKLOAD_DESCRIPTION_TOKEN
       WorkloadDescription PerformanceGoalSpecificationList
      ;
SystemConfiguration: ComponentDefinition
        | SystemConfiguration ComponentDefinition
        ;

DataDistribution: DataDistributionDefinition
       | DataDistribution DataDistributionDefinition
       ;

WorkloadDescription: WorkloadClassList ClientClassList
         WorkloadLocalization
        ;

ComponentDefinition: IoDeviceDefinition
        | NodeSNClassDefinition
        | NodeSDClassDefinition
        | NodeSNClusterDefinition
        | NodeSDClusterDefinition
        | SharedBufferDefinition
        | NodeClusterDefinition
        | NodeDefinition
        | SimpleNodeDefinition
        | LinkClassDefinition
        | LinkDefinition
```

```
        ;

IoDeviceDefinition: DEFINE IO_DEVICE ID WITH '{'
          IO_COPY_DELAY_TOKEN ':' REALNUM ';'
          IO_LOAD_DELAY_TOKEN ':' REALNUM ';'
          IO_SEEK_DELAY_TOKEN ':' REALNUM ';'
          IO_ROT_DELAY_TOKEN ':' REALNUM ';'
          NUM_HEADS ':' INTNUM ';'
          NUM_SECTORS ':' INTNUM ';' '}'
        | DEFINE IO_DEVICE ID WITH '{'
          IO_DELAY_MIN_TOKEN ':' REALNUM ';'
          IO_DELAY_MAX_TOKEN ':' REALNUM ';' '}'
        ;

NodeDefinition: DEFINE NODE ID OF CLASS ID ';'
      | DEFINE NODE ID OF CLASS ID
        '(' FRONT_END_TOKEN ')' ';'
      ;

SimpleNodeDefinition: DEFINE SIMPLE_NODE ID OF CLASS ID ';'
      | DEFINE SIMPLE_NODE ID OF CLASS ID
        '(' FRONT_END_TOKEN ')' ';'
      ;

LinkDefinition: DEFINE LINK_T ID OF CLASS ID WITH '{'
        SRC ':' ID ';'
        DST ':' ID ';' '}'
      ;

LinkClassDefinition: DEFINE LINK_CLASS_TOKEN ID WITH '{'
          PACKET_SIZE ':' INTNUM ';'
          TRANSFER_RATE ':' INTNUM ';' '}'
        ;

NodeSNClusterDefinition: DEFINE NODE_CLUSTER ID OF CLASS ID WITH '{'
           NUM_NODES ':' INTNUM ';'
           PACKET_SIZE ':' INTNUM ';'
           TRANSFER_RATE ':' INTNUM ';'
           BUS_DISCIPLINE ':' ID ';' '}'
         | DEFINE NODE_CLUSTER ID WITH '{'
           NUM_NODES ':' INTNUM ';'
           PACKET_SIZE ':' INTNUM ';'
           TRANSFER_RATE ':' INTNUM ';'
           BUS_DISCIPLINE ':' ID ';'
           MEMBER_NODES ':' '{' ID_List '}' '}'
         ;

NodeSDClusterDefinition: DEFINE SIMPLE_NODE_CLUSTER ID OF CLASS ID
           WITH '{'
           NUM_NODES ':' INTNUM ';'
           PACKET_SIZE ':' INTNUM ';'
```

```
            TRANSFER_RATE ':' INTNUM ';'
            BUS_DISCIPLINE ':' ID ';'
            DEFINE SHARED_DATABASE ID WITH '{'
            IO_DEVICE ':' ID ';'
            NUM_DISKS ':' INTNUM ';' '}' '}'
          | DEFINE SIMPLE_NODE_CLUSTER ID WITH '{'
            NUM_NODES ':' INTNUM ';'
            PACKET_SIZE ':' INTNUM ';'
            TRANSFER_RATE ':' INTNUM ';'
            BUS_DISCIPLINE ':' ID ';'
            MEMBER_NODES ':' '{' ID_List '}'
            DEFINE SHARED_DATABASE ID WITH '{'
            IO_DEVICE ':' ID ';'
            NUM_DISKS ':' INTNUM ';' '}' '}'
          ;

NodeClusterDefinition: DEFINE COMPLEX_NODE_CLUSTER ID WITH '{'
            NUM_NODES ':' INTNUM ';'
            PACKET_SIZE ':' INTNUM ';'
            TRANSFER_RATE ':' INTNUM ';'
            BUS_DISCIPLINE ':' ID ';'
            MEMBER_NODES ':' '{' ID_List '}' '}'
          | DEFINE COMPLEX_NODE_CLUSTER ID WITH '{'
            NUM_NODES ':' INTNUM ';'
            PACKET_SIZE ':' INTNUM ';'
            TRANSFER_RATE ':' INTNUM ';'
            BUS_DISCIPLINE ':' ID ';'
            MEMBER_NODES ':' '{' ID_List '}'
            DEFINE SHARED_DATABASE ID WITH '{'
            IO_DEVICE ':' ID ';'
            NUM_DISKS ':' INTNUM ';' '}' '}'
          ;

SharedBufferDefinition: DEFINE SHARED_BUFFER_TOKEN ID IN
            SIMPLE_NODE_CLUSTER ID WITH '{'
            IO_COPY_DELAY_TOKEN ':' REALNUM ';'
            SB_BUFFER_SIZE_TOKEN ':' INTNUM ';'
            SB_IO_COST_TOKEN ':' REALNUM ';'
            SB_DB_COPY_DELAY_TOKEN ':' REALNUM ';' '}'
          ;

NodeSNClassDefinition: DEFINE NODE_CLASS_TOKEN ID WITH '{'
            CPU_CNT ':' INTNUM ';'
            MPL_TOKEN ':' INTNUM ';'
            CPU_RATE_T ':' REALNUM ';'
            ATTACH_TASK_COST_TOKEN ':' REALNUM ';'
            DM_INTERFACE_COST_TOKEN ':' REALNUM ';'
            DM_CALL_COST_TOKEN ':' REALNUM ';'
            DM_IO_COST_TOKEN ':' REALNUM ';'
            FUNCTION_SHIP_SEND_COST_TOKEN ':' REALNUM ';'
            FUNCTION_SHIP_RECV_COST_TOKEN ':' REALNUM ';'
```

```
          PRIMARY_PREPARE_COST_TOKEN ':' REALNUM ';'
          SEND_PREPARE_COST_TOKEN ':' REALNUM ';'
          SEND_COMMIT_COST_TOKEN ':' REALNUM ';'
          PRIMARY_COMMIT_COST_TOKEN ':' REALNUM ';'
          RECV_PREPARE_COST_TOKEN ':' REALNUM ';'
          RECV_COMMIT_COST_TOKEN ':' REALNUM ';'
          LOG_IO_COST_TOKEN ':' REALNUM ';'
          SECONDARY_PREPARE_COST_TOKEN ':' REALNUM ';'
          SECONDARY_COMMIT_COST_TOKEN ':' REALNUM ';'
          DETACH_TASK_COST_TOKEN ':' REALNUM ';'
          DM_BUFFER_SIZE_TOKEN ':' INTNUM ';'
          LOG_IO_DEVICE_TOKEN ':' ID ';'
          DM_IO_DEVICE_TOKEN ':' ID ';'
          NUM_DISKS ':' INTNUM ';' '}'
         ;

NodeSDClassDefinition: DEFINE SIMPLE_NODE_CLASS_TOKEN ID WITH '{'
          CPU_CNT ':' INTNUM ';'
          MPL_TOKEN ':' INTNUM ';'
          CPU_RATE_T ':' REALNUM ';'
          ATTACH_TASK_COST_TOKEN ':' REALNUM ';'
          DM_INTERFACE_COST_TOKEN ':' REALNUM ';'
          DM_CALL_COST_TOKEN ':' REALNUM ';'
          DM_IO_COST_TOKEN ':' REALNUM ';'
          FUNCTION_SHIP_SEND_COST_TOKEN ':' REALNUM ';'
          FUNCTION_SHIP_RECV_COST_TOKEN ':' REALNUM ';'
          PRIMARY_PREPARE_COST_TOKEN ':' REALNUM ';'
          SEND_PREPARE_COST_TOKEN ':' REALNUM ';'
          SEND_COMMIT_COST_TOKEN ':' REALNUM ';'
          PRIMARY_COMMIT_COST_TOKEN ':' REALNUM ';'
          RECV_PREPARE_COST_TOKEN ':' REALNUM ';'
          RECV_COMMIT_COST_TOKEN ':' REALNUM ';'
          LOG_IO_COST_TOKEN ':' REALNUM ';'
          SECONDARY_PREPARE_COST_TOKEN ':' REALNUM ';'
          SECONDARY_COMMIT_COST_TOKEN ':' REALNUM ';'
          DETACH_TASK_COST_TOKEN ':' REALNUM ';'
          SEND_MESSAGE_COST_TOKEN':' REALNUM ';'
          RECV_MESSAGE_COST_TOKEN':' REALNUM ';'
          DM_BUFFER_SIZE_TOKEN ':' INTNUM ';'
          LOG_IO_DEVICE_TOKEN ':' ID ';'
          BUFFER_INV_COST_TOKEN':' REALNUM ';' '}'
         ;

DataDistributionDefinition: RelationDefinition
           | AliasDefinition
           | InstanceDefinitionSN
           | InstanceDefinitionSD
           | FragmentDefinition
           ;

ID_List: ID
```

```
   | ID_List ',' ID
   ;

RelationDefinition: DEFINE RELATION ID WITH '{'
          RelationSchema
          TUPLES_PER_PAGE ':' REALNUM ';'
          KEY EQ '{' ID_List '}' ';'
          INDEX ON_T '{' ID_List '}' ';' '}'
        | DEFINE RELATION ID WITH '{'
          RelationSchema
          TUPLES_PER_PAGE ':' REALNUM ';'
          KEY EQ '{' ID_List '}' ';'
          CLUSTERED_INDEX ON_T '{' ID_List '}' ';'
          INDEX ON_T '{' ID_List '}' ';' '}'
        ;

AttrDefList:  AttrDef
     |  AttrDefList AttrDef
     ;

AttrDef:  ID ':' NUMERIC ';'
   |  ID ':' SYMBOLIC ';'
   ;

RelationSchema: ATTRIBUTES '{' AttrDefList '}'
      ;

AliasDefinition:  DEFINE ALIAS ID OF RELATION ID ';'
      ;

InstanceDefinitionSD: DEFINE INSTANCE
          OF RELATION ID
          AT ALL_DISKS_TOKEN
          OF SIMPLE_NODE_CLUSTER ID
          WITH NUM_ITEMS ':' INTNUM ';'
         | DEFINE INSTANCE OF RELATION ID AT
          '{' ID_List '}' OF
          SIMPLE_NODE_CLUSTER ID
          WITH NUM_ITEMS ':' INTNUM ';'
         ;

InstanceDefinitionSN: DEFINE INSTANCE
          OF RELATION ID AT ALL_SITES_TOKEN
          WITH NUM_ITEMS ':' INTNUM ';'
        |  DEFINE INSTANCE
          OF RELATION ID AT
          '{' ID_List '}'
          WITH NUM_ITEMS ':' INTNUM ';'
        ;

FragmentDefinition:  HorizontalFragmentDefinition
```

```
        | VerticalFragmentDefinition
        ;

Predicate: '(' SimplePredicate ')'
    | '(' Predicate ')'
    | NOT '(' Predicate ')'
    | '(' Predicate AND_T Predicate ')'
    | '(' Predicate OR Predicate ')'
    ;

SimplePredicate: ID EQ ID
       | ID NEQ ID
       | ID EQ INTNUM
       | ID NEQ INTNUM
       | ID LEQ INTNUM
       | ID GEQ INTNUM
       | ID LT INTNUM
       | ID GT INTNUM
       | ID EQ REALNUM
       | ID NEQ REALNUM
       | ID LEQ REALNUM
       | ID GEQ REALNUM
       | ID LT REALNUM
       | ID GT REALNUM
       ;

H_Fragm_List: ID WITH Predicate ',' TUPLES_PER_PAGE ':' REALNUM ';'
     | H_Fragm_List
       ID WITH Predicate ',' TUPLES_PER_PAGE ':' REALNUM ';'
     ;

HorizontalFragmentDefinition: DEFINE HORIZONTAL FRAGMENTATION OF ID
             AS '{' H_Fragm_List '}'
           ;

V_Fragm_List: ID WITH COLUMNS '{' ID_List '}' ','
       TUPLES_PER_PAGE ':' REALNUM ';'
     | V_Fragm_List
       ID WITH COLUMNS '{' ID_List '}' ','
       TUPLES_PER_PAGE ':' REALNUM ';'
     ;

VerticalFragmentDefinition: DEFINE VERTICAL FRAGMENTATION OF ID
            AS '{' V_Fragm_List '}'
          ;

WorkloadClassList: WorkloadClassDefinition
       | WorkloadClassList WorkloadClassDefinition
       ;

ClientClassList: ClientClassDefinition
```

```
      | ClientClassList ClientClassDefinition
      ;

TxClassDistributionList: TxClassDistribution
          | TxClassDistributionList
            TxClassDistribution
         ;

TxClassDistribution: SUBMIT ID WITH
         PROBABILITY_TOKEN REALNUM ';'
        ;

ArrivalRateVariation: ARRIVAL_RATE ':' REALNUM
          IN '[' REALNUM ',' REALNUM ']' ';'
         ;

ArrivalRateVariations: ArrivalRateVariations ArrivalRateVariation
         | ArrivalRateVariation
         ;

ClientClassDefinition: DEFINE CLIENT_CLASS_TOKEN ID AS '{'
           ARRIVAL_RATE ':' REALNUM ';'
           TxClassDistributionList
           '}'
         | DEFINE CLIENT_CLASS_TOKEN ID AS '{'
           ARRIVAL_RATE ':' REALNUM ';'
           ArrivalRateVariations
           TxClassDistributionList
           '}'          ;

ReferenceLocalityDefinition: InstanceReferenceLocality
           | ReferenceLocalityDefinition
             InstanceReferenceLocality
           ;

InstanceReferenceLocality: ACCESS_PROBABILITY OF INSTANCE
            ID ':' REALNUM ';'
            WRITE_PROBABILITY OF INSTANCE
            ID ':' REALNUM ';'
           ;

WorkloadClassDefinition: DEFINE TRANSACTION_CLASS ID AS '{'
           APPLICATION_BURST ':' REALNUM ';'
           MIN_BLOCKS_ACCESSED ':' INTNUM ';'
           MAX_BLOCKS_ACCESSED ':' INTNUM ';'
           ReferenceLocalityDefinition '}'
         | DEFINE TRANSACTION_CLASS ID AS '{'
           APPLICATION_BURST ':' REALNUM ';'
           MIN_BLOCKS_ACCESSED ':' INTNUM ';'
           MAX_BLOCKS_ACCESSED ':' INTNUM ';'
           PROBABILITY_IO_TOKEN ':' REALNUM ';'
```

```
            IO_DELAY_MIN_TOKEN ':' REALNUM ';'
            IO_DELAY_MAX_TOKEN ':' REALNUM ';'
            ReferenceLocalityDefinition '}'
          | DEFINE TRANSACTION_CLASS ID AS '{'
            APPLICATION_BURST ':' REALNUM ';'
            DMLstmtList '}'
          | DEFINE WORKFLOW_CLASS ID AS '{'
            PROGRAM_TOKEN ':' ID ';'
            CHAIN_TOKEN ':' '{' ID_List '}' ';' '}'
         ;

VarList: Var
   | VarList ',' Var
   ;

Var: ID FROM_TOKEN ID
 ;

DMLstmtList: DMLstmt
     | DMLstmtList DMLstmt
     ;

AccessMethod: ACCESS_METHOD_TOKEN OF ID ':'
       SCAN_CLUSTERED_INDEX_TOKEN ';'
     | ACCESS_METHOD_TOKEN OF ID ':'
       SCAN_UNCLUSTERED_INDEX_TOKEN ';'
     | ACCESS_METHOD_TOKEN OF ID ':'
       SCAN_SEQUENTIAL_TOKEN ';'
     ;

AccessMethodList: AccessMethod
       | AccessMethodList AccessMethod
       ;

Selectivity: SELECTIVITY_TOKEN OF ID ':' REALNUM ';'
     ;

SelectivityList: Selectivity
      | SelectivityList Selectivity
      ;

DMLstmt: SELECT_TOKEN VarList WHERE_TOKEN Predicate WITH
     '{' MERGE_COST_TOKEN ':' REALNUM ';'
       AccessMethodList SelectivityList '}' ';'
   ;

ClientClassDistributionList: ClientClassDistribution
           | ClientClassDistributionList
             ClientClassDistribution
           ;
```

```
ClientClassDistribution: NUM_CLIENTS OF CLASS ID ':' INTNUM
           AT ALL_SITES_TOKEN ';'
         | NUM_CLIENTS OF CLASS ID ':' INTNUM
           AT '{' ID_List '}' ';'
        ;

WorkloadLocalization: DEFINE SYNTHETIC WORKLOAD_TOKEN ID AS '{'
          ClientClassDistributionList '}'
        ;

PerformanceGoalSpecificationList: /* empty */
             | PerformanceGoalSpecificationList
               PerformanceGoalSpecification
             ;

PerformanceGoalSpecification: DEFINE PERFORMANCE_GOAL_TOKEN OF CLASS
               ID ':' REALNUM ';'
            ;
```

Παράρτημα C

Αναλυτική περιγραφή αλγορίθμου δρομολόγησης

Διαδικασία 9 *Υπολογισμός του πίνακα Nopt(i,j)*

```
{high1} ← 1;
{Loops} = 0;
{mult1} = {mult2} ← 0.8;
{already_caclulated} ← {Prob_changed} ← FALSE;
WHILE (high1 <= NumClasses && Loops < SPECIFIED_LOOPS)
    if (already_calculated == FALSE)
        **************************************************************************
        ** Όλα τα παρακάτω υπολογίζονται με βάση τον πίνακα πιθανοτήτων p(i,j)
        ** όπου το στοιχείο p(i,j) φανερώνει την πιθανότητα δρομολόγησης μίας
        ** δοσοληψίας της κλάσης i στον κόμβο επεξεργασίας j.
        **************************************************************************
        /*
        * Η συνάρτηση computeAvgNumOfLocalMisses() υπολογίζει τον πίνακα Mloc(i,j)
        * που περιλαμβάνει τον αριθμό των σελίδων που δεν βρίσκονται στον τοπικό
        * ενταμιευτή, για μία δοσοληψία κλάσης i που εκτελείται στον κόμβο j.
        */
        Mloc = computeAvgNumOfLocalMisses();
        /*
        * Η συνάρτηση computeAvgNumOfDASDMisses() υπολογίζει τον πίνακα
        * MDasd(i,j)  που περιλαμβάνει τον αριθμό των σελίδων που δεν βρίσκονται
        * στον κοινόχρηστο ενταμιευτή, για μία δοσοληψία κλάσης i που εκτελείται
        * στον κόμβο j. Αν δεν υπάρχει κοινόχρηστος ενταμιευτής τότε
        * MDasd(i,j) = Mloc(i,j) για κάθε i,j.
        */
        MDasd = computeAvgNumOfDASDMisses();
        /*
```

```
        * Η συνάρτηση computeProcessingRequirements() υπολογίζει τον πίνακα S(i,j)
        * που περιλαμβάνει το μέσο χρονικό κόστος επεξεργασίας για μία
        * δοσοληψία κλάσης i
        * που εκτελείται στον κόμβο j.
        */
        S = computeProcessingRequirements();
        /*
        * Η συνάρτηση computeProcessorUtilization() υπολογίζει τον πίνακα Util(j)
        * που περιλαμβάνει τον βαθμό χρησιμοποίησης κάθε κόμβου j.
        */
        Util = computeProcessorUtilization();
        /*
        * Η συνάρτηση computeResponseTime() υπολογίζει τον πίνακα R(i,j)
        * όπου υπολογίζεται ο εκτιμώμενος χρόνος απόκρισης για μία
        * δοσοληψία κλάσης i
        * που εκτελείται στον κόμβο j.
        */
        R = computeResponseTime();
        /*
        * Η συνάρτηση computeAvgResponseTime() υπολογίζει τον πίνακα Avg(i)
        * όπου υπολογίζεται ο εκτιμώμενος μέσος χρόνος απόκρισης
        * για κάθε κλάση δοσοληψιών i.
        */
        AvgR = computeAvgResponseTime();
        /*
        * Η συνάρτηση computePerformanceIndex() υπολογίζει τον πίνακα PerfIndex(i)
        * όπου υπολογίζεται ο εκτιμώμενος δείκτης επίδοσης, για κάθε κλάση
        * δοσοληψιών i.
        */
        PerfIndex = computePerformanceIndex();
    end
    {already_caclulated} ← FALSE;
    {exit} ← FALSE;
    /*
    * Η συνάρτηση Findhigh1_th_component() βρίσκει το high1 μεγαλύτερο
    * στοιχείο (κλάση) του πίνακα PerfIndex
    */
    high2 = Findhigh1_th_component();
    /*
    * Οι δύο επόμενες συναρτήσεις υπολογίζουν δύο πίνακες, βάση των οποίων γίνεται
    * υπολογισμός του νέου πίνακα δρομολόγησης
    * (συνάρτηση computeNewProbabilityMatrix())
    */
    F = computeF();
    Low = computeLow();
```

```
WHILE (exit == FALSE)
    q = computeNewProbabilityMatrix();
    **********************************************************************
    ** Υπολογίζεται με βάση τον πίνακα πιθανοτήτων q(i,j) ο νέος πίνακας
    ** που κρατάει τον δείκτη επίδοσης για κάθε κλάση.
    **********************************************************************
    Mloc = computeAvgNumOfLocalMisses();
    MDasd = computeAvgNumOfDASDMisses();
    S = computeProcessingRequirements();
    Util = computeProcessorUtilization();
    NewR = computeResponseTime();
    AvgR = computeAvgResponseTime();
    NewPerfIndex = computePerformanceIndex();
    /*
    * Η συνάρτηση comparePerfIndexes() συγκρίνει δύο διανύσματα και
    * επιστρέφει 1 ή 2 ανάλογα με πιο διάνυσμα είναι πιο "μικρό".
    */
    comparison = comparePerfIndexes(PerfIndex, NewPerfIndex);
    IF (comparison == 2)
        /* Αντικατέστησε τον πίνακα δρομολόγησης */
        {PerfIndex} ← {NewPerfIndex};
        {p} ← {q};
        {already_caclulated} ← TRUE;
        {Improvement} ← TRUE;
        {Prob_changed} ← TRUE;
        {mult2} ← 0.8;
    end
    ELSE
        IF (mult2 > 0.0002) {mult2} ← {mult2} / 2.0;
        ELSE
            IF (Improvement ! = TRUE) {high1} ← {high1} + 1;
            IF (high1 == (NumClasses + 1))
                {high1} ← 1;
                {Loops} ← {Loops} + 1;
                IF (Prob_changed == FALSE) {Loops} ← SPECIFIED_LOOPS;
                {Prob_changed} ← FALSE;
            end
            {Improvement} ← FALSE;
            {mult2} ← 0.8;
            {exit} ← TRUE;
            {already_caclulated} ← FALSE;
        end
    end
end
/*
```

```
        * Υπολόγισε τον βέλτιστο αριθμό από δοσοληψίες της κλάσης i στον κόμβο j,
        * με βάση τον πίνακα πιθανοτήτων p.
        */
        Nopt = computeNOptimal();
    end


end
```

Βιβλιογραφία

[Λαμ95a] Αλέξανδρος Λαμπρινίδης. ‘‘Μέθοδοι ταξινόμησης δοσοληψιών σε ομάδες με παρόμοια χαρακτηριστικά φόρτου εργασίας’’. Τμήμα Επιστήμης Υπολογιστών, Σχολή Θετικών Επιστημών, Πανεπιστήμιο Κρήτης, Τ.Θ. 1470, Ηράκλειο, Ελλάς, Αύγουστος 1995.

[Μαρ95b] Μανόλης Μαραζάκης. ‘‘Προσομοίωση Συστημάτων Επεξεργασίας Δοσοληψιών και Μελέτη Μεθόδων για την Ικανοποίηση Στόχων Επίδοσης’’. Τμήμα Επιστήμης Υπολογιστών, Σχολή Θετικών Επιστημών, Πανεπιστήμιο Κρήτης, Τ.Θ. 1470, Ηράκλειο, Ελλάς, Σεπτέμβριος 1995.

[Ανα96] Αναστασία Αναστασιάδη. ‘‘Μελέτη Οικονομικών Αλγορίθμων για Κατανομή Φόρτου Εργασιών και Διαχείριση Δεδομένων σε Κατανεμημένα Συστήματα ’’. Τμήμα Επιστήμης Υπολογιστών, Σχολή Θετικών Επιστημών, Πανεπιστήμιο Κρήτης, Τ.Θ. 1470, Ηράκλειο, Ελλάς, Οκτώβριος 1996.

[Κλο97] Χρήστος Κλουκίνας. ‘‘Ομαδοποίηση ”εν πτήσει” δοσοληψιών σε ομάδες με παρόμοια χαρακτηριστικά φόρτου εργασίας’’. Τμήμα Επιστήμης Υπολογιστών, Σχολή Θετικών Επιστημών, Πανεπιστήμιο Κρήτης, Τ.Θ. 1470, Ηράκλειο, Ελλάς, Απρίλιος 1997.

[AAMN97] A. Anastasiadi, M. Artavanis, M. Marazakis, and C. Nikolaou. *‘‘TPsim Tutorial - A Simulator for Distributed Transaction Management Systemsd’’*. Department of Computer Science, University of Crete, Institute for Computer Science, Foundation of Research and Technology - Hellas (FORTH) , Crete, Greece, August 1997.

[BCD+92] K. P. Brown, M. J. Carey, D. J. DeWitt, M. Mehta, and J. F. Naughton. ‘‘Resource Allocation and Scheduling for Mixed Database Workloads’’. Technical Report TR 1095, Univ. of Wisconsin, Madison, July 1992.

[BCL93] K. P. Brown, M. J. Carey, and M. Livny. ‘‘Managing Memory to Meet Multiclass Workload Response Time Goals’’. In *Proceedings of the 19-th International VLDB Conference*, 1993. Also available as Technical Report No. 1146, University of Wisconsin.

[Ber90] P. Bernstein. ‘‘Transaction Processing Monitors’’. *Communications of the ACM*, 33(11), November 1990.

[BG81] P. Bernstein and N. Goodman. ‘‘Concurrency Control in Distributed Database Systems’’. *ACM Computing Surveys*, 13(2):185--222, 1981.

[BGTV90] P. Bhattacharya, L. Georgiadis, P. Tsoucas, and I. Viniotis. Optimality and Finite Time Behavior of an Adaptive Multi-Objective Scheduling Algorithm. In *Proc. of the 29th Conference on Decision and Control*, December 1990.

[BHG87] P. A. Bernstein, V. Hadzilacos, and N. Goodman. ''*Concurrency Control and Recovery in Database Systems*''. Addison-Wesley, 1987.

[Bhi88] Anupam Bhide. ''An Analysis of Three Transaction Processing Architectures''. In *Proceedings of the 14th Conference on Very Large Databases, Morgan Kaufman pubs. (Los Altos CA), Bancilhon and DeWitt(eds), Los Angeles*, August 1988.

[BHR90] V. Bohn, T. Haerder, and E. Rahm. ''Extended Memory Support for High Performance Transaction Systems''. Technical report, University of Kaiserslautern, 1990.

[BMCL94] K. P. Brown, M. Mehta, K. J. Carey, and M. Livny. ''Towards Automated Performance Tuning for Complex Workloads''. In *Proceedings of the 20-th International VLDB Conference*, pages 578--587, 1994.

[BS87] A. Bhide and M. Stonebraker. ''Performance Issues in High Performance Transaction Processing Architectures''. In *Proceedings of the 2nd International Workshop on High Performance Transaction Processing*, October 1987.

[BS88] A. Bhide and M. Stonebraker. ''A Performance Comparison of Two Architectures for Transaction Processing ''. In *Proceedings of the 4th International Conference on Data Engineering*, February 1988.

[CDY86] D. W. Cornell, D. M. Dias, and P. S. Yu. ''On Multisystem Coupling Through Function Request Shipping''. *IEEE Transactions on Software Engineering (SE)*, SE-12(10):1006--1017, October 1986.

[CF86] M. Calzarossa and D. Ferrari. ''A Sensitivity Study of the Clustering Approach to Workload Modeling''. *Performance Evaluation*, 6:25--33, 1986.

[CFW+94] J. Y. Chung, D. Ferguson, G. Wang, C. Nikolaou, and J. Teng. ''Goal Oriented Dynamic Buffer Pool Management for Data Base Systems''. Technical Report RC 19807, IBM T.J. Watson Research Center, October 1994. Also published as FORTH-ICS TR-125, October 94.

[CHS88] M. Calzarossa, G. Haring, and G. Serrazi. ''Workload Modeling for Computer Networks''. In U. Kastens and F. J. Ramming, editors, *Architekture und Betrieb von Rechensystemen*, pages 324--339. Springer-Verlag, 1988.

[CKB89] E.I. Cohen, G.M. King, and J.T. Brady. ''Storage Hierarchies''. *IBM Journal*, 28(1):62--76, 1989.

[Cou90a] Transaction Processing Council. ''*TPC Benchmark A : Standard Specification*'', 1990.

[Cou90b] Transaction Processing Council. ''*TPC Benchmark B : Standard Specification*'', 1990.

[Cou93] Transaction Processing Council. *''TPC Benchmark C : Standard Specification''*, 1993.

[Cou95] Transaction Processing Council. *''TPC Benchmark D : Standard Specification''*, 1995.

[CS93] M. Calzarossa and G. Serazzi. ''Workload Characterization: A Survey''. *Proceedings of the IEEE*, 81(8), August 1993.

[Dav78] C.T. Davies. ''Data Processing Spheres of Control''. *IBM Systems Journal*, 17(2):179--198, 1978.

[DDY94] A. Dan, D. M. Dias, and P. S. Yu. ''Buffer Analysis for a Data Sharing Environment with Skewed Data Access''. *IEEE Transactions on Knowledge and Data Engineer*, 6(2):331, 1994.

[DG92] D. DeWitt and J. Gray. ''Parallel Database Systems: The Future of High Performance Database Processing''. *Communications of the ACM*, 35(6), June 1992.

[DGMH+93] U. Dayal, H. Garcia-Molina, M. Hsu, B. Kao, and M.-C. Shan. ''Third Generation TP Monitors: A Database Challenge''. In *Proc. ACM SIGMOD Conf.*, page 393, Washington, DC, May 1993.

[DIRY89] D. M. Dias, B. R. Iyer, J. T. Robinson, and P. S. Yu. ''Integrated Concurrency-Coherency Controls for Multisystem Data Sharing''. *IEEE Transactions on Software Engineering (SE)*, 15(4), April 1989.

[DY93] A. Dan and P. S. Yu. ''Performance Analysis of Buffer Coherency Policies in a Multisystem Data Sharing Environment''. *IEEE Transactions on Parallel and Distributed Systems*, 4(3):289--305, March 1993.

[DZ88] Ding-Zhudu. In *''Gradient Projection Method in Linear and Nonlinear Programming ''*. Hadronic Press, Incorporated, January 1988.

[ESP] ESPRIT III Basic Research Action Project 8144. ''Load Balancing on High Performance Parallel and Distributed Systems''. Dr. Christos Nikolaou is the coordinator of the Lydia project. More information about the project can be found at the URL: http://www.ics.forth.gr/proj/pleiades/projects/LYDIA/.

[Fer84] D. Ferrari. ''On the Foundations of Artificial Workload Design''. In *Proc. ACM SIGMETRICS Conf. on Measurement and Modeling of Computer Systems*, pages 8--14, 1984.

[FNGD93a] D. Ferguson, C. Nikolaou, L. Georgiadis, and K. Davies. ''Goal Oriented, Adaptive Transaction Routing for High Performance Transaction Processing Systems''. In *Proc. 2nd Int. Conf. on Parallel and Distributed Information Systems*, San Diego, CA, January 1993.

[FNGD93b] D. Ferguson, C. Nikolaou, L. Georgiadis, and K. Davies. ''Satisfying Response Time Goals in Transaction Processing Systems''. In *Proc. 2nd Int. Conf. on Parallel and Distributed Information Systems*, pages 138--147, January 1993.

[FNY89] D. Ferguson, C. Nikolaou, and Y. Yemini. ''Microeconomic Algorithms for Dynamic Load Balancing in Distributed Computer Systems''. Technical Report Research Report RC15034, IBM, 1989.

[FNY92] D. Ferguson, C. Nikolaou, and Y. Yemini. ''An Economy for Managing Replicated Data in Autonomous Distributed Systems''. Technical Report Research Report RC18164, IBM, 1992.

[FP95] D. Ferguson and P. Bhattacharya. ''Data Sharing Routing Algorithm (in preparation)'', 1995.

[GHK+95] G. Georgiannakis, C. Houstis, S. Kapidakis, M. Karavassili, C. Nikolaou, A. Labrinidis, M. Marazakis, E. Markatos, M. Mavronicolas, S. Chabridon, E. Gelenbe, E. Born, L. Richter, and R. Riedl. ''Description of the Adaptive Resource Management Problem, Cost functions and Performance objectives''. Technical Report 130, ICS/FORTH, Heraklion, Crete, Greece, April 1995. Deliverable WP1/T.1.1-T.1.2/D1 of project LYDIA (available online at ftp.ics.forth.gr:/tech-reports/1995/1995.TR130.LYDIA.D1.ps.gz).

[GR93] J. Gray and A. Reuter. *''Transaction Processing: Concepts and Techniques''*. Morgan Kaufmann, 1993.

[Gra91] J. Gray. *''The Benchmark Handbook for Database and Transaction Processing Systems''*. Morgan Kaufmann, 1991.

[LY91] A. Leff and P. S. Yu. ''An Adaptive Strategy for Load Sharing in Distributed Database Environments with Information Lags''. *Journal of Parallel and Distributed Computing*, 13(1):91--103, September 1991.

[LYL88] A. Leff, P. S. Yu, and Y. H. Lee. ''Adaptive Transaction Routing in a Heterogeneous Database Environment''. Technical Report Research Report RC14114, IBM, 1988.

[MN91] C. Mohan and I. Narang. ''Recovery and Coherency-Control Protocols for Fast Intersystem Page Transfer and Fine-Granularity Locking in a Shared Disk Transaction Enviroment''. *Proceedings of the 17th International Conference on Very Large Data Bases, Barcelona*, 17:193--207, September 1991.

[MN92] C. Mohan and I. Narang. ''Data Base Recovery in Shared Disks and Client-Server Architectures''. *Proc. 12th International Conference on Distributed Computing Systems, Yokohama*, June 1992.

[MNS91] C. Mohan, I. Narang, and S. Silen. ''Solutions to Hot Spot Problems in a Shared Disks Transaction Enviroment''. *In Proceedings of the 4th International Workshop on High Performance Transaction Systems, Asilomar*, 1991.

[MR92] R. Marek and E. Rahm. '' Performance Evaluation of Parallel Transaction Processing in Shared Nothing Database Systems ''. In *4th Int. PARLE Conference*, Paris, June 1992.

[MR94] Lory D. Molesky and Krithi Ramamritham. ‘‘Efficient Locking for Shared Memory Database Systems’’. Technical Report UM-CS-1994-010, Computer Science Department, University of Massachusetts, March 94.

[MR95] Lory D. Molesky and Krithi Ramamritham. ‘‘Recovery Protocols for Shared Memory Database Systems’’. Technical report, Computer Science Department, University of Massachusetts, Amherst, 95.

[Nei91] J. E. Neilson. ‘‘PARASOL: A Simulator for Distributed and/or Parallel Systems.’’. Technical Report SCS-TR-192, Carleton University, Canada, 1991.

[NLB+97] C. Nikolaou, A. Labrinidis, V. Bohn, M. Artavanis, C. Kloukinas, M. Marazakis, and D. Ferguson. ‘‘The Impact of Workload Clustering on Transaction Routing (in preparation)’’, 1997.

[PMC+90] H. Pirahesh, C. Mohan, J Cheng, T.S. Liu, and P. Selinger. ‘‘Parallelism in Relational Data Base Systems: Architectural Issues and Design Approaches’’. In *2nd Intrnational Symposium, Databases in Parallel and Distributed Systems*, pages 4--29, 1990.

[Rah86] E. Rahm. ‘‘Primary Copy Synchronization for DB-Sharing’’. *Information Systems*, 11(4):275, 1986.

[Rah91a] E. Rahm. ‘‘Recovery Concepts for Data Sharing Systems’’. In *Proceedings of the 21st International Symposium on Fault-Tolerant Computing*, pages 368--377, Montreal, PQ CDN, 1991.

[Rah91b] Erhard Rahm. ‘‘ Use of Global Extended Memory for Distributed Transaction Processing ’’. In *4th Int. Workshop on High Performance Transaction Systems*, Asilomar, CA, September 1991.

[Rah92a] E. Rahm. ‘‘A Framework for Workload Allocation in Distributed Transaction Processing Systems’’. *J. Systems Software*, 18:171--190, 1992.

[Rah92b] Erhard Rahm. ‘‘Performance Evaluation of Extended Storage Architecture for Transaction Processing’’. *SIGMOD record.*, 21(2):308, 1992.

[Rah93a] Erhard Rahm. ‘‘ Evaluation of Closely Coupled Systems for High Performance Database Processing ’’. In *13th Int. Conference on Distributed Computing Systems*, Pittsburgh, USA, May 1993.

[Rah93b] Erhard Rahm. ‘‘Empirical Performance Evaluation of Concurrency and Coherency Control Protocols for Database Sharing Systems’’. *ACM Transactions on Database Systems*, 18(2):333--377, June 1993.

[Rah93c] Erhard Rahm. ‘‘Parallel Query Processing in Shared Disk Database Systems’’. *Sigmod record.*, 22(4):32, 1993.

[Rah96] Erhard Rahm. ‘‘ Dynamic Load Balancing in Parallel Database Systems’’. In *In Proc. EURO-PAR 96 Conf., LNCS, Springer-Verlag*, Lyon, August 1996.

[Rah97] E. Rahm. '' Goal Oriented Performance Control for Transaction Processing''. In *In Proceedings of the 9th ITG/GI MMB, VDE-Verlag*, 1997.

[RM93] E. Rahm and R. Marek. ''Analysis Of Dynamic Load Balancing Strategies for Parallel Shared Nothing Database Systems''. In *Proceedings of the 19th Conference on Very Large Databases, Morgan Kaufman pubs. (Los Altos CA), Dublin*, August 1993.

[RM95] E. Rahm and R. Marek. '' Dynamic Multi-Resource Load Balancing in Parallel Database Systems''. In *In Proceedings of the 21th VLDB Conference*, Zurich, Switzerland, 1995.

[RS95] E. Rahm and T. Stohr. '' Analysis of Parallel Scan Processing in Shared Disk Database Systems''. In *In Proc. EURO-PAR 95 Conf., LNCS, Springer-Verlag*, Stockholm, August 1995.

[Sch94] H. Schwetman. *''CSIM User's Guide''*. Mesquite Inc., 1994.

[WDIY89] J. L. Wolf, D. M. Dias, B. R. Iyer, and P. S. Yu. ''Multisystem Coupling by a Combination of Data Sharing and Data Partitioning''. *IEEE Transactions on Software Engineering*, 15(7):854, July 1989.

[WHMZ93] G. Weikum, C. Hasse, A. Monkeberg, and P. Zabback. ''The COMFORT Automatic Tuning Project''. In *Proc. th Intl. Conf. on Management of Data*. ACM, 1993.

[YBL88] P. S. Yu, S. Balsamo, and Y. H. Lee. ''Dynamic Transactions Routing in Distributed Database Systems''. *IEEE Transactions on Software Engineering*, 14(9):1307--1318, September 1988.

[YCDI86] P. S. Yu, D. W. Cornell, D. M. Dias, and B. R. Iyer. ''On Affinity Based Routing in Multi-System Data Sharing''. In *Proc. Int. Conf. on Very Large Data Bases*, page 249, Kyoto, Japan, August 1986.

[YCDT86] P. S. Yu, D.W. Cornell, D.M. Dias, and A. Thomasian. ''On Coupling Partitioned Database Systems''. In *Proceedings of the 6th International Conference on Distributed Computing Systems*, pages 148--157, 1986.

[YCDT89] P. S. Yu, D. W. Cornell, D. M. Dias, and A. Thomasian. ''Performance Comparison of I/O Shipping and Database Call Shipping: Schemes in Multisystem Partitioned Databases''. *Perfomance Evaluation*, 10(1), October 1989.

[YCHL92] P. S. Yu, M. S. Chen, H. U. Heiss, and S. Lee. ''On Workload Characterization of Relational Database Environments''. *IEEE Transactions on Software Engineering*, 18(4):347--355, April 1992.

[YD92a] P. S. Yu and A. Dan. ''Impact of Workload Partitionability on the Performance Coupling Architectures for Transaction Processing''. In *Proc. of the 4th IEEE Int. Symp. on Parallel and Distributed Processing*, pages 40--49, Arlington, Texas, December 1992. IEEE Computer Society Press.

[YD92b] P. S. Yu and D. M. Dias. ''Analysis of Hybrid Concurrency Control Schemes For a High Data Contention Environment''. *IEEE Transactions on Software Engineering*, 18(2):118, February 1992.

[YD94a] P. S. Yu and A. Dan. ''Performance Analysis of Affinity Clustering on Transaction Processing Coupling Architecture''. *IEEE Transactions on Knowledge and Data Engineering*, 6(5):764--786, oct 1994.

[YD94b] P. S. Yu and A. Dan. ''Performance Evaluation of Transaction Processing Coupling Architectures for Handling System Dynamics''. *IEEE Transactions on Parallel and Distributed Systems*, 5(2):139--153, February 1994.

[YDR+87] P. S. Yu, D. M. Dias, J. T. Robinson, B. R. Iyer, and D. W. Cornell. ''On Coupling Multi-Systems Through Data Sharing''. *Proceeding of the IEEE*, 75(5), May 1987.

[YLL91] P. S. Yu, A. Leff, and Y-H. Lee. ''On Robust Transaction Routing and Load Sharing''. *ACM Transactions on Database Systems*, 16(3):476--512, September 1991.