ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

Μέθοδοι ταξινόμησης δοσοληψιών σε ομάδες με παρόμοια χαρακτηριστικά φόρτου εργασίας

Αλέξανδρος Λαμπρινίδης

Μεταπτυχιακή Εργασία

Ηράκλειο, Αύγουστος 1995

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

Μέθοδοι ταξινόμησης δοσοληψιών σε ομάδες με παρόμοια χαρακτηριστικά φόρτου εργασίας

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

Συγγραφέας:

_____
Αλέξανδρος Λαμπρινίδης
Τμήμα Επιστήμης Υπολογιστών

Εισηγητική Επιτροπή:

_____
Χρήστος Νικολάου, Αναπληρωτής Καθηγητής, Επόπτης

_____
Γιώργος Τζιρίτας, Αναπληρωτής Καθηγητής, Μέλος

_____
Πάνος Τραχανιάς, Επίκουρος Καθηγητής, Μέλος

Δεκτή:

_____
Πάνος Κωσταντόπουλος, Αναπληρωτής Καθηγητής
Πρόεδρος Επιτροπής Μεταπτυχιακών Σπουδών

Ηράκλειο, Αύγουστος 1995

Στους γονείς μου,
Στράτο και Άννα

Περίληψη

Η γνώση των χαρακτηριστικών φόρτου εργασίας (workload characteristics) των δοσοληψιών (transactions) είναι σημαντική για την επιτυχία αλγορίθμων δρομολόγησης δοσοληψιών που προσπαθούν να βελτιώσουν τις επιδόσεις ενός συστήματος κατανεμημένης επεξεργασίας δοσοληψιών (OLTP system). Τα χαρακτηριστικά αυτά δεν εξαρτώνται από τους χρόνους προσέλευσης, αλλά περιλαμβάνουν τις προσπελάσεις που κάνουν οι δοσοληψίες στα αρχεία της βάσης, τις υπολογιστικές ανάγκες της δοσοληψίας, και το πλήθος των σημείων συγχρονισμού.

Το *CLUE* είναι ένα περιβάλλον ομαδοποίησης δοσοληψιών με βάση τα χαρακτηριστικά φόρτου εργασίας που παρουσιάζουν. Αντλεί στοιχεία από το ίχνος (trace) της εκτέλεσης των δοσοληψιών σε ένα σύστημα κατανεμημένης επεξεργασίας και τα χρησιμοποιεί για να κατασκευάσει ομάδες από δοσοληψίες με αυξημένη συγγένεια δεδομένων (data affinity). Στα πλαίσια της εργασίας αυτής, εκτός από το *CLUE* , υλοποιήθηκε ένας απλός και γρήγορος ευρηματικός αλγόριθμος ομαδοποίησης για να ανταπεξέλθει στον τεράστιο όγκο δεδομένων (*HALC*), καθώς και ένα εργαλείο (*TSG*) για την κατασκευή συνθετικών ιχνών που χρησιμοποιούνται ως είσοδος για το *CLUE* .

Στις δοκιμές που έγιναν επαληθεύθηκε η σωστή λειτουργία του *CLUE* με τη βοήθεια συνθετικών ιχνών συγκεκριμένων προδιαγραφών και αξιολογήθηκε η λειτουργία του (σε σύγκριση με τους αλγορίθμους ISODATA και Bond Energy) σε πραγματικά δεδομένα. Η σύγκριση των αποτελεσμάτων έδειξε ότι ο αλγόριθμος *HALC* επιτυγχάνει πολύ μικρούς χρόνους εκτέλεσης, δίνοντας παράλληλα ιδιαίτερα ενθαρρυντικά αποτελέσματα ως προς την ποιότητα της ομαδοποίησης.

Ευχαριστίες

Η παρούσα εργασία δεν θα μπορούσε να ολοκληρωθεί χωρίς την ουσιαστική συμβολή τριων ατόμων τα οποία και ευχαριστώ θερμά:

- Τον επιβλέποντα καθηγητή μου, κ. Χρήστο Νικολάου, ο οποίος μου έδωσε τη δυνατότητα να ασχοληθώ με το θέμα αυτό και με καθοδήγησε σε όλη την διάρκεια της εργασίας.
- Τον κ. Volker Bohn, ο οποίος με βοήθησε πολύ, ιδιαίτερα στο ξεκίνημα της εργασίας αυτής, γράφοντας τις βασικές ρουτίνες υποστήριξης του *CLUE* και δίνοντάς μου τεχνικές συμβουλές σε διάφορες φάσεις της εργασίας αυτής.
- Την Μαρία Καραβασίλη, η οποία στήριξε τις προσπάθειες για τη διόρθωση λαθών του *CLUE* , υλοποίησε την δομή αραιών πινάκων καθώς και τον αλγόριθμο Bond Energy.

Επίσης θα ήθελα να ευχαριστήσω τον δρ. Donald Ferguson, τον τρίτο (εκτός από τους Χρήστο Νικολάου και Volker Bohn) της ερευνητικής ομάδας στην IBM όπου πρωτοεμφανίστηκε η αρχική ιδέα για το *CLUE* και τον *HALC* , καθώς και τα μέλη της εξεταστικής επιτροπής της μεταπτυχιακής μου εργασίας, καθηγητές κ. Γιώργο Τζιρίτα και κ. Πάνο Τραχανιά, οι οποίοι με τις χρήσιμες συμβουλές και παρατηρήσεις τους συνέβαλαν στην τελική διαμόρφωση του κειμένου της εργασίας.

Θα ήθελα να ευχαριστήσω και όλα τα μέλη της ερευνητικής ομάδας των ΠΛΕΙΑΔΩΝ (της ομάδας Παράλληλων και Κατανεμημένων Συστημάτων του Ινστιτούτου Πληροφορικής) για τις συμβουλές και την υποστήριξή τους σε όλη τη διάρκεια της εργασίας αυτής. Ιδιαίτερα θα ήθελα να ευχαριστήσω τους: Κοσμά Παπαχρήστο, Μανώλη Μαραζάκη, Τίτο Σαρειδάκη, Γιώργο Γεωργιαννάκη, Χρήστο Κλουκίνα, Σαράντο Καπιδάκη και Πηνελόπη Κωσταντά–Φανουράκη.

Θα ήταν παράλειψη εκ μέρους μου να μην ευχαριστήσω και τους δρ. Thomas Delica και δρ. Eike Born από την ομάδα BS 2000 της SNI Μονάχου, για την παροχή των δύο ιχνών από πραγματικές εφαρμογές, και για τις συμβουλές τους.

Ακόμα, οφείλω να ευχαριστήσω τους γονείς μου, Στράτο Λαμπρινίδη και Άννα Λαμπρινίδου, για τη συμπαράστασή τους και την αμέριστη υποστήριξη που μου έδωσαν καθόλη τη

διάρκεια των σπουδών μου.

Τέλος, θα ήθελα να ευχαριστήσω το Τμήμα Επιστήμης Υπολογιστών του Πανεπιστημίου Κρήτης καθώς και το Ινστιτούτο Πληροφορικής του Ιδρύματος Τεχνολογίας και Έρευνας για την υλικοτεχνική και οικονομική υποστήριξη που μου παρείχαν καθόλη τη διάρκεια των σπουδών μου.

Περιεχόμενα

Περίληψη **i**

Ευχαριστίες **iii**

Περιεχόμενα **v**

Κατάλογος Πινάκων **ix**

Κατάλογος Σχημάτων **xi**

1 Εισαγωγή **1**

2 Επιστημονικό Υπόβαθρο **5**

2.1 Επεξεργασία Δοσοληψιών 5

2.1.1 Η δοσοληψία 5

2.1.2 Συστήματα Επεξεργασίας Δοσοληψιών 7

2.1.3 Κατανεμημένη Επεξεργασία Δοσοληψιών 9

2.1.4 Αρχιτεκτονικές σύζευξης υπολογιστικών κόμβων 11

2.1.5 Δρομολόγηση Δοσοληψιών 14

2.2 Χαρακτηρισμός Φόρτου Εργασίας 17

2.2.1 Διαλογικά Συστήματα 17

2.2.2 Κατανεμημένα Συστήματα 19

2.2.3 Παράλληλα Συστήματα 21

2.2.4 Βάσεις Δεδομένων 23

2.3 Αλγόριθμοι Ομαδοποίησης 25

2.3.1 Κατηγορίες Αλγορίθμων 25

2.3.2 Γνωστοί Αλγόριθμοι 29

2.4 Ομαδοποίηση Φόρτου Εργασίας 32

3 Ομαδοποίηση δοσοληψιών **35**
3.1 Εισαγωγή . . . 35
3.2 Δομή του CLUE . . . 37
3.2.1 Προεπεξεργασία . . . 39
3.2.2 Μέτρηση αποστάσεων . . . 41
3.2.3 Διάκριση αναγνώσεων – ενημερώσεων . . . 44
3.2.4 Αλγόριθμοι . . . 45
3.3 Ο ευρηματικός αλγόριθμος *HALC* . . . 45
3.3.1 Το Βασικό Βήμα Ομαδοποίησης . . . 45
3.3.2 Αναπροσαρμογή απόστασης ομαδοποίησης . . . 47
3.4 Υλοποίηση . . . 51
3.4.1 Γενικά . . . 51
3.4.2 Πίνακας Προσπελάσεων . . . 51
3.4.3 Γλώσσα Περιγραφής Πειραμάτων . . . 53
3.4.4 Βοηθητικά μηνύματα . . . 54
3.4.5 Μεταφερσιμότητα . . . 54

4 Πειράματα **55**
4.1 Συνθετική παραγωγή ιχνών . . . 55
4.2 Σύγκριση & αξιολόγηση ομαδοποιήσεων . . . 57
4.2.1 Σύγκριση ομαδοποιήσεων . . . 57
4.2.2 Αξιολόγηση ομαδοποιήσεων . . . 59
4.3 Πειράματα . . . 60
4.3.1 Συνθετικά κατασκευασμένα ίχνη . . . 60
4.3.2 Ίχνη από πραγματικές εφαρμογές . . . 66

5 Συμπεράσματα και μελλοντικές κατευθύνσεις **69**
5.1 Συμπεράσματα . . . 69
5.2 Επεκτάσεις του *CLUE* . . . 70
5.2.1 Αξιολόγηση . . . 70
5.2.2 Άλλοι αλγόριθμοι . . . 70
5.2.3 Διάφορες βελτιώσεις . . . 70
5.3 Μελλοντικές κατευθύνσεις . . . 71
5.3.1 Μεγάλος όγκος δεδομένων . . . 71
5.3.2 On the fly clustering . . . 71
5.3.3 Workflow . . . 71

5.3.4 Mixed workloads 72

A TSG Reference Input File **73**

B CLUE Reference Input File **79**

Γ SCRIPT Reference Input File **87**

Ελληνοαγγλικό Ευρετήριο Όρων **91**

Αγγλοελληνικό Ευρετήριο Όρων **97**

Βιβλιογραφία **103**

Κατάλογος Πινάκων

3.1 Πίνακας Προσπελάσεων 41
3.2 Πίνακας αποστάσεων PLDM 42
3.3 Πίνακας αποστάσεων VDM 43

4.1 Γενική μορφή πίνακα εγγύτητας 58
4.2 Παράδειγμα πίνακα εγγύτητας 58
4.3 Συγκριτικά αποτελέσματα εκτέλεσης των 3 αλγορίθμων για διαγώνια ίχνη με σταθερό πλήθος προσπελάσεων 64
4.4 Συγκριτικά αποτελέσματα εκτέλεσης των 3 αλγορίθμων για διαγώνια ίχνη με τυχαίο πλήθος προσπελάσεων 65
4.5 Συγκριτικά αποτελέσματα εκτέλεσης των 3 αλγορίθμων στο ίχνος DOA 66
4.6 Συγκριτικά αποτελέσματα εκτέλεσης των 3 αλγορίθμων στο ίχνος PULS 67

Κατάλογος Σχημάτων

2.1 Αρχιτεκτονική Εφαρμογών Επεξεργασίας Δοσοληψιών 8
2.2 Αρχιτεκτονική Shared Disk 11
2.3 Αρχιτεκτονική Shared Intermediate Memory 12
2.4 Αρχιτεκτονική Shared Everything 13
2.5 Αρχιτεκτονική Shared Nothing 14
2.6 Γράφημα Συμπεριφοράς Χρήστη 18
2.7 Κατανεμημένο Σύστημα Συλλογής Μετρήσεων 19
2.8 Γράφημα Εργασιών 22
2.9 Παράδειγμα Δενδρογράμματος 27
2.10 Εύκολη περίπτωση διαμέρισης 29
2.11 Δύσκολη περίπτωση διαμέρισης 29

3.1 Το *CLUE* εν δράσει 36
3.2 Πίνακας Προσπελάσεων 37
3.3 Λειτουργία του *CLUE* 38
3.4 Πίνακας Προσπελάσεων χωρισμένος σε Διαστήματα 46
3.5 Λύση προβλήματος ping−pong 50
3.6 Αραιός Πίνακας Προσπελάσεων 52

4.1 Πίνακας Προσπελάσεων στο *TSG* 56
4.2 Συνθετικά ίχνη τύπου plateau 60
4.3 Συνθετικά ίχνη τύπου columns 61
4.4 Συνθετικά ίχνη τύπου rows 61
4.5 Συνθετικά ίχνη τύπου diagonalsquares 62
4.6 Χρόνοι εκτέλεσης για diagonalsquares με σταθερές τιμές προσπελάσεων 64
4.7 Χρόνοι εκτέλεσης για diagonalsquares με τυχαίες τιμές προσπελάσεων 65

Κεφάλαιο 1

Εισαγωγή

Το πρώτο σύστημα γραφής για επεξεργασία δοσοληψιών (transaction processing) ανακαλύφθηκε από τους Σουμέριους έξι χιλιάδες χρόνια πριν. Ήταν οι πλάκες από πηλό που τις χρησιμοποιούσαν για να καταγράφουν τις εμπορικές δοσοληψίες που κάνανε. Οι πλάκες πηλού αργότερα αντικαταστάθηκαν από φύλλα παπύρου και τελικά χαρτιού. Στα τέλη του 19ου αιώνα επήλθε δραστική αλλαγή στο τρόπο επεξεργασίας δοσοληψιών με την εισαγωγή της μηχανής επεξεργασίας διάτρητων καρτών από τον Hollerith. Ακολούθησαν τα συστήματα μαζικής επεξεργασίας δοσοληψιών (batch systems) το δεύτερο μισό του 20^{ου} αιώνα και κατόπιν τα online συστήματα επεξεργασίας δοσοληψιών

Συνολικά, μπορεί κανείς να πει ότι ο μηχανισμός των δοσοληψιών θα μπορούσε να αντιστοιχισθεί με τα νομικά συμβόλαια στην ανθρώπινη κοινωνία: βρίσκονται για να ξεκαθαρίσουν την κατάσταση στην περίπτωση που "κάτι πάει στραβά" ([GR93]). Γι' αυτό και η έννοια της δοσοληψίας είναι πολύ παλιά και βαθιά ριζωμένη στις καθημερινές εμπορικές πρακτικές.

Σήμερα, τα συστήματα βάσεων δεδομένων με online επεξεργασία δοσοληψιών (OLTP) κυριαρχούν σε όλα τα επίπεδα της ανθρώπινης δραστηριότητας. Οι απαιτήσεις μάλιστα είναι τόσο μεγάλες, ώστε αρχίζει και διαφαίνεται η ανάγκη για κατανεμημένα συστήματα επεξεργασίας δοσοληψιών. Τα προβλήματα που υπάρχουν σε αυτή την προσέγγιση είναι η ανάγκη για επιπλέον συντονισμό (για να προστατευθούν οι ταυτόχρονες προσπελάσεις στα ίδια δεδομένα) καθώς και για δίκαιο καταμερισμό του φόρτου εργασίας μεταξύ των κόμβων του.

Για να υπάρχει όμως ικανοποιητική επίδοση (performance) του συστήματος, θα πρέπει να υπάρχει εξισορρόπηση του φόρτου εργασίας (load balancing) μεταξύ των κόμβων του συστήματος. Το πρόβλημα της εξισορρόπησης του φόρτου εργασίας είναι συνυφασμένο με το πρόβλημα του χαρακτηρισμού του φόρτου εργασίας (workload characterization), αφού χρειάζεται να μπορεί κανείς να ''μετράει'' το φόρτο του κάθε συστήματος για να μπορεί να τον

μοιράσει εξίσου.

Ιδιαίτερα στις αρχιτεκτονικές Shared Nothing (όπου οι κόμβοι του συστήματος διατηρούν ιδιωτικά τμήματα της βάσης δεδομένων) χρειάζεται να είναι γνωστή η *συγγένεια δεδομένων* (data affinity) που παρουσιάζουν (όσον αφορά τις προσπελάσεις δεδομένων) οι δοσοληψίες με τους κόμβους του συστήματος.

Υπάρχουν επομένως δύο αντικρουόμενες ανάγκες:

- Η ανάγκη να εκτελούνται οι δοσοληψίες στους κόμβους όπου έχουν την μεγαλύτερη *συγγένεια δεδομένων*, δηλαδή "κοντά" στα δεδομένα που χρειάζονται. Με τον τρόπο αυτό οι περισσότερες προσπελάσεις των δοσοληψιών στη βάση δεδομένων θα είναι τοπικές και άρα η παροχή (throughput) θα είναι αυξημένη. Αυτό επιτυγχάνεται με την ομαδοποίηση (clustering) των δοσοληψιών με παρόμοιες προσπελάσεις και τη δρομολόγησή τους (routing) στον κόμβο που έχει το σχετικό τμήμα της βάσης δεδομένων.

- Η ανάγκη να εκτελούνται *ταυτόχρονα* όσον δυνατόν περισσότερες δοσοληψίες, δηλαδή να εκτελούνται οι δοσοληψίες όσο το δυνατόν πιο "μακρυά" (να εκτελούνται με άλλα λόγια σε διαφορετικούς κόμβους). Με τον τρόπο αυτό μειώνεται ο χρόνος απόκρισης (response time) των δοσοληψιών. Αυτό επιτυγχάνεται με τον διασκορπισμό (declustering) των δεδομένων σε διαφορετικούς κόμβους και την ανάλογη δρομολόγηση των δοσοληψιών.

Η γνώση επομένως των χαρακτηριστικών φόρτου εργασίας των δοσοληψιών είναι σημαντική για την επιτυχία αλγορίθμων δρομολόγησης δοσοληψιών που προσπαθούν να αυξήσουν τις επιδόσεις του συστήματος. Τα χαρακτηριστικά αυτά δεν εξαρτώνται από τους χρόνους προσέλευσης, αλλά περιλαμβάνουν τις προσπελάσεις που κάνουν οι δοσοληψίες στα αρχεία της βάσης, τις υπολογιστικές ανάγκες της δοσοληψίας, και το πλήθος των σημείων συγχρονισμού.

Στα [YD92] και [YD94] παρουσιάζονται οι διαφορές που υπάρχουν ανάμεσα στις τρεις αρχιτεκτονικές σύζευξης (coupling architectures) υπολογιστικών κόμβων για κατανεμημένα συστήματα επεξεργασίας δοσοληψιών, όταν χρησιμοποιείται ομαδοποίηση με βάση συγγένεια σε δεδομένα (affinity clustering) για να αντισταθμιστούν οι αρνητικές επιπτώσεις στην επίδοση του συστήματος λόγω της κατανεμημένης αρχιτεκτονικής του.

Το *CLUE* (που κατασκευάστηκε στα πλαίσια αυτής της εργασίας) είναι ένα περιβάλλον ομαδοποίησης δοσοληψιών (Clustering Environment) με βάση τα χαρακτηριστικά φόρτου εργασίας που παρουσιάζουν. Αντλεί στοιχεία από το ίχνος (trace) της εκτέλεσης των δοσοληψιών σε ένα σύστημα επεξεργασίας και τα χρησιμοποιεί για να κατασκευάσει ομάδες από δοσοληψίες με αυξημένη συγγένεια δεδομένων (που κάνουν δηλαδή παρόμοιες προσπελάσεις στη βάση δεδομένων).

Η βασική ιδέα πίσω από το *CLUE* είναι ότι σε μεγάλες εφαρμογές, η γεωγραφική και οργανωτική δομή μιας εταιρίας έχει μεγάλη επίδραση στον τρόπο με τον οποίο αποθηκεύονται και προσπελαύνονται τα δεδομένα. Κάποιοι χρήστες θα χρησιμοποιούν ένα περιορισμένο σύνολο από τερματικά για να προσπελάσουν κάποια δεδομένα, κάτι το οποίο σχετίζεται με την καθημερινή τους εργασία και τον τόπο δουλειάς (π.χ. ο τραπεζικός λογαριασμός που σχετίζεται με το υποκατάστημα της τράπεζας που δουλεύει ο υπάλληλος).

Για να είναι αποτελεσματικός ένας χαρακτηρισμός του φόρτου εργασίας ενός συστήματος θα πρέπει αφενός να αντιστοιχεί όσο το δυνατόν περισσότερο με την πραγματικότητα (δηλαδή να έχει αρκετά καλά αποτελέσματα), και αφετέρου να έχει μικρό χρόνο εκτέλεσης, αφού ο όποιος χρόνος εκτέλεσης του εργαλείου χαρακτηρισμού συμψηφίζεται τελικά με τον χρόνο εκτέλεσης στο υπό παρακολούθηση σύστημα.

Τα προβλήματα που παρουσιάζονται στην προσέγγιση χαρακτηρισμού του φόρτου εργασίας που ακολουθείται στο *CLUE* , είναι κυρίως ο μεγάλος όγκος δεδομένων και η δυσκολία στην αξιολόγηση των αποτελεσμάτων. Ο μεγάλος όγκος των δεδομένων (συνήθως έχουμε δεκάδες χιλιάδες δοσοληψιών και εκαντοντάδες χιλιάδες δεδομένων) είναι απαγορευτικός για τη χρήση γνωστών αλγορίθμων ομαδοποίησης, αφού οι χρόνοι εκτέλεσης είναι πολύ μεγάλοι. Έτσι σχεδιάστηκε ένας απλός ευρηματικός αλγόριθμος (*HALC* , από το **H**euristic **Al**gorithm for **C**lustering) ο οποίος έχει μικρό χρόνο εκτέλεσης και αρκετά καλή ποιότητα αποτελεσμάτων. Ένα δεύτερο πρόβλημα με το μεγάλο όγκο των δεδομένων είναι και ο απαιτούμενος χώρος αποθήκευσης. Ο απαιτούμενος χώρος είναι απαγορευτικά μεγάλος για να χωρέσουν όλα μαζί τα δεδομένα (για τις προσπελάσεις των δοσοληψιών) στην κυρίως μνήμη του υπολογιστή, οπότε χρειάζεται μια ιδιαίτερη υλοποίηση και για αυτό.

Η αξιολόγηση των αποτελεσμάτων του *CLUE* ήταν ένα ακόμα πρόβλημα, αφού δεν υπάρχει μια αποδεδειγμένη βέλτιστη διαμέριση του συνόλου των δοσοληψιών σε ομάδες με παρόμοια χαρακτηριστικά φόρτου εργασίας. Δύο ήταν οι λύσεις που δόθηκαν:

- Η υλοποίηση ενός εργαλείου για *παραγωγή συνθετικών παραδειγμάτων*.
  Το *TSG* (από το Test Suite Generator), το οποίο κατασκευάστηκε στα πλαίσια της εργασίας αυτής, χρησιμοποιείται για να δημιουργηθούν παραδείγματα στα οποία οι κλάσεις των δοσοληψιών υπάρχουν και είναι ευδιάκριτες για τον άνθρωπο, οπότε είναι γνωστό εκ των προτέρων ποια θα πρέπει να είναι τα αποτελέσματα του *CLUE* , ώστε να μπορεί να γίνει η επαλήθευση.

- Η σχεδίαση και η υλοποίηση μιας συνάρτησης κόστους για την τελική ομαδοποίηση των δοσοληψιών που παράγει το *CLUE* , ώστε να μπορέσει να γίνει αξιολόγηση των αποτελεσμάτων για πραγματικά δεδομένα.

Η βασική φιλοσοφία που διέπει την υλοποίηση του *CLUE* είναι να μπορεί να χρησιμοποιηθεί το ίδιο περιβάλλον με ελάχιστες ενδεχομένως αλλαγές και από άλλους αλγόριθμους. Στα πλαίσια αυτής της εργασίας δοκιμάστηκαν άλλοι δύο αλγόριθμοι εκτός από τον *HALC* (ο Bond Energy Algorithm [MSW72], και ο ISODATA [VR92]) οι οποίοι ενσωματώθηκαν με επιτυχία στο *CLUE* και μπόρεσαν έτσι να γίνουν συγκριτικές μελέτες.

Επειδή η εργασία απαιτούσε τη διεξαγωγή πολλών πειραμάτων, αναπτύχθηκε μια απλή γλώσσα περιγραφής πειραμάτων για το *CLUE* . Με τη γλώσσα αυτή υπάρχει η δυνατότητα αλλαγής όλων των παραμέτρων του προγράμματος και του εκάστοτε αλγορίθμου, καθώς και η επιλογή του είδους και της ποσότητας των βοηθητικών μηνυμάτων που εμφανίζει το πρόγραμμα. Με τον τρόπο αυτό, αφενός αποφεύγεται η επιπλέον μετάφραση του κώδικα του προγράμματος κάθε φορά που χρειάζεται να αλλαχθεί μια παράμετρος και αφετέρου τεκμηριώνεται το κάθε πείραμα, αφού όλες οι παράμετροι που χρησιμοποιήθηκαν υπάρχουν σε ένα ξεχωριστό αρχείο.

Το *CLUE* μπορεί να χρησιμοποιηθεί από μεγάλες εταιρείες για τον χαρακτηρισμό του φόρτου εργασίας των συστημάτων online επεξεργασίας δοσοληψιών με σκοπό την αύξηση της επίδοσής των συστημάτων. Μπορεί να χρησιμοποιηθεί και ως εργαλείο κατά τη φάση σχεδιασμού κατανεμημένων βάσεων δεδομένων, όπου μπορεί να βοηθήσει στην διαμέριση της βάσης σύμφωνα με τη συγγένεια δεδομένων των διαφόρων ομάδων δοσοληψιών. Υπάρχει επομένως αρκετά μεγάλη πιθανότητα να εξελιχθεί σε εμπορικό προϊόν.

Έχει εκφραστεί προς το παρόν έντονο ενδιαφέρον από τους εταίρους στο project LYDIA[1] για την αξιολόγηση των αποτελεσμάτων. Υπάρχει όμως η εκτίμηση ότι μόλις τελειώσει η φάση της αξιολόγησης θα αρχίσει να υπάρχει εντονότερο ενδιαφέρον.

Στο επόμενο κεφάλαιο υπάρχει μια αρκετά εκτεταμένη εισαγωγή και επισκόπηση της σχετικής βιβλιογραφίας στις τρεις περιοχές των οποίων άπτεται η εργασία, καθώς επίσης και επισκόπηση της βιβλιογραφίας για την ειδικότερη περιοχή της ομαδοποίησης του φόρτου εργασίας. Στο 3ο κεφάλαιο, γίνεται μια αναλυτική περιγραφή τόσο του περιβάλλοντος ομαδοποίησης (*CLUE*) όσο και του ευρηματικού αλγορίθμου που αναπτύχθηκε (*HALC*). Το 4ο κεφάλαιο είναι αφιερωμένο στην παρουσίαση των πειραμάτων και των μεθόδων για την αξιολόγηση των αποτελεσμάτων. Το τελευταίο κεφάλαιο περιέχει τα συμπεράσματα και τις μελλοντικές κατευθύνσεις της εργασίας αυτής. Επιπλέον, υπάρχει ελληνοαγγλικό και αγγλοελληνικό ευρετήριο όρων για να διευκολυνθεί η ανάγνωση του τεχνικού κειμένου στα ελληνικά. Τέλος, τα τρία παραρτήματα περιέχουν παραδείγματα από τη μορφή των αρχείων για τα διάφορα εργαλεία που έχουν φτιαχθεί.

---

[1]Το project LYDIA είναι το ESPRIT III Basic Research Action Project 8144, με τίτλο *"Load Balancing on High Performance Parallel and Distributed Systems"*, με επιστημονικό υπεύθυνο τον Δρ. Χρήστο Νικολάου, *(nikolau@ics.forth.gr)* και είναι η πηγή χρηματοδότησης για το *CLUE* .

Κεφάλαιο 2

Επιστημονικό Υπόβαθρο

Η εργασία άπτεται τριών διαφορετικών περιοχών: Επεξεργασία Δοσοληψιών (Transaction Processing), Χαρακτηρισμός Φόρτου Εργασίας (Workload Characterization), Ομαδοποίηση (Clustering). Στις επόμενες παραγράφους παρουσιάζονται εν συντομία οι τρεις αυτές περιοχές και επιπλέον σε κάθε περιοχή γίνεται αναφορά σε δημοσιεύσεις σχετικές με το θέμα της εργασίας αυτής.

2.1 Επεξεργασία Δοσοληψιών

2.1.1 Η δοσοληψία

Η έννοια της δοσοληψίας (transaction) υπάρχει από πολύ παλιά κυρίως ως η αποτύπωση μιας μεμονωμένης εμπορικής συναλλαγής, μάλιστα δε, η πρακτική αυτή διατηρείται ακόμα και σήμερα. Για παράδειγμα, στα βιβλία εσόδων/εξόδων που τηρούν οι επιχειρήσεις, η κάθε γραμμή αντιστοιχεί σε μια μόνο πράξη, μια μόνο δοσοληψία μεταξύ του επιχειρηματία και κάποιου άλλου. Με τον τρόπο αυτό, για να υπολογίσει ένας επιχειρηματίας το υπόλοιπο που έχει σε χρήματα, θα πρέπει να κάνει έναν *ισολογισμό*, να εφαρμόσει δηλαδή όλες τις δοσοληψίες στην αρχική κατάσταση του συστήματος (δηλαδή στο προηγούμενο υπόλοιπο) και να βρει τη νέα κατάσταση (δηλαδή το καινούριο υπόλοιπο).

Η δοσοληψία στις βάσεις δεδομένων δεν είναι τίποτα άλλο από μια αφαίρεση (abstraction) για τον μετασχηματισμό από μια δοσμένη κατάσταση της βάσης δεδομένων σε μια άλλη. Η χρήση της δοσοληψίας στις βάσεις δεδομένων κατέληξε να είναι αναγκαία γιατί χρησιμοποιείται ως η βασική μονάδα συνεπούς και αξιόπιστου υπολογισμού.

Παρόλο που η έννοια της δοσοληψίας είναι αρκετά γνωστή και σχετικά απλή, η υλοποίησή της σε κάποιο σύστημα επεξεργασίας δοσοληψιών δεν είναι τετριμμένη, αφού θα πρέπει (για την σωστή λειτουργία του συστήματος και την ορθότητα της βάσης δεδομένων) να

διασφαλίζονται οι εξής ιδιότητες:

- **Ατομικότητα** (Atomicity): Όλες οι αλλαγές κατάστασης που προκαλεί η δοσοληψία πρέπει να είναι ατομικές, δηλαδή είτε συμβαίνουν όλες είτε καμία απολύτως.

- **Συνέπεια** (Consistency): Η δοσοληψία πρέπει να είναι μια σωστή "πράξη" πάνω στη βάση δεδομένων, να μην παραβιάζει δηλαδή τους περιορισμούς ακεραιότητας που ενδεχομένως υπάρχουν.

- **Απομόνωση** (Isolation ή Serializability): Παρόλο που είναι δυνατόν να εκτελούνται δύο ή περισσότερες διεργασίες ταυτόχρονα, θα πρέπει να υπάρχει μια σαφής σειρά εκτέλεσης. Θα πρέπει δηλαδή για μια δοσοληψία T που εκτελείται ταυτόχρονα με κάποιες άλλες, να εμφανίζονται οι άλλες δοσοληψίες ότι εκτελούνται είτε πριν, είτε μετά την T, αλλά ποτέ μαζί.

- **Μονιμότητα** (Durability ή Persistency): Όταν μια δοσοληψία τελειώσει επιτυχώς (κάνει δηλαδή commit), τότε οι αλλαγές που έχει προκαλέσει στη βάση δεδομένων θα πρέπει να επιβιώσουν σε οποιαδήποτε βλάβη του συστήματος.

Οι βασικές αυτές ιδιότητες έχουν γίνει γνωστές στη βιβλιογραφία με το όνομα *ACID ιδιότητες* (ACID properties), από τα αρχικά των ονομάτων των ιδιοτήτων.

Το κλασσικό παράδειγμα δοσοληψίας (που χρησιμοποιείται και σαν benchmark [Gra93]) είναι η περίπτωση της *δοσοληψίας χρέωσης/πίστωσης* (Debit/Credit Transaction), όπου ένα χρηματικό ποσό αφαιρείται από έναν τραπεζικό λογαριασμό και προστίθεται σε έναν άλλο.

Στο παράδειγμα αυτό, για να εξασφαλίζεται η ατομικότητα μιας δοσοληψίας, θα πρέπει είτε να γίνουν και οι δύο πράξεις ταυτόχρονα είτε καμία από τις δύο, ενώ για να υπάρχει συνέπεια θα πρέπει να μην γίνεται χρέωση σε λογαριασμό με μηδενικό υπόλοιπο. Επιπλέον, για να υπάρχει απομόνωση, δεν θα πρέπει η οποιαδήποτε δοσοληψία χρέωσης/πίστωσης να μπορεί να διακρίνει ότι ενδεχομένως κάποια άλλη εφαρμογή διαβάζει ή γράφει στον ίδιο τραπεζικό λογαριασμό, ενώ για να να εξασφαλίζεται μονιμότητα, θα πρέπει όταν μεταφερθούν χρήματα από ένα λογαριασμό σε έναν άλλο, τότε, παρόλο που το σύστημα μπορεί να πάθει κάποια βλάβη, η μεταβίβαση αυτή των χρημάτων θα πρέπει να υπάρχει και αφού επισκευαστεί και ξαναλειτουργήσει το σύστημα.

Μία δοσοληψία συνήθως αποτελείται από μια ακολουθία αναγνώσεων και ενημερώσεων που γίνονται σε στοιχεία της βάσης δεδομένων, μαζί με κάποια επιπλέον βήματα υπολογισμών. Επιπλέον, το πρόγραμμα της δοσοληψίας θα πρέπει να αρχίζει και να τελειώνει με κάποιες ειδικές λέξεις–κλειδιά (συνήθως με `BeginTransaction` και `EndTransaction`) ούτως ώστε να μπορέσουν να υλοποιηθούν οι ACID ιδιότητες.

Δεδομένου ότι μια δοσοληψία θα πρέπει να εξασφαλίζει την ατομικότητα, ο τερματισμός μιας δοσοληψίας μπορεί να είναι:

- είτε **επιτυχής**, οπότε η δοσοληψία κάνει `commit` και οι αλλαγές στη βάση δεδομένων είναι πλέον μόνιμες,
- είτε **ανεπιτυχής**, οπότε η δοσοληψία κάνει `abort` διακόπτοντας την εκτέλεσή της και έπειτα γίνεται `rollback` για να αναιρεθούν οι όποιες αλλαγές έγιναν στη βάση δεδομένων.

2.1.2 Συστήματα Επεξεργασίας Δοσοληψιών

Ένα *Σύστημα Επεξεργασίας Δοσοληψιών* (Transaction Processing System) παρέχει εργαλεία για να διευκολύνει ή να αυτοματοποιήσει τον προγραμματισμό, την εκτέλεση και τη διαχείριση εφαρμογών. Οι εφαρμογές επεξεργασίας δοσοληψιών συνήθως υποστηρίζουν ένα δίκτυο από συσκευές που υποβάλλουν ερωτήσεις και κάνουν ενημερώσεις στις εφαρμογές. Βασιζόμενες σε αυτά τα δεδομένα οι εφαρμογές διατηρούν μια βάση δεδομένων που αναπαριστά κάποια κατάσταση από τον πραγματικό κόσμο. Οι απαντήσεις που δίνουν οι εφαρμογές καθοδηγούν συσκευές ενεργοποίησης και επιπλέον αλλάζουν ή ελέγχουν την κατάσταση. Οι εφαρμογές, οι βάσεις δεδομένων και τα δίκτυα εξελίσσονται για πολλές δεκαετίες τώρα. Μάλιστα, τα συστήματα καταλήγουν να είναι όλο και περισσότερο γεωγραφικά κατανεμημένα, ετερογενή (δηλαδή περιλαμβάνουν εξοπλισμό και προγράμματα από πολλούς κατασκευαστές), συνεχώς διαθέσιμα (δηλαδή δεν υπάρχει προγραμματισμένος χρόνος παύσης της λειτουργίας των συστημάτων), και έχουν αυστηρές χρονικές προθεσμίες απόκρισης ([GR93]).

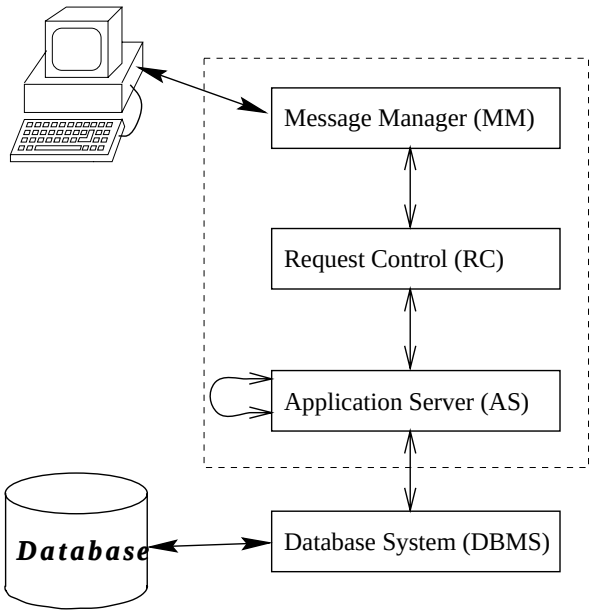
Για να εξασφαλιστούν οι ACID ιδιότητες στις δοσοληψίες που επεξεργάζεται ένα ΣΕΔ χρησιμοποιούνται:

- έλεγχος συνδρομικότητας (Concurrency Control), συνήθως μέσω κλειδώματος (locking), για να εξασφαλιστεί η απομόνωση των δοσοληψιών, και
- μηχανισμοί επαναφοράς (Recovery), συνήθως logging, για να εξασφαλιστεί η μονιμότητα και η ατομικότητα των δοσοληψιών, αφού όλες οι αλλαγές γράφονται σε κάποιο δίσκο και έτσι μπορούν να διατηρηθούν αλλά και να αναιρεθούν (στην περίπτωση που μια δοσοληψία τερματίζει ανεπιτυχώς)

Transaction Monitors

Το βασικότερο συστατικό ενός Σύστηματος Επεξεργασίας Δοσοληψιών είναι το Transaction Monitor το οποίο συντονίζει την ροή των δοσοληψιών μεταξύ τερματικών, άλλων συσκευών

και προγραμμάτων εφαρμογών που μπορούν να εξυπηρετήσουν τις δοσοληψίες. Αφενός, δηλαδή, αποτελεί ένα είδος συνεκτικού συνδέσμου μεταξύ όλων αυτών των ετερόκλητων στοιχείων και αφετέρου δίνει ένα γενικό interface κρύβοντας τις επιμέρους λεπτομέρειες ([Ber90]).



Σχήμα 2.1: Αρχιτεκτονική Εφαρμογών Επεξεργασίας Δοσοληψιών

Για να μπορέσει να έχει τέτοια συμπεριφορά, το Transaction Monitor επιβάλλει μια συγκεκριμένη δομή στις εφαρμογές επεξεργασίας δοσοληψιών που καλείται να εξυπηρετήσει. Έτσι, η μεγάλη πλειοψηφία των εφαρμογών είναι κατάλληλα δομημένες ώστε να επιτελούν τις ακόλουθες λειτουργίες:

1. Αλληλεπίδραση με το τερματικό, ώστε να συλλεχθεί η είσοδος.
2. Μετατροπή της εισόδου στην τυποποιημένη μορφή αιτήσεων (requests).
3. Έναρξη της δοσοληψίας.
4. Εξακρίβωση του τύπου της δοσοληψίας.
5. Εκτέλεση του αντίστοιχου προγράμματος εφαρμογής.
6. Επιτυχής τερματισμός της δοσοληψίας.
7. Αποστολή των αποτελεσμάτων στο τερματικό.

Οι λειτουργίες αυτές ομαδοποιούνται στις ακόλουθες συνιστώσες (σχήμα 2.1).:

- τον Message Manager (MM) που επιτελεί τα βήματα (1), (2) και (7),
- τον Request Control (RC) που επιτελεί τα βήματα (3), (4) και (6), και
- τον Application Server (AS) που επιτελεί το βήμα (5) σε συνεργασία με το (ή τα) DBMS.

2.1.3 Κατανεμημένη Επεξεργασία Δοσοληψιών

Οι ολοένα αυξανόμενες απαιτήσεις σε ταχύτητα επεξεργασίας δοσοληψιών και φόρτο εργασίας, σε συνδυασμό με τα δίκτυα υψηλών ταχυτήτων και την ανάπτυξη των παράλληλων υπολογιστών, έχουν οδηγήσει στη δημιουργία των κατανεμημένων βάσεων δεδομένων όπου η επεξεργασία των δοσοληψιών γίνεται κατανεμημένα.

Τα σύχρονα συστήματα είναι στην πραγματικότητα συστήματα βάσεων δεδομένων και επικοινωνιών (Database / Data Communications Systems ή για συντομία DB/DC), αφού εκτός από τις λειτουργίες που έχουν άμεση σχέση με τη διαχείριση της βάσης δεδομένων, παρέχουν και τη δυνατότητα επικοινωνίας και συντονισμού μεταξύ των διαφόρων υπολογιστικών κόμβων, όμως, σε όλες τις περιπτώσεις, η πρόσβαση στα διάφορα τμήματα της βάσης δεδομένων είναι διάφανη (transparent) στον προγραμματιστή/χρήστη.

Το μοίρασμα δεδομένων και εφαρμογών σε ένα δίκτυο υπολογστικών κόμβων έχει πολλά πλεονεκτήματα ([OV91]):

- Τοπική *αυτονομία*, αφού τα δεδομένα είναι μοιρασμένα και μπορεί έτσι μια ομάδα χρηστών να έχει τα δεδομένα που χρησιμοποιεί συχνά στα δικά της μηχανήματα. Κάτι τέτοιο είναι ιδιαίτερα χρήσιμο σε οργανισμούς με αποκεντρωμένη δομή.
- Αυξημένη *επίδοση*, γιατί αφενός με την κατανομή των δεδομένων μειώνεται η συμφόρηση λόγω I/O, και αφετέρου με την χρήση πολλών υπολογιστικών κόμβων υπάρχει αρκετή παραλληλία και άρα μειώνεται και η συμφόρηση στη CPU.
- Αυξημένη *αξιοπιστία / διαθεσιμότητα*, επειδή τα δεδομένα μπορούν να βρίσκονται (λόγω αντιγραφής) σε πολλά διαφορετικά σημεία στο σύστημα, και έτσι, ακόμα και αν κάποιοι κόμβοι παρουσιάσουν βλάβη, το υπόλοιπο σύστημα θα συνεχίσει να λειτουργεί και με μεγάλη πιθανότητα θα μπορούν να εντοπιστούν όλα τα δεδομένα.
- *Οικονομία* σε κόστος επικοινωνίας (ιδιαίτερα αν μιλάμε για γεωγραφικά κατανεμημένες βάσεις δεδομένων, όπου μπορούμε να μοιράσουμε τα δεδομένα κατάλληλα και να κάνουμε την όποια επεξεργασία τοπικά), αλλά και στο κατασκευαστικό κόστος (αφού το σύστημα μπορεί πια να φτιαχθεί με φτηνότερα κομμάτια και να έχει την ίδια περίπου ισχύ όσο ένα μεγάλο σύστημα με έναν υπολογιστή).

- Μεγάλη *επεκτασιμότητα* αφού ένα κατανεμημένο περιβάλλον μπορεί πιο εύκολα να ανταποκριθεί σε αλλαγές του μεγέθους της βάσης δεδομένων (πχ απλά προσθέτοντας επεξεργαστές και μονάδες αποθήκευσης στο υπάρχον σύστημα).

- Ευκολία στην *κοινή πρόσβαση* δεδομένων μεταξύ διαφορετικών συστημάτων.

Όμως υπάρχουν και αρκετά μειονεκτήματα στις κατανεμημένες βάσεις δεδομένων ([OV91]):

- Η *έλλειψη εμπειρίας* που υπάρχει σε τέτοια συστήματα, αφού τα συστήματα που υπάρχουν σήμερα είτε είναι δοκιμαστικά, είτε έχουν φτιαχθεί για συγκεκριμένες κατηγορίες εφαρμογών, οπότε δεν υπάρχει δοκιμασμένη λύση για γενικές εφαρμογές.

- Η μεγάλη *πολυπλοκότητα* που διακρίνει τα συστήματα αυτά, σε αντίθεση με τα παραδοσιακά κεντρικοποιημένα συστήματα.

- Το επιπλέον *κόστος* που έχουν τα κατανεμημένα συστήματα για την αγορά επικοινωνιακού εξοπλισμού.

- Δυσκολία στον *καταμερισμό ελέγχου* που υπάρχει αφού απαιτείται μεταξύ των κόμβων του συστήματος συντονισμός και ακόμα ακριβής συγχρονισμός για κάποιες περιπτώσεις.

- Προβλήματα *ασφάλειας* επειδή σε αντίθεση με τα παραδοσιακά κεντρικοποιημένα συστήματα, ο έλεγχος δεν μπορεί να γίνει σε ένα μόνο σημείο, και επιπλέον υπάρχουν τα γνωστά προβλήματα της ασφάλειας των δικτύων υπολογιστών.

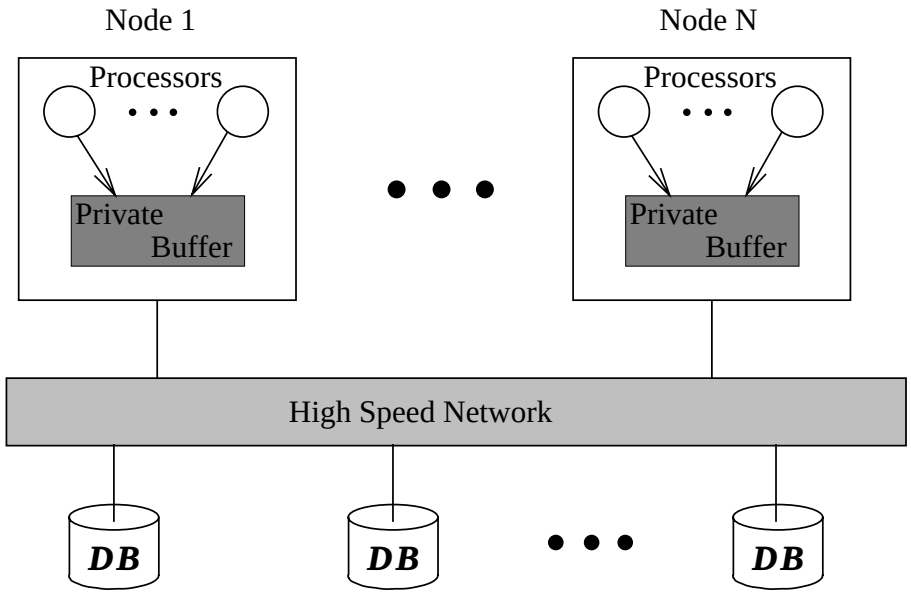
- *Δυσκολία για καθολική αλλαγή* των συστημάτων των περισσότερων εταιριών, οι οποίες έχουν επενδύσει τεράστια χρηματικά ποσά στα παραδοσιακά συστήματα.

Οι παράλληλες βάσεις δεδομένων, μια εναλλακτική προσέγγιση στο πρόβλημα της αυξανόμενης ζήτησης για μεγάλες ταχύτητες επεξεργασίας δοσοληψιών και διαχείριση μεγάλου φόρτου εργασίας, έχουν πολλά κοινά σημεία με τις κατανεμημένες βάσεις δεδομένων. Η βασική διαφορά τους είναι ο βαθμός της σύζευξης (coupling) των διαφόρων υπολογιστικών κόμβων από τους οποίους αποτελούνται: στην περίπτωση των παράλληλων βάσεων δεδομένων οι κόμβοι βρίσκονται σχετικά κοντά (ενδεχομένως στον ίδιο υπολογιστή), ενώ στις κατανεμημένες βάσεις δεδομένων οι κόμβοι μπορεί να είναι τελείως απομακρυσμένοι γεωγραφικά.

Η λύση που προτείνεται με τις παράλληλες βάσεις δεδομένων συνοψίζεται στην παρατήρηση ότι μπορεί κανείς να αυξήσει το εύρος μεταγωγής δεδομένων (I/O Bandwidth) μέσω παραλληλισμού, όμως δυστυχώς υπάρχουν πολλά προβλήματα ακόμα για να μπορούν να χρησιμοποιηθούν σε μεγάλη έκταση ([Val93]).

2.1.4 Αρχιτεκτονικές σύζευξης υπολογιστικών κόμβων

Ο τρόπος με τον οποίο συνδέονται οι διάφοροι υπολογιστικοί κόμβοι για να αποτελέσουν ένα σύστημα Κατανεμημένης Επεξεργασίας Δοσοληψιών δεν είναι μοναδικός. Υπάρχουν 4 διαφορετικές βασικές αρχιτεκτονικές σύζευξης υπολογιστικών κόμβων (Processing Nodes Coupling Architectures): η αρχιτεκτονική Shared Disk, η αρχιτεκτονική Shared Intermediate Memory, η αρχιτεκτονική Shared Everything και η αρχιτεκτονική Shared Nothing ([YD94]).



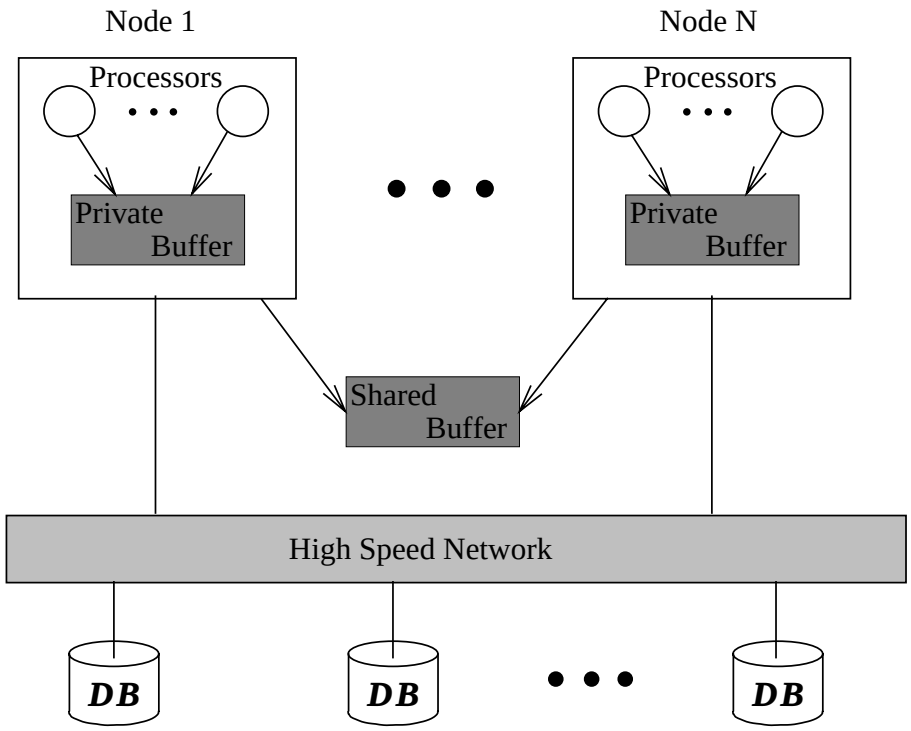
Σχήμα 2.2: Αρχιτεκτονική Shared Disk

Η αρχιτεκτονική Shared Disk (SD)

Σε αυτή την αρχιτεκτονική (σχήμα 2.2), ο κάθε υπολογιστικός κόμβος τρέχει το δικό του ανεξάρτητο λειτουργικό σύστημα και απλά όλοι οι κόμβοι του συστήματος έχουν άμεση πρόσβαση σε ολόκληρη τη βάση δεδομένων (δηλαδή μοιράζονται τους δίσκους στους οποίους βρίσκεται αποθηκευμένη η βάση δεδομένων). Για να καθορίζεται μια σειρά εκτέλεσης των δοσοληψιών υπάρχει ένας καθολικός μηχανισμός ελέγχου συνδρομικότητας, ο οποίος μπορεί να υλοποιηθεί είτε κεντρικά είτε κατανεμημένα.

Ο κάθε κόμβος έχει έναν ιδιωτικό ενταμιευτή (buffer) στον οποίο αποθηκεύονται τα τμήματα της βάσης που έχει προσπελάσει τελευταία ο κόμβος. Όταν γίνει κάποια ενημέρωση σε κάποιο τμήμα της βάσης που βρίσκεται στον ιδιωτικό ενταμιευτή ενός κόμβου, τότε τα αντίγραφα του ίδιου τμήματος της βάσης δεδομένων που βρίσκονται σε ιδιωτικούς ενταμιευτές

άλλων κόμβων ακυρώνονται (invalidated), ενώ παράλληλα ενημερώνεται και ο δίσκος στον οποίο βρίσκεται αποθηκευμένο αυτό το τμήμα της βάσης.



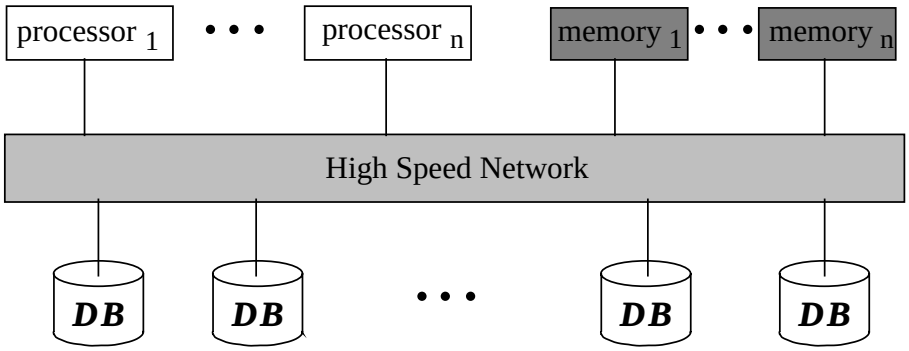
Σχήμα 2.3: Αρχιτεκτονική Shared Intermediate Memory

Η αρχιτεκτονική Shared Intermediate Memory (SIM)

Αυτή η αρχιτεκτονική (σχήμα 2.3) βασίζεται στην αρχιτεκτονική Shared Disk με την προσθήκη ενός ενδιάμεσου επιπέδου μνήμης που δρα ως ένας καθολικός κοινόχρηστος ενταμιευτής (global shared buffer).

Έτσι, στην περίπτωση που κάποιος υπολογιστικός κόμβος αποτύχει να βρει ένα τμήμα της βάσης στον ιδιωτικό του ενταμιευτή, θα ψάξει κατόπιν στον καθολικό κοινόχρηστο ενταμιευτή και μετά θα ψάξει σε κάποιο δίσκο, ενώ και στις δύο περιπτώσεις το τμήμα της βάσης δεδομένων που βρέθηκε θα αντιγραφεί στον ιδιωτικό ενταμιευτή του κόμβου.

Το πλεονέκτημα αυτής της αρχιτεκτονικής είναι ότι μπορεί να μειώσει τις δυσάρεστες συνέπειες της ακύρωσης, αφού όλα τα ακυρωμένα τμήματα της βάσης καταλήγουν στον καθολικό ενταμιευτή. Τέλος, μπορεί κανείς να παρατηρήσει ότι υπάρχει αναλογία με τα συστήματα *client–server*, όπου ο ενταμιευτής του server παίζει ανάλογο ρόλο με τον καθολικό κοινόχρηστο ενταμιευτή της αρχιτεκτονικής SIM.



Σχήμα 2.4: Αρχιτεκτονική Shared Everything

Η αρχιτεκτονική Shared Everything (SE)

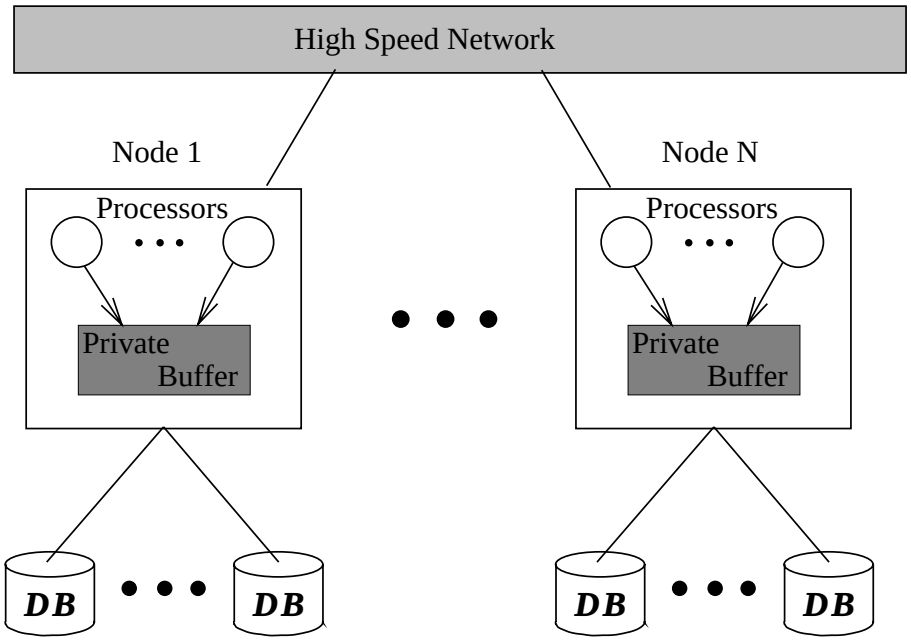
Στην αρχιτεκτονική αυτή (όπως φαίνεται και από το όνομά της) τα πάντα είναι κοινόχρηστα, αφού τόσο οι δίσκοι (κάτι που συμβαίνει και στις δύο προαναφερθείσες αρχιτεκτονικές), αλλά και οι μνήμες−ενταμιευτές είναι κοινόχρηστοι (σχήμα 2.4).

Η αρχιτεκτονική αυτή ονομάζεται αλλιώς και αρχιτεκτονική πολυεπεξεργαστών με ισχυρή σύνδεση (tightly coupled multiprocessor architecture), αφού για να μπορέσει ο πολυεπεξεργαστής να δουλέψει θα πρέπει οι χρονικές καθυστερήσεις στο δίκτυο επικοινωνίας να είναι πολύ μικρές. Η αρχικτονική SE δεν χρησιμοποιείται σαν αρχιτεκτονική για ένα ολόκληρο σύστημα κατανεμημένης επεξεργασίας δοσοληψιών (κυρίως επειδή το πλήθος από επεξεργαστές που μπορεί να υποστηρίξει είναι απαγορευτικά μικρό για τις περισσότερες εφαρμογές), αλλά μπορεί να αποτελέσει το δομικό λίθο σε μεγαλύτερα συστήματα, δηλαδή μπορεί να αποτελέσει την αρχιτεκτονική του υπολογιστικού κόμβου στις υπόλοιπες αρχιτεκτονικές.

Η αρχιτεκτονική Shared Nothing (SN)

Η τελευταία αρχιτεκτονική έχει το ιδιαίτερο χαρακτηριστικό ότι η βάση δεδομένων μοιράζεται αρχικά σε κάποια τμήματα (διαμερίσεις − partitions) ένα για τον κάθε υπολογιστικό κόμβο. Τα τμήματα αυτά αποθηκεύονται στον ιδιωτικό δίσκο (ή τους δίσκους) του κάθε κόμβου και μπορούν να προσπελαστούν άμεσα μόνο από τον κόμβο−ιδιοκτήτη του κάθε τμήματος της βάσης δεδομένων (σχήμα 2.5).

Η αρχιτεκτονική αυτή έρχεται σε άμεση αντίθεση με την αρχιτεκτονική SD, όπου ο οποιοσδήποτε κόμβος μπορεί να έχει άμεση πρόσβαση σε ολόκληρη τη βάση δεδομένων. Για να εξυπηρετηθούν πιθανές ανάγκες για προσπέλαση τμημάτων της βάσης δεδομένων που βρίσκονται σε διαφορετικούς κόμβους θα πρέπει να υπάρχει η δυνατότητα *μεταβίβασης αίτησης*



Σχήμα 2.5: Αρχιτεκτονική Shared Nothing

προσπέλασης (database request shipping). Με τον μηχανισμό αυτό, μια δοσοληψία μπορεί να εκτελείται σε έναν κόμβο και να στέλνει αιτήσεις για προσπέλαση δεδομένων που μπορεί να βρίσκονται σε άλλους κόμβους (χωρίς απαραίτητα να ξέρει η δοσοληψία αν η προσπελάσεις αυτές είναι τοπικές ή απομακρυσμένες, δηλαδή με διαφάνεια).

Η αρχιτεκτονική Shared Nothing είναι η πλέον διαδεδομένη αρχιτεκτονική σύζευξης υπολογιστικών κόμβων για καταναμεμημένη επεξεργασία δοσοληψιών και είναι η αρχιτεκτονική των συστημάτων με τα οποία ασχολείται η εργασία αυτή.

2.1.5 Δρομολόγηση Δοσοληψιών

Εκτός από την αρχιτεκτονική με την οποία είναι φτιαγμένο ένα σύστημα επεξεργασίας δοσοληψιών, ιδιαίτερα σημαντικό από πλευράς επίδοσης του όλου συστήματος είναι και το κομμάτι που είναι υπεύθυνο για τη *δρομολόγηση δοσοληψιών* (transaction routing), το οποίο αποφασίζει σε ποιόν κόμβο θα πρέπει να εκτελεστεί μια νέα δοσοληψία.

Ένας καλός αλγόριθμος δρομολόγησης δοσοληψιών θα πρέπει να επιτυγχάνει υψηλή ταχύτητα επεξεργασίας δοσοληψιών και μικρούς χρόνους απόκρισης. Για να επιτευχθεί κάτι τέτοιο απαιτείται καλή *εξισορρόπηση φόρτου εργασίας* (load balancing) μεταξύ των υπολογιστικών κόμβων του συστήματος. Το θέμα της εξισορρόπησης του φόρτου εργασίας έχει μελετηθεί επανειλημμένα για την περίπτωση κατανεμημένων συστημάτων γενικά ([CK88])

και για την περίπτωση κατανεμημένων λειτουργικών συστημάτων ([Bla90])

Δυστυχώς, η περίπτωση των συστημάτων κατανεμημένης επεξεργασίας δοσοληψιών είναι περισσότερο πολύπλοκη από την γενική περίπτωση, αφού θα πρέπει να συνυπολογιστούν και άλλοι παράγοντες όπως το εύρος μεταγωγής δεδομένων των δίσκων, η συμφόρηση στα δεδομένα λόγω ταυτόχρονων προσπελάσεων, και η ταχύτητα επικοινωνίας μέσω του δικτύου.

Ειδικά για την περίπτωση των αρχικτεκτονικών Shared Nothing, η χρήση δυναμικών αλγορίθμων δεν αποτελεί τη σωστότερη προσέγγιση, αφού για κάθε δοσοληψία υπάρχει μεγάλη συγγένεια δεδομένων με έναν συγκεκριμένο υπολογιστικό κόμβο, επειδή στο συγκεκριμένο κόμβο βρίσκεται σημαντικό ποστοστό από το τμήμα της βάσης δεδομένων που χρειάζεται η δοσοληψία (δεδομένης μιας αρχικής διαμέρισης της βάσης). Έτσι, παρόλο που ένας κόμβος μπορεί να είναι αρκετά φορτωμένος, θα πρέπει οι δοσοληψίες με μεγάλη συγγένεια δεδομένων να καταλήγουν σε αυτόν, αφού ακόμα και αν δρομολογηθούν σε διαφορετικό κόμβο, πολλές από τις προσπελάσεις των δοσοληψιών θα μεταβιβαστούν τελικά (μέσω της δυνατότητας μεταβίβασης αίτησης προσπέλασης) στον αρχικό κόμβο, αυξάνοντας με τον τρόπο αυτό την κίνηση στο δίκτυο και μειώνοντας τη συνολική επίδοση του συστήματος.

Κατηγορίες Αλγορίθμων Δρομολόγησης

Υπάρχουν πολλοί αλγόριθμοι για δρομολόγηση δοσοληψιών, αλλά μπορεί κανείς να διακρίνει τις εξής χαρακτηριστικές κατηγορίες ([GHK+95]) :

1. **Στατικοί Αλγόριθμοι**
   Βασίζονται κυρίως στον τρόπο με τον οποίο έχει διαμεριστεί η βάση δεδομένων στους διάφορους κόμβους και παίρνουν υπόψη τους τη συγγένεια δεδομένων που παρουσιάζουν οι δοσοληψίες με έναν κόμβο για να κατασκευάσουν ένα στατικό πίνακα δρομολόγησης.

   Διακρίνονται σε δύο υποκατηγορίες:

   - Αλγόριθμοι *αιτιοκρατικής δρομολόγησης* (π.χ. [CDY86]) και
   - Αλγόριθμοι *πιθανοτικής δρομολόγησης* (π.χ. [YCDT89]).

2. **Δυναμικοί Αλγόριθμοι** :
   Παρόλο που οι στατικοί αλγόριθμοι είναι αρκετά αποδοτικοί και επιβαρύνουν ελάχιστα το χρόνο απόκρισης της δοσοληψίας (αφού δεν χρειάζεται να μαζέψουν και να αναλύσουν δεδομένα από τη δυναμική κατάσταση του συστήματος), δεν έχουν τη δυνατότητα να αντιδράσουν σε αλλαγές της κατάστασης του συστήματος. Οι δυναμικοί αλγόριθμοι είναι η κατηγορία των αλγορίθμων που παίρνουν υπόψη τους την ολένα μεταβαλλόμενη κατάσταση του συστήματος και μπορούν επομένως να προσαρμοστούν στις όποιες

αλλαγές (π.χ. του φόρτου εργασίας ή της διαμόρφωσης του συστήματος). Το τίμημα που πληρώνει όμως κανείς για να τους χρησιμοποιήσει είναι η επιπλέον καθυστέρηση στο χρόνο απόκρισης της κάθε δοσοληψίας.

Μίγμα στατικού αλγορίθμου και δυναμικού αλγορίθμου είναι ο αλγόριθμος που παρουσιάζεται στο [HS93], αφού χρησιμοποιεί η συγγένεια δεδομένων που παρουσιάζουν οι δοσοληψίες με τους κόμβους ως αρχική πολιτική δρομολόγησης δοσοληψιών, ενώ μόλις διαπιστώσει μεγάλη ανισορροπία φόρτου εργασίας μεταξύ των κόμβων υιοθετεί δυναμική πολιτική εξισορρόπησης.

Γενικά, οι δυναμικοί αλγόριθμοι δρομολόγησης δοσοληψιών διακρίνονται σε τέσσερις υποκατηγορίες:

- Αλγόριθμοι που βασίζονται στο *μέγεθος ουράς* (π.χ. [YBL88])
- Αλγόριθμοι που βασίζονται στην *ιστορία δρομολογήσεων* (π.χ. [YBL88])
- *Παλινδρομικοί* αλγόριθμοι (π.χ. [YL91])
- Λοιποί αλγόριθμοι με *επανατροφοδότηση* (π.χ. [WHMZ93])

3. **Αλγόριθμοι προσανατολισμένοι σε στόχους επίδοσης** :
   Μια ειδική κατηγορία δυναμικών αλγορίθμων αναφέρεται σε αλγόριθμους που είναι προσανατολισμένοι σε συγκεκριμένους στόχους επίδοσης (goal oriented algorithms). Οι στόχοι επίδοσης συνήθως τίθενται από τον υπεύθυνο διαχείρισης του συστήματος (system administrator) και ουσιαστικά αποτελούν μια συμφωνία για την ελάχιστη εξυπηρέτηση (χρόνος απόκρισης) που θα έχουν οι διάφορες κλάσεις δοσοληψιών ([FNGD93]).

4. **Μικροοικονομικοί Αλγόριθμοι** :
   Οι μικροοικονομικοί αλγόριθμοι ([FNY93], [FNY88]) αποτελούν μια εντελώς ξεχωριστή κατηγορία αλγορίθμων δρομολόγησης δοσοληψιών αφού προσπαθούν να εφαρμόσουν ανταγωνιστικά οικονομικά κριτήρια ως πολιτικές δρομολόγησης. Η αντιμετώπιση αυτή έρχεται σε αντίθεση με όλους τους παραδοσιακούς αλγορίθμους (όπου υπάρχει συνεργασία μεταξύ όλων των μερών του συστήματος), επειδή οι κόμβοι και οι δοσοληψίες *ανταγωνίζονται* (σε μικροοικονομικό επίπεδο) για τους πόρους του συστήματος.

2.2 Χαρακτηρισμός Φόρτου Εργασίας

Η ποιοτική και ποσοτική περιγραφή του φόρτου εργασίας ενός συστήματος (που συνοπτικά ονομάζεται *χαρακτηρισμός φόρτου εργασίας* – workload characterization) είναι θεμελιώδης για οποιαδήποτε μελέτη της απόδοσής του.

Τη στιγμή που η συμπεριφορά του πραγματικού φόρτου εργασίας είναι πολύπλοκη και αναπαράγεται δύσκολα, απαιτείται ένα μοντέλο για να την αναπαραστήσει. Ένα τέτοιο μοντέλο θα πρέπει να μπορεί να συλλαμβάνει τόσο τη στατική, όσο και τη δυναμική συμπεριφορά του πραγματικού φόρτου εργασίας του συστήματος. Επιπλέον θα πρέπει να είναι συμπαγές, ακριβές και εύκολο να αναπαράγεται εύκολα.

Μια στρατηγική για να κατασκευάσει κανείς μοντέλα φόρτου εργασίας προτάθηκε στο [CS93] και περιλαμβάνει τα ακόλουθα βήματα:

1. Καταρχήν να ορίσει τη βασική μονάδα φόρτου εργασίας
2. Να αναλύσει τις κατανομές των παραμέτρων του φόρτου εργασίας και να εντοπίσει τους ακρίτες (outliers), δηλαδή τα σημεία με συμπεριφορά σημαντικά διαφορετική από τα υπόλοιπα.
3. Να κάνει δειγματοληψία (sampling) στα δεδομένα των μετρήσεων για να μειώσει το συνολικό όγκο του δείγματος.
4. Να κάνει κάποιου είδους στατιστική ανάλυση που συνήθως μπορεί να τελειώσει με μια στατική ανάλυση (π.χ. ομαδοποίηση των δεδομένων) ή δυναμική ανάλυση (π.χ. χρήση στοχαστικών διαδικασιών).
5. Να αποφασίσει για την αντιπροσωπευτικότητα του υπό κατασκευή μοντέλου.

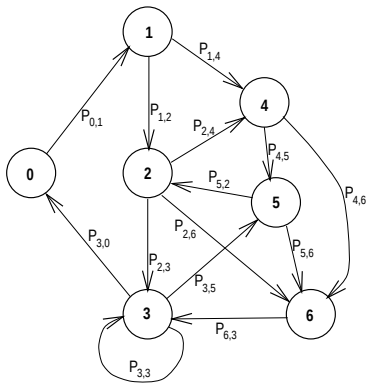
Για τη μελέτη των διαφόρων τύπων μοντέλων φόρτου εργασίας συστημάτων που έχουν προταθεί κατά καιρούς, υιοθετείται η ταξινόμηση που παρουσιάζεται στο [CM94]. Υπάρχουν 4 κύριες κατηγορίες (όχι απαραίτητα ξένες μεταξύ τους) οι οποίες είναι τα Διαλογικά (Interactive) Συστήματα, τα Κατανεμημένα Συστήματα, τα Παράλληλα Συστήματα, και τα συστήματα Βάσεων Δεδομένων. Η διαφορά μεταξύ Κατανεμημένων και Παράλληλων Συστημάτων εντοπίζεται κυρίως στο βαθμό σύζευξης (coupling), όπου τα Κατανεμημένα Συστήματα έχουν περισσότερο χαλαρή σύζευξη από τα Παράλληλα Συστήματα. Μάλιστα, τα Κατανεμημένα Συστήματα συνήθως βρίσκονται γεωγραφικά διασκορπισμένα.

2.2.1 Διαλογικά Συστήματα

Στα διαλογικά συστήματα, σε αντίθεση με τα συστήματα μαζικής επεξεργασίας (batch systems), η δυναμική συμπεριφορά (και ιδιαίτερα τα διάφορα χρονο–μεταβλητά χαρακτηριστικά) του

φόρτου εργασίας τους είναι το κυρίαρχο χαρακτηριστικό για τη δημιουργία οποιουδήποτε μοντέλου.

Η πρώτη προσπάθεια για μοντελοποίηση του φόρτου εργασίας ενός διαλογικού συστήματος έγινε με τη βοήθεια στοχαστικών μοντέλων ([Har83]). Ένα χρόνο αργότερα, η χρήση των Γραφημάτων Συμπεριφοράς Χρήστη (User Behaviour Graphs) παρουσιάστηκε στο [Fer84]. Τα γραφήματα αυτά προσπάθησαν να αναπαραστήσουν το σύνολο των εντολών που μπορεί να δώσει ένας χρήστης ως ένα προσανατολισμένο γράφημα με κορυφές τις διάφορες εντολές προς εκτέλεση και ακμές που έχουν βάρος ίσο με την πιθανότητα ο χρήστης να εκτελέσει την μια εντολή μετά την άλλη (σχήμα 2.6).



Σχήμα 2.6: Γράφημα Συμπεριφοράς Χρήστη

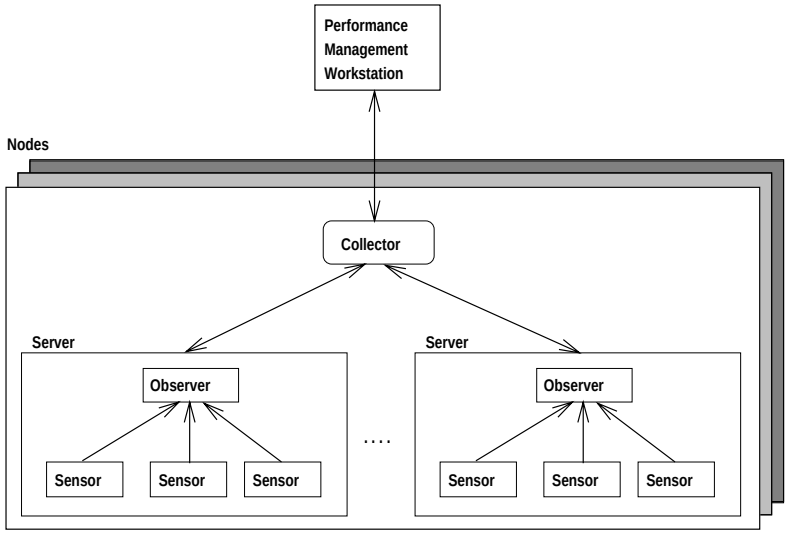
Δυστυχώς, Γραφήματα Συμπεριφοράς Χρήστη αντιπροσωπευτικά των πραγματικών συστημάτων κατέληγαν να έχουν πάρα πολλούς κόμβους και έτσι η όποια προσομοίωση ή υπολογισμός σε αυτά ήταν αδύνατη. Στο [CF86] παρουσιάστηκε η ιδέα της ομαδοποίησης κόμβων του γραφήματος, ώστε να προκύψει τελικά γράφημα με πλήθος κόμβων όχι απαγορευτικό για υπολογισμούς.

Τέλος, η εισαγωγή της έννοιας του παραλληλισμού στα Γραφήματα Συμπεριφοράς Χρήστη είναι μια καινούρια ιδέα και προτάθηκε στο [ACRS94], όπου παρουσιάζεται ένα μοντέλο φόρτου εργασίας αποτελούμενο από ένα παραγωγικό (generative) και ένα εκτελέσιμο (executable) τμήμα. Το παραγωγικό μοντέλο παρέχει μια περιγραφή του φόρτου εργασίας ανεξάρτητη από το σύστημα (system independent) και βασίζεται στη μελέτη της συμπεριφοράς των χρηστών από ένα λογικό σημείο παρατήρησης (logical viewpoint). Μόλις κατασκευαστεί μια πιθανοτική περιγραφή τόσο της σειριακής όσο και της παράλληλης συμπεριφοράς του συστήματος, κατασκευάζεται το εκτελέσιμο μοντέλο, το οποίο έχει και τα συγκεκριμένα χαρακτηριστικά της αρχιτεκτονικής του συστήματος.

2.2.2 Κατανεμημένα Συστήματα

Στα Κατανεμημένα Συστήματα υπάρχει μεγάλη δυσκολία στη συλλογή μετρήσεων, αφού όλα τα δεδομένα συλλέγονται σε τοπικό επίπεδο (σε κάθε κόμβο δηλαδή) και πρέπει να συγκεντρωθούν και να συσχετισθούν, ώστε να αποτελέσουν ένα ‘‘καθολικό’’ ίχνος (trace) της εκτέλεσης.

Πρόσφατα προτάθηκε μια νέα αρχιτεκτονική λογισμικού για εργαλεία παρακολούθησης επίδοσης ετερογενών κατανεμημένων συστημάτων ([FMS+94]). Σε αυτήν την αρχιτεκτονική, ορίζεται μια ιεραρχία με διάφορα επίπεδα στα οποία παρέχεται ολοένα αυξανόμενο πλήθος λεπτομερειών στις μετρήσεις (σχήμα 2.7). Το χαμηλότερο επίπεδο της ιεραρχίας (οι αισθη−τήρες − sensors) είναι υπεύθυνο για τη συλλογή των μετρήσεων από κάθε κόμβο. Τα δεδομένα αυτά συσχετίζονται στο κάθε επίπεδο της ιεραρχίας, και, έτσι στο κορυφαίο επίπεδο (top level) φτιάχνεται μια καθολική εικόνα των αναγκών του συστήματος σε πόρους σε όλους τους κόμβους του δικτύου.



Σχήμα 2.7: Κατανεμημένο Σύστημα Συλλογής Μετρήσεων

Η αντιμετώπιση που ακολουθήθηκε στο project LYDIA[1] στο πρόβλημα αυτό είναι η ακόλουθη ([BDE+95]).

Έστω μια συνεχής ακολουθία από μονάδες εργασίας, πιθανόν τυχαία, που ξεκινάει σε κάθε περιφερειακό ή τερματικό του κατανεμημένου συστήματος. Η επεξεργασία κάθε μονάδας εργασίας από το σύστημα δημιουργεί μια *ιστορία ζωής (life history)*, η οποία εξαρτάται τόσο

[1]ESPRIT III Basic Research Action Project 8144, ‘‘Load Balancing on High Performance Parallel and Distributed Systems’’.

από το σύστημα που την επεξεργάζεται, όσο και από τις υπόλοιπες μονάδες εργασίας που υπάρχουν στο σύστημα εκείνη τη στιγμή.

Οι προαναφερθείσες ιστορίες ζωής είναι πολύ σημαντικές για τη μελέτη δυναμικών αλγορίθμων κατανομής φόρτου εργασίας, επειδή περιέχουν πληροφορία για τις μονάδες εργασίας, τις απαιτήσεις τους σε συγκεκριμένους πόρους του συστήματος (π.χ. χρόνος CPU, πλήθος των χρησιμοποιημένων σελίδων, το σύνολο των bytes που μεταφέρθηκαν, τα locks που αποκτήθηκαν, το πλήθος των SQL κλήσεων, το πλήθος των νημάτων που χρησιμοποιήθηκαν, κ.λ.π.) καθώς και το χρόνο που σπατάλησαν περιμένοντας να ελευθερωθεί κάποιος από τους πόρους του συστήματος.

Σε πολλά συστήματα, οι πληροφορίες για την ιστορία των διαφόρων μονάδων εργασίας μαζεύονται σε αρχεία και μπορεί να τις επεξεργαστεί κανείς off−line. Η συλλογή ιχνών (traces) με τέτοιο τρόπο, όμως, αντιμετωπίζει πολλά προβλήματα:

- Δημιουργεί μεγάλο επιπλέον κόστος στην εκτέλεση των μονάδων εργασίας.
- Απλά κατασκευάζει αρχεία από ίχνη για συγκεκριμένες διεργασίες εξυπηρέτησης σε συγκεκριμένους κόμβους στο κατανεμένο σύστημα και όχι ολοκληρωμένα αρχεία που να δείχνουν τη ροή της μονάδας εργασίας από τις διάφορες διεργασίες εξυπηρέτησης οι οποίες πιθανόν τρέχουν σε διαφορετικούς κόμβους του κατανεμημένου συστήματος.
- Αρχεία με ίχνη υπάρχουν διασκορπισμένα σε ολόκληρο το κατανεμημένο σύστημα και δεν υπάρχει κάποιος εύκολος τρόπος για να εντοπίσει κανείς και κατόπιν να τα συνενώσει και να αξιοποιήσει την πληροφορία αυτή συνολικά.

Για να αντιμετωπιστούν αυτά τα προβλήματα το project LYDIA ([SR95]) προτείνει την σχεδίαση και την υλοποίηση μιας *Γλώσσας Ορισμού Interface* (Interface Definition Language − IDL). Η γλώσσα αυτή θα δίνει τη δυνατότητα σε πληροφορίες για το φόρτο εργασίας να μπορούν να ρέουν από τον ένα διαχειριστή πόρων του συστήματος στον άλλον και από τον έναν κόμβο στον άλλο, αλλά πάντα θα υπάρχει μια σαφής αντιστοίχιση με τη μονάδα εργασίας που τις παράγει. Στο τέλος της ζωής μιας μονάδας εργασίας, οι σχετικές με αυτήν πληροφορίες φόρτου εργασίας μπορούν να αποθηκεύονται σε μια σχετική βάση δεδομένων. Κατόπιν οι πληροφορίες αυτές μπορούν να χρησιμοποιούνται ποικιλοτρόπως, π.χ. από αλγόριθμους κατανομής φόρτου εργασίας και για παρακολούθηση της επίδοσης του συστήματος.

Αφού λύσει κανείς το πρόβλημα της συλλογής των δεδομένων από τις μετρήσεις, μπορεί να αρχίσει να ψάχνει για παραμέτρους που θα μπορέσουν να αντιπροσωπεύσουν το φόρτο εργασίας και τα χαρακτηριστικά του. Διάφορες πειραματικές μελέτες έχουν εστιάσει το ενδιαφέρον τους στην ανάλυση της παρατηρούμενης κίνησης σε διάφορα δίκτυα καθώς και

σε κατανεμημένα συστήματα αρχείων, αλλά ακόμα δεν έχει προταθεί ένα ρεαλιστικό μοντέλο φόρτου εργασίας.

Υπάρχει και μια άλλη κατηγορία μελετών, όπου η προσπάθεια εστιάζεται στην δημιουργία ιεραρχικών μοντέλων φόρτου εργασίας:

- Στο [CHS88] το μοντέλο του φόρτου εργασίας χωρίζεται σε τρία επίπεδα: τα τερματικά των χρηστών, τους υπολογιστικούς κόμβους και το υποσύστημα επικοινωνιών. Ο χωρισμός αυτός προκύπτει από τη λογική διαίρεση των συστατικών του υλικού (hardware) στα κατανεμημένα συστήματα.

- Στο [RVH94] το μοντέλο του φόρτου εργασίας του δικτύου (*Networkload*) χωρίζεται σε τρία επίπεδα: συνδιαλέξεις (sessions), εντολές και αιτήσεις. Ο χωρισμός αυτός φτιάχθηκε για να αντανακλά τον τρόπο με τον οποίο παράγεται ο φόρτος εργασίας.

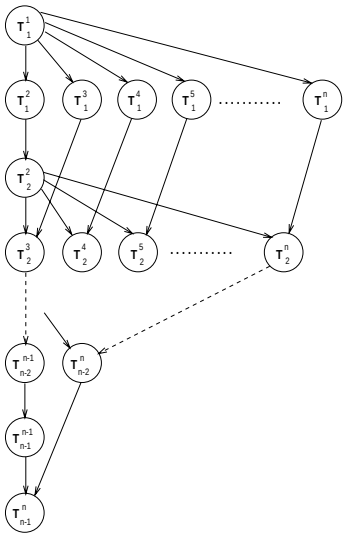
2.2.3 Παράλληλα Συστήματα

Η ισχυρή σύνδεση που υπάρχει μεταξύ των κόμβων ενός παράλληλου συστήματος (σε αντίθεση με τη ‘‘χαλαρή’’ σύνδεση που υπάρχει σε ένα κατανεμημένο σύστημα) οδηγεί στην παραλληλοποίηση των μονάδων εργασίας και επομένως οδηγεί σε αλληλοεξαρτήσεις όπως η επικοινωνία μεταξύ των εργασιών, ο συγχρονισμός και περιορισμοί σειράς εκτέλεσης (precedence constraints).

Η αρχιτεκτονική του παράλληλου υπολογιστή παίζει επίσης σημαντικό ρόλο στην επίδοση των παράλληλων εφαρμογών. Για παράδειγμα, σημαντική διαφορά στο που βρίσκονται τα δεδομένα και πως προσπελαύνονται από άλλους κόμβους, παίζει ο τύπος της αρχιτεκτονικής του παράλληλου υπολογιστή, και ειδικότερα αν είναι τύπου UMA (Uniform Memory Access – Ομοιόμορφης Προσπέλασης Μνήμης), ή τύπου NUMA (Non–UMA – Μη Ομοιόμορφης Προσπέλασης Μνήμης) ή τύπου NORMA (NO Remote Memory Access – Χωρίς Απομακρυσμένες Προσπελάσεις Μνήμης).

Αν και ο συχνότερος τρόπος για να περιγράψει κανείς ένα παράλληλο πρόγραμμα είναι με ένα γράφημα εργασιών (π.χ. σαν αυτό του σχήματος 2.8), η χρήση του γραφήματος εργασιών μπορεί να βοηθήσει στην αξιολόγηση μόνο των *στατικών παραμέτρων* της υπό μελέτη εφαρμογής. Συγκεκριμένα, μπορεί κανείς να υπολογίσει εύκολα τον *βαθμό–εισόδου (in–degree)*, δηλαδή το πλήθος των ακμών που καταλήγουν σε έναν κόμβο, και τον *βαθμό–εξόδου (out–degree)*, δηλαδή το πλήθος των ακμών που ξεκινάνε από έναν κόμβο, όπως αναφέρονται στο [CM94].

Απαραίτητες όμως για οποιαδήποτε μελέτη της επίδοσης μιας παράλληλης εφαρμογής είναι και οι *δυναμικές παράμετροι*, οι οποίες περιγράφουν την συμπεριφορά της εφαρμογής



Σχήμα 2.8: Γράφημα Εργασιών

κατά την εκτέλεσή της και πρέπει να συλλεχθούν όσο τρέχει η εφαρμογή (at run−time) από ειδικά εργαλεία παρακολούθησης επίδοσης του συστήματος. Αυτές οι παράμετροι μπορούν να αποκαλύψουν τη συμπεριφορά της εφαρμογής όσον αφορά τις ανάγκες της σε υπολογιστική ισχύ, επικοινωνίες και συγχρονισμό.

Ο μεγάλος όγκος δεδομένων από μετρήσεις δημιουργεί ένα σημαντικό πρόβλημα όσον αφορά στην ανάλυση και την ερμηνεία τους. Προσπάθειες για να λυθεί το πρόβλημα αυτό έχουν γίνει πολλές και έχουν προταθεί πολλά εργαλεία τόσο σε πειραματικό στάδιο όσο και σε εμπορικό επίπεδο. Αναφέρουμε ως παράδειγμα τα εξής εμπορικά προϊόντα:

- EXPRESS ([SAB+94]), και
- FORGE ([MHC94]).

καθώς και τα εξής που βρίσκονται σε πειραματικό στάδιο:

- Ο ARRAYTRACER ([NS95]) είναι ένα υψηλού επιπέδου, μικρού επιπλέον κόστους εργαλείο για ανάλυση επίδοσης παράλληλων εφαρμογών, που ειδικεύεται στην *ιχνοληψία πινάκων* από παράλληλες εφαρμογές και παρέχει στον χρήστη δυνατότητα αλλαγής του μεγέθους της ιχνοληψίας και είναι εύκολο στη χρήση του.
- 'Ενα εργαλείο που βοηθάει στην στατιστική ανάλυση και την οπτικοποίηση των δεδομένων από τις μετρήσεις είναι το *MEDEA* (MEasurement Description Evaluation and Analysis) που προτάθηκε πρόσφατα στο [CMM+94].

- Το *PAPS* είναι ένα σύνολο εργαλείων για πρόβλεψη της επίδοσης παράλληλων προγραμμάτων, βασίζεται σε μοντέλα επίδοσης με Petri Nets, και χρησιμοποιεί το μοντέλο *Προγράμματος–Πόρου–Αντιστοίχισης* (Program–Resource–Mapping model), που παρουσιάστηκε στο [WH94].

2.2.4 Βάσεις Δεδομένων

Τη στιγμή που οι βάσεις δεδομένων παίζουν ένα ολοένα μεγαλύτερο ρόλο στις περισσότερες υπολογιστικές εγκαταστάσεις (ανεξάρτητα του είδους των υπολογιστικών συστημάτων που χρησιμοποιούνται, που μπορεί να είναι κεντρικοποιημένα, κατανεμημένα ή παράλληλα) αξίζει κανείς να εξετάσει ξεχωριστά το πρόβλημα του χαρακτηρισμού του φόρτου εργασίας για τις βάσεις δεδομένων.

Στις πρώτες μελέτες σε IMS βάσεις δεδομένων ([GLJ76], [LS76]), οι μετρήσεις που μαζεύτηκαν ήταν κυρίως ακολουθίες από δοσοληψίες, τμήματα και blocks και μελετήθηκαν με στατιστική ανάλυση ακολουθιών από γεγονότα.

Σε μια σχετικά πρόσφατη μελέτη ([KD89]) οι συγγραφείς αποδεικνύουν (χρησιμοποιώντας ένα πολύ μεγάλο αρχείο μετρήσεων από συστήματα IMS δύο διαφορετικών εταιρειών) ότι για μια συγκεκριμένη υλοποίηση (IMS) και ένα συγκεκριμένο μοντέλο βάσης δεδομένων (ιεραρχικό), η συμπεριφορά των δοσοληψιών, όσον αφορά τις απαιτήσεις τους σε προσπελάσεις στη βάση δεδομένων, μπορεί να προβλεφθεί.

Σε πρόσφατες μελέτες σε σχεσιακές βάσεις δεδομένων, οι μετρήσεις που μαζεύτηκαν είχαν να κάνουν κυρίως με την είσοδο/έξοδο του διαχειριστή του ενταμιευτή, πληροφορία για locks καθώς και την σύνθεση και κατανομή των εντολών SQL. Για παράδειγμα:

- Στο [Kla92] αποδεικνύεται (με ανάλυση των μετρήσεων από μια βάση δεδομένων) ότι γενικά μοντέλα για ιχνοληψία προγραμμάτων δεν είναι εφαρμόσιμα σε περιπτώσεις λεπτομερών μελετών χαρακτηρισμού φόρτου εργασίας σε συστήματα βάσεων δεδομένων. Ο συγγραφέας προτείνει ένα γενικό, ιεραρχικό μοντέλο με τρία επίπεδα αφαίρεσης: το επίπεδο εφαρμογής, το επίπεδο δοσοληψίας και το επίπεδο φυσικών πόρων.

- Ένας αναλυτής φόρτου εργασίας για σχεσιακές βάσεις δεδομένων (REDWAR – RElational Database Workload AnalyzeR) παρουσιάζεται στο [YCHL92]. Το εργαλείο αυτό αποσκοπεί στο χαρακτηρισμό του φόρτου εργασίας ενός συστήματος DB2. Η μελέτη φόρτου εργασίας στο σύστημα αυτό βασίζεται στη δομή και την πολυπλοκότητα των εντολών SQL, τη σύσταση και τη δυναμική συμπεριφορά των δοσοληψιών / ερωτήσεων και τη σύνθεση των σχέσεων.

- Ο εντοπισμός της όποιας πρότυπης μορφής στις προσπελάσεις των δοσοληψιών σε

μια βάση δεδομένων, μπορεί να βοηθήσει στον υπολογισμό της πιθανότητας εύρεσης στοιχείων στον ενταμιευτή (buffer hit probability) το οποίο είναι πολύ σημαντικό στοιχείο στη διαχείριση του φόρτου εργασίας. Μια μελέτη σε αυτή την κατεύθυνση γίνεται στο [DYC94], όπου προτείνεται η ομαδοποίηση (με έναν δυαδικό αναδρομικό αλγόριθμο διαμέρισης) των σελίδων της βάσης δεδομένων σε διαμερίσεις, και το πλήθος των προσπελάσων των σελίδων είναι περίπου το ίδιο για σελίδες που ανήκουν στην ίδια διαμέριση.

Τέλος, η ιδέα της ομαδοποίησης δοσοληψιών με παρόμοια χαρακτηριστικά φόρτου εργασίας (και η σχετική έρευνα επί του θέματος) αναπτύσεται στο κεφάλαιο 2.4, μετά την σύντομη επισκόπη των αλγορίθμων ομαδοποίησης του κεφαλαίου 2.3.

2.3 Αλγόριθμοι Ομαδοποίησης

Η ομαδοποίηση (clustering) είναι ένα σύνολο από μεθόδους, τεχνικές και αλγορίθμους για να δημιουργηθούν κάποιες *ομάδες* (clusters) από ‘‘όμοια’’ στοιχεία. Παίζει πολύ σημαντικό ρόλο σε όλες σχεδόν τις επιστήμες, αφού η συστηματική οργάνωση δεδομένων σε ‘‘καλές’’ ομάδες είναι θεμελειώδης για την κατανόηση και τη μάθηση.

Πολλοί διαφορετικοί ορισμοί έχουν δοθεί όσον αφορά την έννοια της ομάδας ([Eve74]):

- Οι ομάδες μπορούν να περιγραφούν ως συνδεδεμένες περιοχές ενός πολυδιάστατου χώρου, οι οποίες περιέχουν μια σχετικά μεγάλη *πυκνότητα* σημείων και διαχωρίζονται από τις υπόλοιπες ομάδες με άλλες περιοχές με μια ιδιαίτερα μικρή πυκνότητα σημείων.
- Μία ομάδα είναι ένα σύνολο από στοιχεία που μοιάζουν μεταξύ τους, ενώ στοιχεία από διαφορετικές ομάδες δεν μοιάζουν μεταξύ τους.

Κάθε προσπάθεια ομαδοποίησης ή ταξινόμησης, πρέπει λίγο ή πολύ να απαντήσει στα ακόλουθα ερωτήματα ([Boc94]) :

- Τι θεωρούμε ως κλάση ή ομάδα; (Ένα σύνολο από αντικείμενα, μια έννοια, ή μια ασαφής (fuzzy) κατασκευή;)
- Ποιές είναι οι χαρακτηριστικές ιδιότητες που καθορίζουν μια ομάδα; (Η εμφάνιση μιας ιδιότητας ή ενός συνδυασμού ιδιοτήτων; Η αμοιβαία ομοιότητα μεταξύ των μελών;)
- Τι είναι το σύστημα ταξινόμησης ή ποια είναι η δομή της ομαδοποίησης; (Μία διαμέριση, μια ιεραρχία, ένα δίκτυο από κλάσεις, ή ένα εννοιολογικό πλαίσιο;)
- Τι δομικές σχέσεις υπάρχουν μεταξύ των ομάδων; (Σχέση υποσυνόλου, επικάλυψης, ή σημασιολογική σχέση;)
- Ποια είναι μια ‘‘καλή’’ (‘‘ισχυρή’’, ‘‘ασθενής’’) δομή ομαδοποίησης;
- Τι στόχος θα έπρεπε να ικανοποιηθεί από την ταξινόμηση;

2.3.1 Κατηγορίες Αλγορίθμων

Μπορεί κανείς να διακρίνει αρκετές κατηγορίες μεθόδων ταξινόμησης οι περισσότερο δια−δεδομένες όμως είναι οι ακόλουθες([JD88]).

Η πρώτη διάκριση που χρησιμοποιείται είναι κατά πόσο ένα συγκεκριμένο στοιχείο μπορεί να ανήκει σε παραπάνω από μια διαφορετικές κλάσεις. Στην περίπτωση που κάτι τέτοιο είναι επιτρεπτό, τότε έχουμε *ταξινόμηση με επικάλυψη*, ενώ στην αντίθετη περίπτωση έχουμε *ταξινόμηση χωρίς επικάλυψη*.

Στις μεθόδους ταξινόμησης χωρίς επικάλυψη υπάρχει μια δεύτερη διάκριση που έχει να κάνει με το αν οι αλγόριθμοι βασίζονται εξ ολοκλήρου στον πίνακα εγγύτητας (proximity matrix), οπότε δουλεύουν *χωρίς επίβλεψη*, (unsupervised) ή χρειάζονται επιπλέον στοιχεία για να βγάλουν συμπεράσματα, οπότε χρειάζονται κάποια *επίβλεψη* (supervised).

Τέλος, στις μεθόδους που δεν χρειάζονται επίβλεψη, διακρίνονται άλλες δύο κατηγορίες με βάση την δομή που επιβάλουν στα στοιχεία που ταξινομούν: οι *Ιεραρχικές* (Hierarchical) και οι Μη−Ιεραρχικές (Non−Hierarchical) ή *Διαμεριστικές* (Partitional). Η βασική διαφορά είναι ότι οι ιεραρχικές μέθοδοι ταξινομούν τα στοιχεία σε διαμερίσεις, οι οποίες είναι φωλιασμένες η μια στην άλλη, ενώ στις διαμεριστικές μεθόδους η διαμέριση είναι μόνο μια.

Όσον αφορά τους αλγόριθμους που υλοποιούν τις συγκεκριμένες μεθόδους, υπάρχουν επίσης σημαντικές διαφορές ([JD88]):

- Οι *συγχωνευτικοί* (agglomerative) αλγόριθμοι τοποθετούν αρχικά το κάθε σημείο σε μια ξεχωριστή ομάδα και σταδιακά συγχωνεύουν αυτές τις ομάδες σε ολοένα και μεγαλύτερες ομάδες για να καταλήξουν σε μια ομάδα που περιέχει όλα τα αρχικά σημεία.

  Οι *διαιρετικοί* (divisive) αλγόριθμοι, ακολουθούν την ακριβώς αντίθετη προσέγγιση και τοποθετούν αρχικά όλα τα σημεία σε μια κλάση και προσπαθούν διαδοχικά να διαμερίσουν την κλάση αυτή σε ολοένα μικρότερες κλάσεις για να καταλήξουν σε κλάσεις με ένα μόνο σημείο.

- Οι *σειριακοί* (serial) αλγόριθμοι χειρίζονται διατάξεις σημείων μια τη φορά, ενώ οι *ταυτόχρονοι* (simultaneous) αλγόριθμοι μπορούν να χειριστούν πολλές διατάξεις σημείων μαζί.

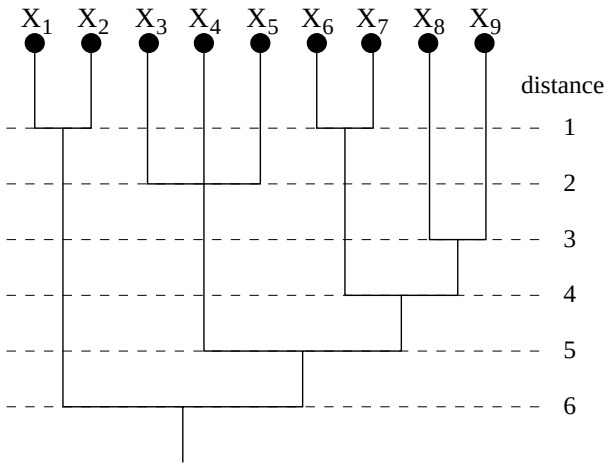
- Οι *μονοθετικοί* (monothetic) αλγόριθμοι αντιμετωπίζουν τα διάφορα ξεχωριστά χαρακτηριστικά (features) που μπορεί να έχει το κάθε στοιχείο ένα τη φορά, ενώ οι *πολυθετικοί* (polythetic) αλγόριθμοι αντιμετωπίζουν τα διάφορα ξεχωριστά χαρακτηριστικά όλα μαζί.

- Ανάλογα με το μαθηματικό φορμαλισμό που χρησιμοποιούνε οι αλγόριθμοι διακρίνονται δύο κατηγορίες: όσοι χρησιμοποιούν *θεωρία γραφημάτων* (graph theory) και όσοι χρησιμοποιούν *άλγεβρα πινάκων* (matrix algebra).

Ιεραρχικοί Αλγόριθμοι

Οι ιεραρχικοί αλγόριθμοι (ή για την ακρίβεια ιεραρχικοί αλγόριθμοι χωρίς επικάλυψη και χωρίς επίβλεψη) ομαδοποιούν τα αρχικά στοιχεία κατασκευάζοντας μια ακολουθία από φωλιασμένες διαμερίσεις. Αυτή η ακολουθία, συνήθως οπτικοποιείται με ένα δεντρόγραμμα (dendrogram) όπου μπορεί κανείς εύκολα να δει τον τρόπο με τον οποίο ομαδοποιούνται τα

στοιχεία (βλέπε σχήμα 2.9). Αξίζει να σημειωθεί ότι ‘‘κόβοντας’’ ένα δενδρόγραμμα οριζόντια προκύπτει μια ομαδοποίηση των στοιχείων.



Σχήμα 2.9: Παράδειγμα Δενδρογράμματος

Όσον αφορά τον τρόπο με τον οποίο υπολογίζεται η απόσταση μεταξύ των διάφορων ομάδων, διακρίνουμε δύο βασικές κατηγορίες:

- *Μέθοδοι θεωρίας γραφημάτων*: Η απόσταση μεταξύ δύο ομάδων υπολογίζεται χρησιμοποιώντας το γράφημα των σημείων στις δύο ομάδες. Υπάρχουν τρεις εναλλακτικές λύσεις για αυτό:
  - *Απλή σύνδεση* (single link): Η απόσταση μεταξύ δύο ομάδων είναι η ελάχιστη απόσταση μεταξύ δύο σημείων τέτοιων ώστε το ένα σημείο είναι από την πρώτη ομάδα και το άλλο σημείο από τη δεύτερη ομάδα.
  - *Μέση σύνδεση* (average link): Η απόσταση μεταξύ δύο ομάδων είναι η μέση απόσταση μεταξύ όλων των ζευγαριών σημείων, όπου το ένα σημείο είναι από την πρώτη ομάδα και το άλλο σημείο από τη δεύτερη ομάδα.
  - *Πλήρη σύνδεση* (complete link): Η απόσταση μεταξύ δύο ομάδων είναι η μέγιστη απόσταση μεταξύ δύο σημείων τέτοιων ώστε το ένα σημείο είναι από την πρώτη ομάδα και το άλλο σημείο από τη δεύτερη ομάδα.
- *Γεωμετρικές μέθοδοι*: Ορίζονται τα κέντρα των ομάδων (cluster centers) και κατόπιν χρησιμοποιούνται αυτά για να βρίσκονται οι αποστάσεις μεταξύ των ομάδων.

  Υπάρχουν τρεις εναλλακτικές λύσεις για αυτό:

- *Κέντρο Βάρους* (centroid): Το κέντρο της ομάδας είναι το κέντρο βάρους των σημείων που την αποτελούν. Οι αποστάσεις μεταξύ των κέντρων των ομάδων υπολογίζονται με την Ευκλείδεια απόσταση.
- *Μέσος* (median): Το κέντρο της ομάδας είναι ο μέσος όρος από τα κέντρα των ομάδων που συγχωνεύτηκαν για να τη σχηματίσουν. Οι αποστάσεις μεταξύ των κέντρων των ομάδων υπολογίζονται με την Ευκλείδεια απόσταση.
- *Ελάχιστη τυπική απόκλιση* (minimum variance): Το κέντρο της ομάδας είναι το κέντρο βάρους των σημείων που την αποτελούν. Η διαφορά από την πρώτη περίπτωση βρίσκεται στο ότι η απόσταση μεταξύ των κέντρων των ομάδων υπολογίζεται ως η αύξηση στο άθροισμα των τετραγώνων των αποστάσεων κάθε σημείου της ομάδας από το κέντρο της ομάδας, που προκαλείται από την συγχώνευση των δύο ομάδων.

Διαμεριστικοί Αλγόριθμοι

Οι διαμεριστικοί αλγόριθμοι (ή για την ακρίβεια διαμεριστικοί αλγόριθμοι χωρίς επικάλυψη και χωρίς επίβλεψη) ομαδοποιούν τα αρχικά στοιχεία και κατασκευάζουν μια μοναδική διαμέριση προσπαθώντας να ανακαλύψουν ‘‘φυσικές’’ ομαδοποιήσεις.

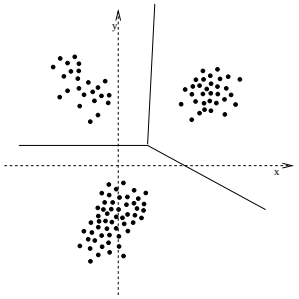
Μια πρώτη σύγκριση των διαμεριστικών αλγορίθμων με τους ιεραρχικούς, αποκαλύπτει ότι οι ιεραρχικοί αλγόριθμοι είναι ιδιαίτερα βολικοί στη βιολογία, κοινωνιολογία και ψυχολογία, αλλά είναι πρακτικά ασύμφοροι στις θετικές επιστήμες όπου ο όγκος των δεδομένων είναι πολύ μεγάλος, και χρησιμοποιούνται κυρίως διαμεριστικοί αλγόριθμοι.

Συνήθως στα δεδομένα του αλγορίθμου συμπεριλαμβάνεται και ο αριθμός K των ομάδων που θα πρέπει να κατασκευάσει ο αλγόριθμος. Με βάση το K θα πρέπει να υπολογιστεί μια διαμέριση του αρχικού συνόλου των σημείων, τέτοια ώστε τα στοιχεία μιας ομάδας να είναι περισσότερο όμοια μεταξύ τους σε σχέση με στοιχεία άλλων ομάδων.

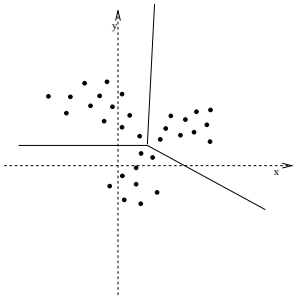
Υπάρχουν δύο είδη κριτηρίων που χρησιμοποιούνται για να βοηθήσουν στη δημιουργία των ομάδων με όμοια στοιχεία:

- *Καθολικά κριτήρια* (global criteria): Κάθε ομάδα αναπαριστάται από ένα πρότυπο στοιχείο (prototype) το οποίο χρησιμοποιείται ως αντιπρόσωπος της ομάδας για να βρεθεί αν κάποιο άλλο στοιχείο είναι όμοιο με τα στοιχεία της ομάδας.
- *Τοπικά κριτήρια* (local criteria): Κατά την ομαδοποίηση εκμεταλεύονται τις όποιες τοπικές δομές.

Η θεωρητική λύση στο πρόβλημα της διαμέρισης είναι απλή: αφού επιλεγεί ένα κριτήριο, τότε για όλες τις πιθανές διαμερίσεις του αρχικού συνόλου σε K ομάδες, διαλέγει κανείς αυτή που βελτιστοποιεί το κριτήριο. Εύκολα διαπιστώνει κανείς ότι η παραπάνω λύση δεν



Σχήμα 2.10: Εύκολη περίπτωση διαμέρισης



Σχήμα 2.11: Δύσκολη περίπτωση διαμέρισης

είναι εφικτή, αφού και η απλή απαρίθμηση όλων των πιθανών διαμερίσεων ενός συνόλου μόνο προκαλεί συνδυαστική έκρηξη (combinatorial explosion) ακόμα και για σχετικά μικρά μεγέθη.

Η λύση που ακολουθείται συνήθως, βασίζεται σε μια (τυχαία) αρχική διαμέριση του συνόλου των σημείων σε K ξένες μεταξύ τους ομάδες, και με βάση κάποια κριτήρια γίνονται μετατάξεις σημείων από τη μια ομάδα στην άλλη. Η λύση αυτή, παρόλο που είναι αρκετά γρήγορη, έχει το μειονέκτημα ότι βασίζεται αρκετά στην αρχική διαμέριση. Επιπλέον ένα γενικότερο πρόβλημα είναι οι ‘‘κακές’’ περιπτώσεις, όπου οι ομάδες είναι αρκετά ‘‘κοντά’’ μεταξύ τους, οπότε δεν μπορούν να ξεχωρίσουν εύκολα (βλέπε σχήμα 2.11).

Ένας άλλο τρόπος για να γλυτώσει κανείς τη συνδυαστική έκρηξη είναι να μπορέσει να βρει έναν τρόπο και να διαγράψει από το σύνολο των δυνατών διαμερίσεων ένα σημαντικό ποσοστό που δεν είναι ενδιαφέρουσες.

2.3.2 Γνωστοί Αλγόριθμοι

Στις επόμενες παραγράφους παρουσιάζονται εν συντομία κάποιοι χαρακτηριστικοί αλγόριθμοι − τεχνικές ομαδοποίησης.

Ο αλγόριθμος ελάχιστου ζευγνύοντος δένδρου

Ο αλγόριθμος *ελάχιστου ζευγνύοντος δένδρου* (Minimum Spanning Tree, ή MST) είναι ένας ιεραρχικός αλγόριθμος ο οποίος βασίζεται στη θεωρία γραφημάτων ([Zah71]). Συγκεκριμένα, ο αλγόριθμος βασίζεται στη δημιουργία ενός ελάχιστου ζευγνύοντος δένδρου, το οποίο να συνδέει όλα τα σημεία ενός δοσμένου συνόλου.

Το βάρος σε κάθε ακμή του δένδρου είναι σε αυτή την περίπτωση ίσο με την ευκλείδεια απόσταση των δύο σημείων που ενώνονται από την εν λόγω ακμή. Αν οριστεί ως βάρος ολόκληρου του δένδρου το άθροισμα των βαρών όλων των ακμών του, τότε για ένα δοσμένο γράφημα, το ελάχιστο ζευγνύον δένδρο, αποτελεί και το δένδρο με το μικρότερο συνολικό

βάρος. Κατόπιν μπορεί κανείς με διάφορες μεθόδους να ενώσει επιμέρους ακμές και να ελαττώσει το πλήθος των κόμβων του γραφήματος.

Κριτήριο Τετραγωνικού Σφάλματος

Ένα αρκετά διαδεδομένο κριτήριο που χρησιμοποιείται σε διάφορους διαμεριστικούς αλγόριθμους είναι το κριτήριο του Τετραγωνικού Σφάλματος (Square error criterion).

Αρχικά, το σύνολο των n στοιχείων διαμερίζεται σε K ομάδες $\{C_1, C_2, \ldots, C_K\}$ τέτοιες ώστε η ομάδα C_k να έχει n_k στοιχεία και κάθε στοιχείο να βρίσκεται σε μια μόνο ομάδα, δηλαδή να ισχύει $\sum_{k=1}^{K} n_k = n$.

Το μέσο διάνυσμα για την ομάδα C_k ορίζεται ως:

$$m^{(k)} = (1/n_k) \sum_{i=1}^{n_k} x_i^{(k)}$$

Το τετραγωνικό σφάλμα για την ομάδα C_k είναι:

$$e_k^2 = \sum_{i=1}^{n_k} (x_i^{(k)} - m^{(k)})^T (x_i^{(k)} - m^{(k)})$$

Το τετραγωνικό σφάλμα για ολόκληρη την ομαδοποίηση είναι:

$$E_K^2 = \sum_{k=1}^{K} e_k^2$$

Ο σκοπός της ομαδοποίησης με βάση το κριτήριο του Τετραγωνικού Σφάλματος είναι να βρεθεί εκείνη η διαμέριση σε K ομάδες που να ελαχιστοποιεί το E_K^2 για δοσμένο K.

ISODATA

Ο ISODATA είναι από τους περισσότερο γνωστούς διαμεριστικούς αλγορίθμους.

Η φιλοσοφία του είναι αρκετά απλή:

- Αρχικά ο αλγόριθμος υποθέτει μια αρχική διαμέριση σε K ομάδες.
- Κατόπιν υπολογίζονται τα κέντρα των ομάδων καθώς και οι απόστάσεις των σημείων που ανήκουν σε μια ομάδα από το κέντρο της ομάδας.
- Τα σημεία τοποθετούνται στις πλησιέστερες ομάδες και επαναλαβάνεται η όλη διαδικασία.
- Τέλος, υπάρχει η δυνατότητα ενοποίησης / διαμέρισης των κλάσεων καθώς και ανθρώπινης παρέμβασης.

Υπάρχουν αρκετές διαφορετικές υλοποιήσεις του αλγορίθμου, στην εργασία αυτή όμως επιλέχτηκε η υλοποίηση από το [VR92], η οποία βασίζεται στον $K - Means$ αλγόριθμο ο οποίος διατηρεί σταθερό το πλήθος των κλάσεων και δεν χρειάζεται ανθρώπινη παρέμβαση.

Bond Energy Algorithm

Ο αλγόριθμος Bond Energy (ή Αλγόριθμος Ενέργειας Δεσμού) προτάθηκε στο [MSW72] και έκτοτε χρησιμοποιήθηκε ευρύτατα κυρίως για κάθετη τμηματοποίηση (vertical fragmentation) των σχέσεων σε κατανεμημένες βάσεις δεδομένων, όπου ομαδοποιούνται εκείνες οι ιδιότητες που παρουσιάζουν την μικρότερη αλληλεπίδραση μεταξύ τους.

Στο [NCWD84] εμφανίζεται αναλυτικά ο αλγόριθμος Ενέργειας Δεσμού για κάθετη τμηματοποίηση καθώς και οι προεκτάσεις που προτείνουν οι συγγραφείς, ενώ στο [AH90] γίνεται μια κριτική για τις χρήσεις του αλγορίθμου συνολικά. Ο αλγόριθμος Ενέργειας Δεσμού εμφανίζεται και στο [OV91], όπου επίσης χρησιμοποιείται για κάθετη τμηματοποίηση, καθώς και στο [CMVN92] όπου συγκρίνεται μαζί με άλλους αλγορίθμους κάθετης τμηματοποίησης.

Η βασική ιδέα πίσω από τον Αλγόριθμο Ενέργειας Δεσμού είναι η έννοια του *δεσμού* μεταξύ των n ιδιοτήτων της βάσης δεδομένων, ο οποίος ποσοτικοποιεί τη συγγένεια μεταξύ δύο οποιωνδήποτε ιδιοτήτων. Η συγγένεια αυτή ανάγεται στην ομοιότητα στη σύνθεση των προσπελάσεων που γίνονται στις ιδιότητες της βάσης δεδομένων από τις διάφορες δοσοληψίες. Με άλλα λόγια θα πρέπει ο δεσμός μεταξύ δύο ιδιοτήτων να έχει μεγαλύτερη τιμή, όσο περισσότερες ίδιες δοσοληψίες προσπελαύνουν τις δύο ιδιότητες.

Συγκεκριμένα ο δεσμός μεταξύ δύο ιδιοτήτων A_x και A_y είναι:

$$bond(A_x, A_y) = \sum_{z=1}^{n} aff(A_z, A_x) \times aff(A_z, A_y)$$

όπου ορίζουμε ως μέτρο συγγένειας (affinity):

$$aff(A_i, A_j) = \sum_{k|u_{ki}=1 \wedge u_{kj}=1} acc_k$$

και επιπλέον:

u_{ki} = 1, αν η δοσοληψία k χρησιμοποιεί την a_i
= 0, αλλιώς (όπου $u_{ki} \in AUM$)

$type_k$ = 'r' / 'u'

n_k = πλήθος προσπελάσεων για την δοσοληψία k

$freq_k$ = συχνότητα εμφάνισης της δοσοληψίας k

acc_k = $n_k \times freq_k$

2.4 Ομαδοποίηση Φόρτου Εργασίας

Η ομαδοποίηση δοσοληψιών/διεργασιών/προγραμμάτων με βάση το φόρτο που παρουσιάζουν στο σύστημα έχει προταθεί επανειλημμένα στο παρελθόν ([FSZ83], [Raa93]) και είναι γνωστή με το όνομα *Workload Clustering* ή *Ομαδοποίηση Φόρτου Εργασίας*. Αποσκοπεί στην εύρεση χαρακτηριστικών ομάδων με παρόμοιο φόρτο εργασίας και την χρησιμοποίησή τους για την όποια μελέτη του συστήματος.

Τα σημαντικότερα προβλήματα που έχει η γενική αντιμετώπιση του θέματος αυτού είναι ([FSZ83]):

- Η επιλογή του αλγόριθμου ομαδοποίησης.
- Η συνάρτηση μέτρησης απόστασης (συνήθως ευκλείδεια).
- Η αλλαγή στην κλίμακα μερικών από τις μεταβλητές.
- Η κανονικοποίηση. Συνηθισμένη περίπτωση είναι αυτή της ανηγμένης μεταβλητής (βλέπε και σελίδα 39), ή η περίπτωση της ανάθεσης βαρών στις διάφορες μεταβλητές.
- Η αντιμετώπιση των ακριτών. Μια μέθοδος είναι η λογαριθμική κλιμάκωση, άλλη είναι η ολοκληρωτική εξάλειψή τους από τα δεδομένα εισόδου.
- Η αποδοτική αντιμετώπιση των μεγάλων μεγεθών του προβλήματος. Ένας τρόπος είναι η δειγματοληψία, όπου ([Art78]) ακόμα και σε περιπτώσεις όπου ένα ελάχιστο ποσοστό (2%) του αρχικού δείγματος χρησιμοποιήθηκε τα αποτελέσματα δεν παρουσίαζαν σημαντική απόκλιση.

  Ένας άλλος τρόπος είναι με χρήση μεθόδων *ανάλυσης πρωταρχικών τμημάτων* (principal components analysis), όπου γίνεται προσπάθεια να βρεθούν εκείνες οι συνιστώσες του αρχικού δείγματος που μπορούν να αναπαραστήσουν και τις υπόλοιπες.

Στο [Raa93], όπου υπάρχει μια παρόμοια αναφορά των προβλημάτων που αντιμετωπίζει κανείς κατά την ομαδοποίηση φόρτου εργασίας, στηλιτεύεται η ανυπαρξία αξιολόγησης των αποτελεσμάτων της ομαδοποιήσης όσον αφορά τη σταθερότητα και την εγκυρότητά τους, η οποία διαφαίνεται σε όλες σχεδόν τις μελέτες ομαδοποίησης φόρτου εργασίας. Επιπλέον γίνεται η παρατήρηση ότι ο αλγόριθμος $K-Means$ είναι ο πιο διαδεδομένος στις περισσότερες περιπτώσεις ομαδοποίησης φόρτου εργασίας.

Η χρήση μεθόδων ομαδοποίησης για τη συρρίκνωση μεγάλων γραφημάτων συμπεριφοράς χρήστη προτείνεται στο [CF86] και αποδεικνύεται ως αρκετά ικανοποιητική λύση για την περίπτωση των διαλογικών συστημάτων.

Πρόσφατα, έχει προταθεί η ομαδοποίηση δοσοληψιών με βάση τις ανάγκες τους σε πόρους του συστήματος επεξεργασίας δοσοληψιών. Η πρώτη αναφορά στο θέμα έγινε στο [YD92], όπου παρουσιάζεται η ιδέα της ομαδοποίησης δοσοληψιών με βάση τη συγγένεια δεδομένων *(affinity clustering)*, και αργότερα στο [YD94].

Στις δύο αυτές περιπτώσεις χρησιμοποιείται η συγγένεια δεδομένων ως βάση για το διαχωρισμό δοσοληψιών σε ομάδες που κάνουν παρόμοιες προσπελάσεις στη βάση δεδομένων. Ο λόγος για τη χρήση της πολιτικής αυτής είναι για να μπορέσει να υπάρξει εξισορόπηση του φόρτου εργασίας μεταξύ των διάφορων υπολογιστικών κόμβων του συστήματος. Επιπλέον, στις δύο προαναφερθείσες μελέτες διερευνήθηκαν οι περιπτώσεις κατά τις οποίες ένας συγκεκριμένος φόρτος εργασίας (workload) είναι διαμερίσιμος σε ίσα μέρη ώστε να επιτευχθεί η επιθυμητή εξισορόπηση.

Στο [YD94], τέλος, γίνεται σύγκριση μεταξύ των τριών διαφορετικών αρχιτεκτονικών σύζευξης υπολογιστικών κόμβων, όσον αφορά τις διαφορές στην επίδοση με τη χρήση της ομαδοποίησης δοσοληψιών με βάση τη συγγένεια δεδομένων.

Κεφάλαιο 3

Ομαδοποίηση δοσοληψιών

3.1 Εισαγωγή

Η γνώση των χαρακτηριστικών φόρτου εργασίας των δοσοληψιών είναι σημαντική για την επιτυχία αλγόριθμων δρομολόγησης δοσοληψιών που προσπαθούν να αυξήσουν τις επιδόσεις ενός κατανεμημένου συστήματος επεξεργασίας δοσοληψιών. Τα χαρακτηριστικά αυτά δεν εξαρτώνται από τους χρόνους προσέλευσης, αλλά περιλαμβάνουν τις προσπελάσεις που κάνουν οι δοσοληψίες στα αρχεία της βάσης, τις υπολογιστικές ανάγκες της δοσοληψίας και το πλήθος των σημείων συγχρονισμού.

Η συνηθισμένη πρακτική μέχρι τώρα είναι να χρησιμοποιούνται προσχεδιασμένα μικρά προγράμματα δοσοληψιών ([Gra93]), τα οποία θα αναπαριστούν (σε όποιο βαθμό αυτό είναι δυνατό) δοσοληψίες που αντιστοιχούν σε συγκεκριμένες εμπορικές εφαρμογές (π.χ. η δοσοληψία χρέωσης/πίστωσης). Έτσι μπορεί να γίνει μια προσεγγιστική μελέτη του φόρτου εργασίας του συστήματος, η οποία όμως με την πολυπλοκότητα και την μεγάλη ποικιλομορφία των σημερινών εφαρμογών καταλήγει να είναι αναποτελεσματική.

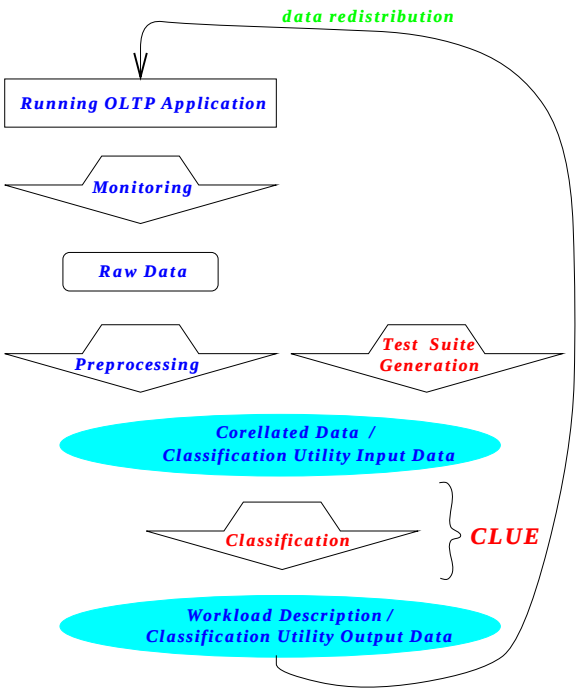
Η βασική ιδέα πίσω από το *CLUE* είναι ότι σε μεγάλες εφαρμογές, η γεωγραφική και οργανωτική δομή μιας εταιρίας έχει μεγάλη επίδραση στον τρόπο με τον οποίο αποθηκεύονται και προσπελαύνονται τα δεδομένα. Κάποιοι χρήστες θα χρησιμοποιούν ένα περιορισμένο σύνολο από τερματικά για να προσπελάσουν κάποια δεδομένα, κάτι το οποίο σχετίζεται με την καθημερινή τους εργασία και τον τόπο δουλειάς (π.χ. ο τραπεζικός λογαριασμός που σχετίζεται με το υποκατάστημα της τράπεζας που δουλεύει ο υπάλληλος).

Γι'αυτόν το λόγο, χρησιμοποιήθηκε ο συνδυασμός του *κωδικού δοσοληψίας* (transactionID), του *κωδικού χρήστη* (userID), και του *κωδικού τερματικού* (termID) ως ένα μοναδικό προσδιοριστικό δοσοληψιών που πρόκειται να παρουσιάσουν παρόμοια συμπεριφορά όσον αφορά τις προσπελάσεις στη βάση δεδομένων στο μέλλον. Αυτός ο συνδυασμός ονομάζεται *κωδικός τριπλέτας* και μαζί με τα επιπλέον στατιστικα στοιχεία ονομάζεται *τριπλέτα*. Η επιλογή των

τριών αυτών χαρακτηριστικών έγινε επειδή αφενός αποτελούν το ελάχιστο υποσύνολο χαρακτηριστικών που μπορούν να χρησιμοποιηθούν για το ζητούμενο σκοπό και αφετέρου είναι κοινά σε όλα τα συστήματα επεξεργασίας δοσοληψιών.

Ο συνηθισμένος αριθμός από δοσοληψίες είναι συνήθως αρκετές χιλιάδες, ενώ ο τυπικός αριθμός από χρήστες είναι συνήθως δεκάδες χιλιάδες και ο τυπικός αριθμός από τερματικά είναι επίσης δεκάδες χιλιάδες. Παρόλο που ο τυπικός αριθμός των τριπλετών δεν είναι ίσος με το γινόμενο των τριων μεγεθών που προαναφέρθηκαν, συνήθως ο αριθμός αυτός είναι επίσης στην τάξη των δεκάδων χιλιάδων. Είναι αδύνατο όμως στους αλγόριθμους δρομολόγησης δοσοληψιών να ξεχωρίσουν μεταξύ τόσων πολλών δοσοληψιών, και επιπλέον ο χρόνος που χρειάζεται για να πάρει ο αλγόριθμος δρομολόγησης μια απόφαση αυξάνει άμεσα το χρόνο απόκρισης της δοσοληψίας. Έτσι θα πρέπει να γίνει κάποια μορφή συμπίεσης του συνολικού αριθμού των τριπλετών, με τέτοιο τρόπο όμως, ούτως ώστε να μην χαθούν πληροφορίες σημαντικές για την επιτυχία των αλγορίθμων δρομολόγησης.

Το *CLUE* είναι ένα περιβάλλον ομαδοποίησης δοσοληψιών με βάση τα χαρακτηριστικά φόρτου εργασίας που παρουσιάζουν. Συγκεκριμένα, χρησιμοποιεί την ομοιότητα στις προσπελάσεις που κάνουν οι δοσοληψίες στη βάση δεδομένων για να κατασκευάσει ομάδες δοσοληψιών με μεγάλη *συγγένεια δεδομένων* μεταξύ τους.



Σχήμα 3.1: Το *CLUE* εν δράσει

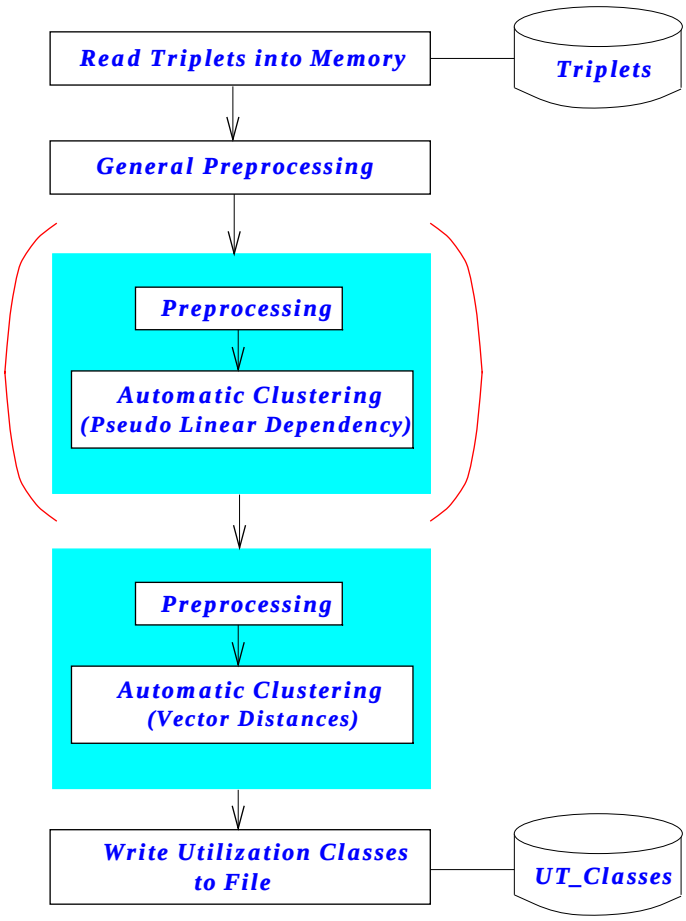
3.2 Δομή του CLUE

Στην τρέχουσα υλοποίησή του, το *CLUE* δουλεύει ανεξάρτητα από το σύστημα online επεξεργασίας δοσοληψιών. Γίνεται πρώτα η συλλογή των ιχνών (traces) από την εκτέλεση των δοσοληψιών στο σύστημα και αυτά αποθηκεύονται σε ένα αρχείο. Κατόπιν, μετά από κατάλληλη προεπεξεργασία, το *CLUE* αντλεί στοιχεία από το αρχείο με τα ίχνη της εκτέλεσης των δοσοληψιών (σχήμα 3.1) και τα χρησιμοποιεί για να κατασκευάσει ομάδες από δοσοληψίες με αυξημένη συγγένεια δεδομένων (δηλαδή που κάνουν παρόμοιες προσπελάσεις στη βάση δεδομένων). Εναλλακτικά, το *CLUE* μπορεί να αντλήσει στοιχεία από κάποιο συνθετικά κατασκευασμένο παράδειγμα με σκοπό να επαληθευτεί η σωστή λειτουργία του.

Σε κάθε περίπτωση το *CLUE* χρειάζεται από τον χρήστη να προσδιορίσει ένα πάνω φράγμα για το μέγιστο πλήθος των κλάσεων που θα πρέπει να κατασκευάσει, το οποίο στις συνηθισμένες περιπτώσεις κυμαίνεται γύρω στο 100. Το όριο των περίπου 100 κλάσεων προκύπτει από την υπάρχουσα εμπειρία με τα συστήματα κατανεμημένης επεξεργασίας δοσοληψιών, αλλά και από τον περιορισμό, που τίθεται από τους αλγόριθμους δρομολόγησης, για μικρό πλήθος κλάσεων.

DB / T	B_1	B_2	B_3		B_j		B_k
T_1							
T_2							
T_3							
T_i					#reads #writes		
T_n							

Σχήμα 3.2: Πίνακας Προσπελάσεων



Σχήμα 3.3: Λειτουργία του *CLUE*

Αναλυτικά, τα στάδια λειτουργίας του *CLUE* είναι τα εξής (βλέπε και σχήμα 3.3):

1. Αρχικά, διαβάζονται τα στοιχεία από το αρχείο με τα ίχνη της εκτέλεσης των δοσοληψιών και αποθηκεύονται σε μία δυναμική δομή δεδομένων στην κύρια μνήμη του υπολογιστή. Η δυναμική δομή αναπαριστά έναν *πίνακα προσπελάσεων* (reference matrix), όπου οι γραμμές του είναι τριπλέτες και οι στήλες του είναι σελίδες της βάσης δεδομένων (σχήμα 3.2). Κάθε σημείο του πίνακα περιέχει τον αριθμό των προσπελάσεων για ανάγνωση και των προσπελάσεων για ενημέρωση που έχει κάνει συνολικά η συγκεκριμένη τριπλέτα [1] στη συγκεκριμένη σελίδα της βάσης δεδομένων.

2. Κατόπιν γίνεται προεπεξεργασία των ιχνών.

3. Ακολουθεί η εκτέλεση του αλγόριθμου και η ομαδοποίηση των τριπλετών.

[1] για την ακρίβεια η δοσοληψία που ζητήθηκε από τον συγκεκριμένο χρήστη στο συγκεκριμένο τερματικό

4. Τέλος, αποθηκεύονται τα αποτελέσματα σε αρχείο στο δίσκο.

Αξίζει να σημειωθεί ότι το πρώτο και το τελευταίο στάδιο είναι υποχρεωτικά και γίνονται μόνο μία φορά, ενώ τα στάδια 2 και 3 μπορούν να γίνουν όσες φορές χρειαστεί (με ανάλογη καταχώριση στο αρχείο περιγραφής του πειράματος) και με διαφορετικές παραμέτρους. Στις επόμενες παραγράφους παρουσιάζονται τα στάδια της προεπεξεργασίας και οι τρόποι που χρησιμοποιήθηκαν για τη μέτρηση αποστάσεων μεταξύ των τριπλετών, ενώ ο ευρηματικός αλγόριθμος παρουσιάζεται αναλυτικά στο επόμενο υποκεφάλαιο.

3.2.1 Προεπεξεργασία

Στο στάδιο της προεπεξεργασίας γίνονται διάφορες στατιστικές κυρίως μετατροπές στα αρχικά δεδομένα για να γίνουν ‘‘καλύτερα’’ για τους διάφορους αλγόριθμους που θα τα χρησιμοποιήσουν στη συνέχεια. Το στάδιο αυτό είναι πολύ σημαντικό στην καλή λειτουργία του *CLUE* γιατί αρκετοί από τους αλγόριθμους που χρησιμοποιήθηκαν ήταν ‘‘ευαίσθητοι’’ στην αρχική κατάσταση (η οποία επηρεάζεται πάρα πολύ από το στάδιο αυτό).

Κλιμάκωση

Η πρώτη πράξη που μπορεί κανείς να εφαρμόσει στα στοιχεία του πίνακα είναι μια κλιμάκωση όλων των στοιχείων του. Με αυτό τον τρόπο, αφενός μπορεί κανείς να θέσει όλα τα στοιχεία του πίνακα σε μια γνωστή εκ των προτέρων κλίμακα και αφετέρου να αποφύγει περιπτώσεις υπερχείλισης (overflow) στους περαιτέρω υπολογισμούς.

Η κλιμάκωση προς το παρόν γίνεται με απλή διαίρεση όλων των στοιχείων του πίνακα με το μεγαλύτερο στοιχείο, οπότε η τελική κλίμακα των στοιχείων είναι το κλειστό διάστημα $[0, 1]$.

Άλλη μέθοδος κλιμάκωσης των στοιχείων του πίνακα προσπελάσεων θα μπορούσε να είναι η *λογαριθμική*, όπου με την πράξη αυτή κάθε στοιχείο του πίνακα θα αντικατασταθεί από τον λογάριθμο του σε μια συγκεκριμένη βάση. Η μέθοδος αυτή είναι ιδανική για τις περιπτώσεις όπου οι τιμές εισόδου διαφέρουν πάρα πολύ μεταξύ τους (τάξεις μεγέθους).

Κανονικοποίηση

Κατόπιν μπορεί να γίνει μια κανονικοποίηση των τιμών του πίνακα χρησιμοποιώντας μία από τις γνωστές νόρμες από τα μαθηματικά (π.χ. τη νόρμα μεγίστου), αν και στα πειράματα που πραγματοποιήθηκαν δεν φάνηκε να έχει σημαντική αξία. Ο σημαντικότερος λόγος για αυτό είναι ότι, συνήθως, οι διαφορές που υπάρχουν μεταξύ των τριπλετών όσον αφορά τα πλήθη των προσπελάσεων δεν είναι τόσο μεγάλες και κυμαίνονται στις ίδιες περίπου τάξεις μεγέθους.

Ενναλακτικά, μπορεί κανείς να αντικαταστήσει το κάθε στοιχείο x του πίνακα με την *ανηγμένη τιμή* του x^*, δηλαδή με

$$x^* = \frac{x - \mu}{\sigma}$$

όπου μ είναι η μέση τιμή και σ η τυπική απόκλιση όλων των σημείων του πίνακα, δηλαδή

$$\mu = \frac{1}{n \times k} \sum_{i=1}^{n} \sum_{j=1}^{k} A_{i,j}$$

όπου n είναι το πλήθος των τριπλετών, k το πλήθος των σελίδων της βάσης δεδομένων και $A_{i,j}$ το σύνολο των προσπελάσεων που κάνει η τριπλέτα i στη σελίδα j.

$$\sigma = \sqrt{\frac{1}{n \times k} \sum_{i=1}^{n} \sum_{j=1}^{k} (A_{i,j} - \mu)^2}$$

Διάταξη

Το πιο σημαντικό κομμάτι της προεπεξεργασίας των στοιχείων του πίνακα προσπελάσεων είναι η *διάταξη* (sorting) που τους γίνεται, γιατί υπάρχουν αλγόριθμοι (όπως ο *HALC* και ο ISODATA) που είναι αρκετά ευαίσθητοι στην αρχική διάταξη των στοιχείων.

Με τη διάταξη των τριπλετών του πίνακα η προσπάθεια είναι να τοποθετηθούν οι τριπλέτες που πιθανότατα έχουν παρόμοιες προσπελάσεις στην βάση δεδομένων όσο το δυνατόν πιο κοντά μεταξύ τους στον πίνακα. Ο λόγος είναι ότι με αυτόν τον τρόπο θα είναι πιο εύκολο για τον κάθε αλγόριθμο να εντοπίσει τις τριπλέτες που πρέπει να ομαδοποιηθούν μαζί.

Στο θέμα της διάταξης των στοιχείων δύο είναι οι παράμετροι:

- Ο *αλγόριθμος* διάταξης, ο οποίος επηρεάζει μόνο την ταχύτητα εκτέλεσης του *CLUE* και όχι την ποιότητα των αποτελεσμάτων. Στην παρούσα υλοποίηση χρησιμοποιήθηκε ο γνωστός αλγόριθμος HEAPSORT.

- Το *κριτήριο* διάταξης, το οποίο επηρεάζει άμεσα την ποιότητα των αποτελεσμάτων.

Τρία ήταν τα κριτήρια διάταξης που υλοποιήθηκαν:

- Το πρώτο κριτήριο (REF, από το REFerences) διατάσει τις τριπλέτες σύμφωνα το *συνολικό πλήθος προσπελάσεων* που κάνουν στη βάση δεδομένων. Το σκεπτικό πίσω από αυτή την απόφαση είναι ότι, με μεγάλη πιθανότητα, τριπλέτες που έχουν το ίδιο περίπου πλήθος προσπελάσεων θα είναι και τριπλέτες που θα έχουν παρόμοιες προσπελάσεις, άρα θα πρέπει να τοποθετηθούν κοντά στον πίνακα προσπελάσεων.

- Το δεύτερο κριτήριο (LEX, από το LEXicographically) υλοποιεί τη *λεξικογραφική διάταξη* στις τριπλέτες. Η διάταξη αυτή προκύπτει αν θεωρήθεί η κάθε στήλη του πίνακα ως μία

ξεχωριστή ιδιότητα (όπως ακριβώς το 1ο, το 2ο, ... γράμμα σε λέξεις που διατάσσονται λεξικογραφικά).

Αν πριν διατάξουμε λεξικογραφικά τις τριπλέτες, διατάξουμε τις στήλες με φθίνουσα σειρά *συνολικού αριθμού προσπελάσεων*, τότε προκύπτει ένα άλλο κριτήριο (LEXDT, από το LEXicographically after Data are sorted by Total number of references). Στο κριτήριο αυτό η λεξικογραφική σειρά εφαρμόζεται και πάλι, αλλά αυτή τη φορά οι σημαντικές ιδιότητες που είναι υπεύθυνες για τις μεγάλες διαφορές στην διάταξη (το ρόλο με άλλα λόγια των πρώτων γραμμάτων στη λεξικογραφική διάταξη λέξεων) είναι οι σελίδες που έχουν μεγάλο συνολικό αριθμό προσπελάσεων.

Μια παραλλαγή του παραπάνω είναι αντί το κριτήριο διάταξης των σελίδων να είναι ο συνολικός αριθμός προσπελάσεων στην κάθε μία, να είναι το πόσο *δημοφιλής* είναι η συγκεκριμένη σελίδα, δηλαδή, πόσες τριπλέτες προσπελαύνουν τη σελίδα αυτή (ανεξάρτητα δηλαδή από αριθμό προσπελάσεων). Το κριτήριο αυτό ονομάστηκε (LEXDP, από το LEXicographically after Data are sorted by Popularity).

- Το τρίτο κριτήριο (BEA) υλοποιεί την διάταξη που προτείνεται στον *αλγόριθμο ενέργειας δεσμού*. Είναι το καλύτερο ποιοτικά κριτήριο, αλλά χρειάζεται και αρκετό χρόνο για να ολοκληρώσει τη διάταξη των στοιχείων.

3.2.2 Μέτρηση αποστάσεων

Η μέτρηση αποστάσεων μεταξύ των τριπλετών είναι θεμελειώδης για κάθε αλγόριθμο ομαδοποίησης και υλοποιήθηκε με όσο το δυνατόν περισσότερο αποδοτικό τρόπο. Υλοποιήθηκαν δύο συναρτήσεις υπολογισμού απόστασης: το Pseudo Linear Dependency Metric και το Vector Distance Metric.

	D_1	D_2	D_3	D_4
T_1	10	15	0	0
T_2	20	10	0	0
T_3	0	20	20	30
T_4	0	0	25	20
T_5	0	0	25	15

Πίνακας 3.1: Πίνακας Προσπελάσεων

Pseudo Linear Dependency Metric

Η πρώτη συνάρτηση μέτρησης απόστασης μεταξύ τριπλετών ονομάζεται Pseudo Linear Dependency Metric (PLDM) και "κατάγεται" από τον τύπο της γραμμικής εξάρτησης πινάκων, αλλά κατέληξε σε κάτι τελείως διαφορετικό.

Συγκεκριμένα, η απόσταση α μεταξύ δύο τριπλετών x και y ορίζεται ως:

$$\alpha = \max_{i=1,\ldots,k} \left\{ \begin{array}{ll} 0 & \text{αν } x_i = y_i = 0 \\ \frac{\|x_i - y_i\|}{\max(x_i, y_i)} & \text{αλλιώς} \end{array} \right\}$$

όπου k είναι το πλήθος των σελίδων της βάσης δεδομένων, και x_i είναι το πλήθος των προσπελάσεων που η τριπλέτα x κάνει στη σελίδα i.

Εύκολα παρατηρεί κανείς πως ισχύει πάντα ότι:

$$\alpha \in [0, 1]$$

	T_1	T_2	T_3	T_4	T_5
T_1	0	0.5	1	1	1
T_2	0.5	0	1	1	1
T_3	1	1	0	1	1
T_4	1	1	1	0	0.25
T_5	1	1	1	0.25	0

Πίνακας 3.2: Πίνακας αποστάσεων PLDM

Έστω ότι ο πίνακας προσπελάσεων είναι αυτός που φαίνεται στον πίνακα 3.1, όπου εύκολα διακρίνονται δύο ξεχωριστές κλάσεις με παρόμοιες προσπελάσεις στη βάση δεδομένων (η μία κλάση έχει τις T_1 και T_2, ενώ η δεύτερη κλάση έχει τις T_4 και T_5), καθώς και μία τριπλέτα που δεν μοιάζει πολύ σε καμία από τις δύο κλάσεις, αλλά είναι περισσότερο "κοντά" στην δεύτερη κλάση.

Ο πίνακας που δείχνει τις αποστάσεις σύμφωνα με την συνάρτηση απόστασης PLDM θα είναι ο 3.2. Παρατηρούμε ότι ο πίνακας είναι συμμετρικός ως προς τη διαγώνιό του (κάτι το οποίο είναι αναμενόμενο) η οποία έχει όλα τα στοιχεία ίσα με 0.

Η συνάρτηση απόστασης PLDM είναι ιδιαίτερα "αυστηρή", αφού η απόσταση γίνεται ίση με τη μέγιστη τιμή της (= 1), ακόμα και όταν απλά η μία από τις δύο υπό σύγκριση τριπλέτες έχει πρόσβαση σε ένα δεδομένο που δεν έχει η άλλη. Η αυστηρότητα αυτού του κριτηρίου χρησιμοποιείται από τον *HALC* για να κατασκευαστούν "καλές" ομάδες τριπλετών, με παρόμοιες προσπελάσεις στη βάση δεδομένων και κατόπιν χρησιμοποείται το επόμενο

κριτήριο για να συνεχίσει την ομαδοποίηση και να καταλήξει ο αλγόριθμος στον απαιτούμενο αριθμό κλάσεων.

Πραγματικά μπορεί κανείς να δει από το παράδειγμα ότι η συνάρτηση απόστασης PDLM είναι αρκετά αυστηρή, αφού υπολόγισε απόσταση κάτω του 1 μόνο μεταξύ στοιχείων που είναι τελείως όμοια.

Vector Distance Metric

Η δεύτερη συνάρτηση μέτρησης απόστασης ονομάζεται Vector Distance Metric (VDM) και δεν είναι άλλη από την κλασσική ευκλείδεια απόσταση.

Συγκεκριμένα, η απόσταση α μεταξύ δύο τριπλετών x και y ορίζεται ως:

$$\alpha = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

όπου k είναι το πλήθος των σελίδων της βάσης δεδομένων, και x_i είναι το πλήθος των προσπελάσεων που η τριπλέτα x κάνει στη σελίδα i.

Αυτή η συνάρτηση μέτρησης απόστασης θεωρεί την κάθε τριπλέτα ως ένα διάνυσμα σε ένα $k-$διάστατο χώρο, όπου κάθε διάσταση αντιστοιχεί στη σελίδα D_i της βάσης δεδομένων.

	T_1	T_2	T_3	T_4	T_5
T_1	0.0	11.2	37.8	36.8	34.3
T_2	11.2	0.0	42.2	39.0	36.8
T_3	37.8	42.2	0.0	22.9	25.5
T_4	36.8	39.0	22.9	0.0	5.0
T_5	34.3	36.8	25.5	5.0	0.0

Πίνακας 3.3: Πίνακας αποστάσεων VDM

Έστω πάλι ότι ο πίνακας προσπελάσεων είναι αυτός που φαίνεται στον πίνακα 3.1, όπου εύκολα διακρίνονται δύο ξεχωριστές κλάσεις με παρόμοιες προσπελάσεις στη βάση δεδομένων (η μία κλάση έχει τις T_1 και T_2, ενώ η δεύτερη κλάση έχει τις T_4 και T_5), καθώς και μία τριπλέτα που δεν μοιάζει πολύ σε καμία από τις δύο κλάσεις, αλλά είναι περισσότερο ‘‘κοντά’’ στην δεύτερη κλάση.

Ο πίνακας που δείχνει τις αποστάσεις σύμφωνα με την συνάρτηση απόστασης VDM θα είναι ο 3.3. Παρατηρούμε ότι ο πίνακας είναι συμμετρικός ως προς τη διαγώνιό του (κάτι το οποίο είναι αναμενόμενο), η οποία έχει όλα τα στοιχεία ίσα με 0.

Εύκολα διαπιστώνει κανείς ότι υπάρχουν τρεις κατηγορίες τιμών αποστάσεων μεταξύ των τριπλετών στον πίνακα 3.3:

- *Σχετικά μικρές τιμές*, (π.χ. 5.0 και 11.2) όπου είναι προφανής η ανάγκη για ομαδοποίηση των αντίστοιχων τριπλετών,
- *Σχετικά μεγάλες τιμές*, (π.χ. 37.8 και 42.2) όπου είναι προφανές ότι δεν μπορεί να υπάρξει ομαδοποίηση των αντίστοιχων τριπλετών,
- *Μεσαίου μεγέθους τιμές*, (π.χ. 22.9 και 25.5) όπου δεν είναι προφανής η απόφαση που πρέπει να παρθεί.

Με άλλα λόγια, η ευκλείδεια απόσταση δίνει τη δυνατότητα να ομαδοποιηθούν μαζί ακόμα και τριπλέτες που μοιάζουν σχετικά λίγο, όσον αφορά τις προσπελάσεις τους στη βάση δεδομένων.

3.2.3 Διάκριση αναγνώσεων – ενημερώσεων

Στα στοιχεία για τις προσπελάσεις που κάνουν οι διάφορες δοσοληψίες σε σελίδες της βάσης δεδομένων υπάρχει η διάκριση αναγνώσεων (reads) / εγγραφών (writes). Η διάκριση αυτή διατηρείται όταν κατασκευάζεται ο πίνακας προσπελάσεων (σχήμα 3.2) καθώς και σε όλες τις πράξεις που γίνονται πάνω στον πίνακα (π.χ. εύρεση μέσων όρων στις προσπελάσεις σε μία σελίδα).

Στον υπολογισμό των αποστάσεων, οι δύο διαφορετικές τιμές προσπελάσεων (άλλη για αναγνώσεις και άλλη για εγγραφές στην ίδια σελίδα) ενώνοντα σε ένα νούμερο και χρησιμοποιούνται για να γίνουν οι όποιες συγκρίσεις. Η τιμή αυτή υπολογίζεται με βάση σχετικά βάρη, και συγκεκριμένα σύμφωνα με τον τύπο:

$$\text{σύνολο προσπελάσεων} = \frac{\beta_\alpha \times \Sigma_\alpha + \beta_\varepsilon \times \Sigma_\varepsilon}{\frac{\beta_\alpha + \beta_\varepsilon}{2}} = 2 \times \frac{\beta_\alpha \times \Sigma_\alpha + \beta_\varepsilon \times \Sigma_\varepsilon}{\beta_\alpha + \beta_\varepsilon}$$

όπου Σ_α είναι το σύνολο των αναγνώσεων,
Σ_ε είναι το σύνολο των εγγραφών,
β_α είναι το σχετικό βάρος των αναγνώσεων,
β_ε είναι το σχετικό βάρος των εγγραφών.

Η διάκριση μεταξύ αναγνώσεων και εγγραφών, εκτός από την διαφορά στην βαρύτητα κατά την εύρεση του συνόλου προσπελάσεων, μπορεί να έχει και άλλη χρησιμότητα. Με το να υπάρχουν ξεχωριστά στοιχεία για τις αναγνώσεις και τις εγγραφές, μπορούν εύκολα να εντοπιστούν οι περιπτώσεις σελίδων που προσπελαύνονται μόνο για ανάγνωση (read–only pages), κάτι το οποίο μπορεί να βοηθήσει τον αλγόριθμο σε ορισμένες περιπτώσεις (βλέπε Κεφάλαιο 5).

3.2.4 Αλγόριθμοι

Η χρήση της γλώσσας περιγραφής πειραμάτων επέτρεψε την χρήση πολλών διαφορετικών αλγορίθμων ομαδοποίησης στο περιβάλλον του *CLUE*. Έτσι, εκτός από τον *HALC* (για τον οποίο θα υπάρχει λεπτομερής περιγραφή στο επόμενο υποκεφάλαιο), υλοποιήθηκε επιπλέον ο αλγόριθμος Bond Energy, και ο αλγόριθμος ISODATA.

3.3 Ο ευρηματικός αλγόριθμος *HALC*

Ο αλγόριθμος *HALC* (από το Heuristic Algorithm for Clustering)[2] είναι ένας απλός και γρήγορος ευρηματικός αλγόριθμος για ομαδοποίηση δοσοληψιών με παρόμοια χαρακτηριστικά φόρτου εργασίας.

Η ομαδοποίηση που κάνει είναι *αυτόματη*[3] με την έννοια ότι η απόσταση ομαδοποίησης αναπροσαρμόζεται από τον ίδιο τον αλγόριθμο και όχι από κάποιον εξωτερικό παράγοντα. Ο αλγόριθμος είναι συγχωνευτικός:
Αρχικά, όλες οι τριπλέτες αποτελούν μια κλάση η κάθε μία ξεχωριστά και σε κάθε επανάληψη (iteration) του αλγόριθμου ενοποιούνται οι κλάσεις με παρόμοιες προσπελάσεις, ώσπου να καταλήξει ο αλγόριθμος σε τόσες κλάσεις, όσες είχε ζητήσει ο χρήστης.

Δύο ξεχωριστές φάσεις εκτελούνται σε κάθε επανάληψη του αλγορίθμου:

1. Το *Βασικό Βήμα Ομαδοποίησης* (Elementary Clustering Step, ή *ECStep*), το οποίο είναι ένα πέρασμα (scan) από όλες τις τριπλέτες μια φορά και η ενοποίηση στην ίδια κλάση αυτών που απέχουν λιγότερο από την τρέχουσα απόσταση ομαδοποίησης (clustering distance) η οποία διατηρείται σταθερή σε όλη τη διάρκεια του *ECStep*.

2. Ο *επαναπροσδιορισμός της απόστασης ομαδοποίησης* (*RACDIN*, από το Re−Adjust Clustering Distance If Necessary), σύμφωνα με κάποια κριτήρια. Στη φάση αυτή παίρνεται επίσης η απόφαση για τη συνέχιση ή τον τερματισμό της ομαδοποίησης.

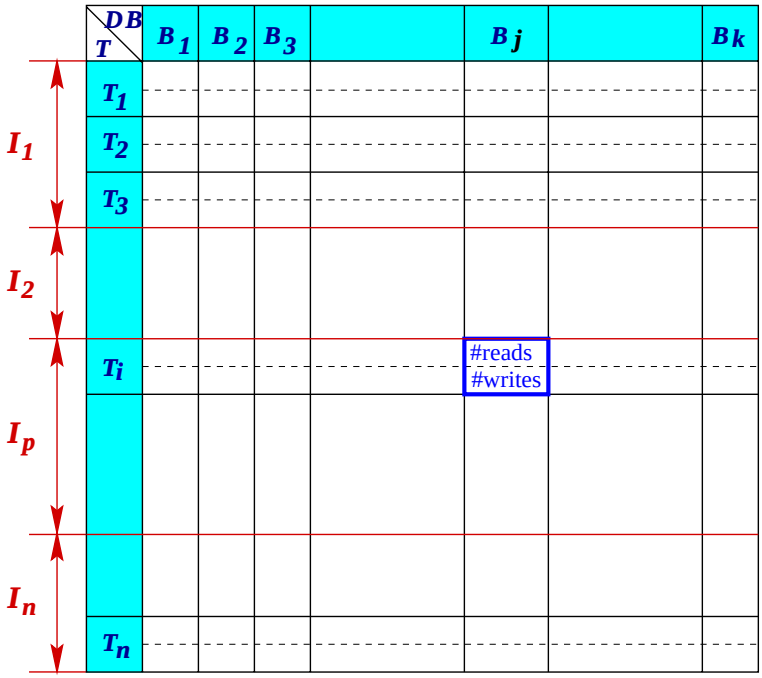
3.3.1 Το Βασικό Βήμα Ομαδοποίησης

Η βασική ιδέα του αλγόριθμου είναι ότι, δοθείσης μιας καλής προεπεξεργασίας, τριπλέτες με παρόμοιες προσπελάσεις στη βάση δεδομένων θα πρέπει να έχουν τοποθετηθεί (μετά το

[2] Μια διαφορετική, "ανεπίσημη" εξήγηση για την επιλογή του ονόματος είναι το HAL Clustering. HAL ήταν το όνομα του Η/Υ στην ταινία η "Οδύσσεια του Διαστήματος" και ως γνωστόν αντικαθιστώντας κανείς τα γράμματα του HAL με τα επόμενα γράμματα στην αλφαβήτα σχηματίζει τη λέξη IBM. Με τον τρόπο αυτό δηλώνεται η στενή σχέση του αλγόριθμου αυτού με την IBM, όπου είχε προταθεί αρχικά.

[3] θα μπορούσε να χαρακτηριστεί *ήμι−αυτόματη* δεδομένου ότι υπάρχουν πολλές παράμετροι που καθορίζουν τη συμπεριφορά του αλγόριθμου. Οι τιμές όμως των παραμέτρων αυτών έχουν παγιωθεί πλέον (με την εξαίρεση βέβαια του επιθυμητού πλήθους κλάσεων), οπότε δεν είναι απαραίτητο να γίνει οποιαδήποτε αλλαγή τους από τον χρήστη.

στάδιο διάταξης) σχετικά κοντά στον πίνακα προσπελάσεων. Σύμφωνα με αυτό, ο αλγόριθμος σε κάθε βασικό βήμα ομαδοποίησης χωρίζει το σύνολο των τριπλετών σε *διαστήματα* (intervals) και συγκρίνει τις αποστάσεις των τριπλετών που βρίσκονται μόνο μέσα στο ίδιο διάστημα (βλέπε σχήμα 3.4).



Σχήμα 3.4: Πίνακας Προσπελάσεων χωρισμένος σε Διαστήματα

Ο χωρισμός του αρχικού συνόλου των τριπλετών σε διαστήματα τυχαιοποιείται για να μην επηρεάσει τα αποτελέσματα της ομαδοποίησης. Το μέγεθος που θα έχει το κάθε διάστημα αλλάζει δυναμικά όσο προχωράει ο αλγόριθμος και μειώνεται το πλήθος των κλάσεων πλησιάζοντας όλο και περισσότερο στο επιθυμητό όριο. Συγκεκριμένα, σε κάθε επανάληψη του αλγορίθμου, υπολογίζεται μια τιμή βάσης για το μέγεθος που θα έχουν τα διαστήματα και το μέγεθος του κάθε διαστήματος αποφασίζεται από την τιμή βάσης και κάποια επιτρεπόμενη παρέκκλιση θετική ή αρνητική από αυτήν.

Η τιμή βάσης για το μέγεθος των διαστημάτων ισούται:

$$\text{τιμή βάσης} = \frac{\text{τρέχων αριθμός κλάσεων}}{\text{επιθυμητό πλήθος κλάσεων}}$$

Η τιμή αυτή αντανακλά την ανάγκη να συρρικνωθούν όλες οι κλάσεις από ένα τέτοιου μεγέθους διάστημα σε μια κλάση τελικά, ούτως ώστε να πετύχουμε το επιθυμητό πλήθος.

Κάθε υποβήμα επεξεργάζεται ένα διάστημα. Σε κάθε υποβήμα, το κορυφαίο μόνο στοιχείο ενός διαστήματος (έστω X) συγκρίνεται με όλα τα υπόλοιπα (έστω Y_i) που υπάρχουν στο

ίδιο διάστημα. Αν για κάποιο ζευγάρι στοιχείων (X, Y_i) η απόσταση βρεθεί μικρότερη από την τρέχουσα απόσταση ομαδοποίησης, τότε ομαδοποιούνται στην κλάση του κορυφαίου στοιχείου X. Κατά την ενοποίηση των δύο στοιχείων, η δεύτερη από τις δύο εξαφανίζεται και η πρώτη ενημερώνεται κατάλληλα ώστε να αποτελέσει το νέο κέντρο της κλάσης.

Όπως είχε αναφερθεί και προηγούμενα, αρχικά υπάρχουν τόσες κλάσεις όσες και οι τριπλέτες, και σιγά σιγά οι κλάσεις ενοποιούνται σε μεγαλύτερες κλάσεις μέχρι να φτάσει ο αριθμός των κλάσεων στο επιθυμητό όριο. Κάθε φορά που ομαδοποιούνται δύο κλάσεις X και Y μαζί, υπάρχουν δύο επιλογές όσον αφορά την συνολική συμπεριφορά σε προσπελάσεις στη βάση δεδομένων της παραγόμενης κλάσης X':

- *Άθροιση* όλων των προσπελάσεων των δύο κλάσεων.
- Εύρεση του *μέσου όρου* των προσπελάσεων των δύο κλάσεων, σύμφωνα με τον τύπο:

$$X'_i = \frac{|X| * X_i + |Y| * Y_i}{|X| + |Y|}, \ \ \forall i = 1, \ldots, n$$

  όπου |X| είναι το πλήθος των μελών της κλάσης X, |Y| είναι το πλήθος των μελών της κλάσης Y και n είναι το πλήθος των σελίδων της βάσης δεδομένων.

Αξίζει, τέλος να σημειωθεί ότι η ενοποίηση δύο κλάσεων είναι μόνιμη και δεν μπορεί να αντιστραφεί.

Η σύγκριση μόνο του κορυφαίου στοιχείου του διαστήματος με τα υπόλοιπα και όχι όλων των στοιχείων του διαστήματος μεταξύ τους, βασίζεται και πάλι στην ίδια ιδέα που οδήγησε στον χωρισμό του αρχικού συνόλου σε διάστήματα καθώς επίσης και στην τυχαιότητα που έχει επιβληθεί κατά την κατασκευή αυτών των διαστημάτων.

3.3.2 Αναπροσαρμογή απόστασης ομαδοποίησης

Υπάρχουν δύο αντικρουόμενα κριτήρια για να βοηθήσουν στην αυτόματη αναπροσαρμογή της απόστασης ομαδοποίησης από τον *HALC*.

Κριτήριο Ποιότητας

Το κριτήριο για τον έλεγχο της ποιότητας της ομαδοποίησης συνοψίζεται στο:
"αν η συμπίεση του πλήθους των κλάσεων σε ένα βασικό βήμα ομαδοποίησης ανεβεί ένα ποσοστό (π.χ. 10%) τότε θα πρέπει να *μειωθεί* η απόσταση ομαδοποίησης".

Αυτό γίνεται για να προστατευθεί ο αλγόριθμος από περιπτώσεις όπου για κάποιο λόγο η απόσταση ομαδοποίησης έχει μεγαλώσει αρκετά, σε βαθμό που να ομαδοποιούνται κλάσεις που δεν έχουν αρκετά παρόμοιες προσπελάσεις στη βάση δεδομένων, και άρα να υποβαθμίζεται με αυτόν τον τρόπο η ποιότητα της παραγόμενης ομαδοποίησης.

Κριτήριο Ποσότητας

Το κριτήριο για τον έλεγχο της ποσότητας της ομαδοποίησης συνοψίζεται στο:
"αν η συμπίεση του πλήθους των κλάσεων σε ένα βασικό βήμα ομαδοποίησης γίνει μικρότερη από ένα ποσοστό $k\%$ της αρχικής συμπίεσης του πλήθους των κλάσεων, τότε θα πρέπει να *αυξηθεί* η απόσταση ομαδοποίησης".

Αυτό γίνεται για να προστατευθεί ο αλγόριθμος από περιπτώσεις όπου για κάποιο λόγο η απόσταση ομαδοποίησης έχει μειωθεί αρκετά, σε βαθμό που να μην ομαδοποιούνται αρκετές κλάσεις σε κάθε βήμα, και άρα να υποβαθμίζεται με αυτόν τον τρόπο η ταχύτητα σύγκλισης του αλγορίθμου (με άλλα λόγια η ποσότητα των ομαδοποιήσεων).

Ο ορισμός της αρχικής συμπίεσης μπορεί να δεχτεί τέσσερις διαφορετικές ερμηνείες:

- Μπορεί να είναι η συμπίεση που θα παρατηρηθεί στην *πρώτη επανάληψη* του αλγορίθμου. Έτσι, η όποια σύγκριση θα γίνεται με βάση την πρώτη συμπίεση η οποία είναι αρκετά ενδεικτική. Το πρόβλημα με την προσέγγιση αυτή είναι ότι η πρώτη συμπίεση εξαρτάται πολύ από την αρχική απόσταση ομαδοποίησης, η οποία μπορεί προφανώς να απέχει αρκετά από την κανονική απόσταση ομαδοποίησης, οπότε η αρχική συμπίεση είναι αναξιόπιστη (μπορεί για παράδειγμα να είναι 0).

- Μπορεί να είναι η *μέγιστη συμπίεση* που έχει παρατηρηθεί μέχρι τώρα. Η προσέγγιση αυτή έχει το πλεονέκτημα ότι βασίζεται στην συμπεριφορά του αλγορίθμου που έχει παρατηρηθεί μέχρι εκείνη τη στιγμή σε σχέση με τα συγκεκριμένα δεδομένα του προβλήματος.

- Μπορεί να είναι η *μέση συμπίεση* που έχει παρατηρηθεί μέχρι τώρα. Εκτός από το πλεονέκτημα της προηγούμενης προσέγγισης, η προσέγγιση αυτή έχει το επιπλέον πλεονέκτημα ότι είναι αρκετά ρεαλιστική σε αντίθεση με την προηγούμενη προσέγγιση που βασιζόταν στη μέγιστη τιμή.

- Μπορεί, τέλος, να είναι μια τιμή συμπίεσης που *θα προσδιορίζει ο χρήστης*. Η προσέγγιση αυτή δίνει πλήρη ελευθερία στο χρήστη, αλλά έχει το μειονέκτημα ότι θα πρέπει ο χρήστης να έχει την απαιτούμενη εμπειρία και εξοικείωση με το πρόβλημα ώστε να διαλέξει μια καλή τιμή αρχικής συμπίεσης.

Συνολική εικόνα

Ο αλγόριθμος ξεκινάει με την αρχική απόσταση ομαδοποίησης να παρέχεται από το χρήστη ή να είναι μια καλή πρόβλεψη από τον ίδιο τον αλγόριθμο. Κατόπιν, σε κάθε επανάληψη εκτε–λείται ένα βασικό βήμα ομαδοποίησης και έπειτα γίνεται μια αξιολόγηση των αποτελεσμάτων

της ομαδοποίησης με βάση τα δύο προαναφερθέντα κριτήρια, όπου αποφασίζεται κατά πόσο πρέπει να γίνει κάποια αλλαγή. Σε περίπτωση που αποφασιστεί ότι κάτι πρέπει να αλλάξει, εκτελείται μια προσομοίωση ενός βασικού βήματος ομαδοποίησης με τη νέα απόσταση ομαδοποίησης. Η προσομοίωση αυτή δεν είναι τίποτα άλλο από την κανονική εκτέλεση ενός βασικού βήματος με τη διαφορά ότι δεν γίνονται οι όποιες ομαδοποιήσεις αποφασιστούν, αλλά απλά αυξάνεται ένας μετρητής. Κατόπιν τα αποτελέσματα επαναξιολογούνται και η διαδικασία επαναλαμβάνεται.

Για να εξασφαλιστεί η "λογική" συμπεριφορά του αλγορίθμου, έχουν ενσωματωθεί κάποιες παράμετροι και κάποιες επιπλέον ασφαλιστικές δικλείδες στην προσπάθεια επαναπροσδιορισμού της απόστασης ομαδοποίησης. Μπορεί για παράδειγμα να οριστεί ένα κάτω και ένα πάνω φράγμα για την απόσταση ομαδοποίησης, ούτως ώστε να μην ξεπεραστεί ποτέ από τον αλγόριθμό (π.χ. στην περίπτωση της συνάρτησης απόστασης PLDM, απόσταση πάνω από 1 δεν έχει νόημα).

Επιπλέον, κάθε αύξηση / μείωση της απόστασης ομαδοποίησης γίνεται με βάση ένα ποσοστό αύξησης / μείωσης, το οποίο μπορεί να προσδιοριστεί από το χρήστη. Επίσης, υπάρχει έλεγχος για να μην ξεπεράσει οποιαδήποτε αλλαγή της απόστασης ομαδοποίησης σε ένα μόνο βήμα ένα προκαθορισμένο ποσοστό.

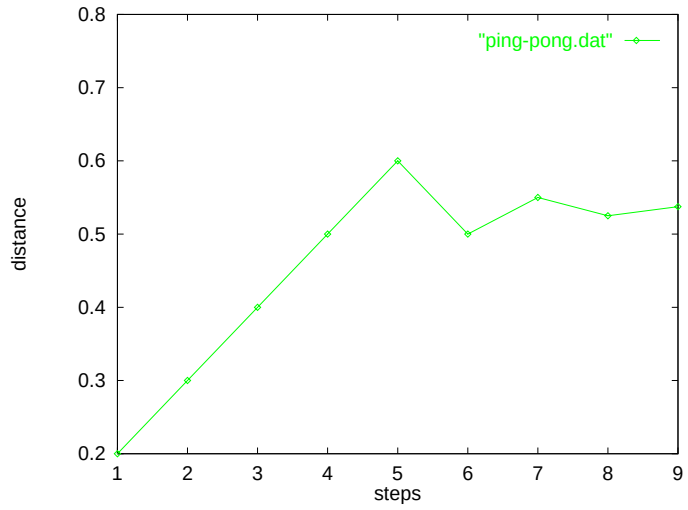
Άλλες τρεις επεκτάσεις στη βασική ιδέα του αλγόριθμου είναι ιδιαίτερα σημαντικές. Η πρώτη είναι ο τρόπος με τον οποίο αντιμετωπίζονται τα *ping–pongs*.

Ως ping–pong χαρακτηρίζεται η ακόλουθη περίπτωση: όσο ο αλγόριθμος βρίσκεται σε φάση επαναπροσδιορισμού της απόστασης ομαδοποίησης, να αποφασίζεται συνέχεια αύξηση (μείωση) της απόστασης ομαδοποίησης και κάποια στιγμή να αποφασιστεί μείωση (αύξηση) της απόστασης (στην ίδια εκτέλεση του *RACDIN*). Εμφανίζεται δηλαδή η περίπτωση όπου η αύξηση (μείωση) στην απόσταση είναι μεγαλύτερη από την πρέπουσα και πρέπει να μειωθεί (αυξηθεί) λιγάκι.

Αυτό συμβαίνει συνήθως όταν λόγω του ενός από τα δύο κριτήρια αποφασίζεται η συνεχόμενη αύξηση (μείωση) της απόστασης ομαδοποίησης, και όταν ικανοποιηθεί το εν λόγω κριτήριο, το άλλο κριτήριο παύει να ικανοποιείται πλέον, και προσπαθεί να αλλάξει εκ νέου την απόσταση ομαδοποίησης. Εύκολα καταλαβαίνει κανείς ότι από αυτή την περίπτωση μπορεί να καταλήξει ο αλγόριθμος σε ατελείωτη ανακύκλωση, οπότε θα πρέπει να προβλεφθεί μια συνθήκη τερματισμού.

Υπάρχουν έξι δυνατότητες για τη λύση του προβλήματος ping–pong:

- Απλά να υπολογιστεί ο *μέσος όρος* των δύο τελευταίων αποστάσεων και να τερματίσει η διαδικασία επαναπροσδιορισμού της απόστασης ομαδοποίησης.
- Να επιτραπεί να συνεχίσει ο αλγόριθμος, αλλά από εκείνη τη στιγμή και έπειτα, οι όποιες



Σχήμα 3.5: Λύση προβλήματος ping−pong

αυξομειώσεις στην απόσταση ομαδοποίησης να γίνονται με τα μισά από τα προηγούμενα ποσοστά, ώστε να είναι περισσότερο λεπτομερείς. Η διαδικασία αυτή μάλιστα μπορεί να επαναληφθεί όσες φορές έχει προσδιορίσει ο χρήστης ώστε η απόσταση ομαδοποίησης να συγκλίνει σε μία καλύτερη τιμή (σχήμα 3.5).

- Να διατηρηθεί η *τελευταία* απόσταση ομαδοποίησης και να τερματίσει η διαδικασία επαναπροσδιορισμού της.
- Να διατηρηθεί η *προηγούμενη* απόσταση ομαδοποίησης και να τερματίσει η διαδικασία επαναπροσδιορισμού της.
- Να διατηρηθεί η *μεγαλύτερη* απόσταση ομαδοποίησης μεταξύ της τελευταίας και της προηγούμενης και να τερματίσει η διαδικασία επαναπροσδιορισμού της.
- Να διατηρηθεί η *μικρότερη* απόσταση ομαδοποίησης μεταξύ της τελευταίας και της προηγούμενης και να τερματίσει η διαδικασία επαναπροσδιορισμού της.

Η δεύτερη επέκταση έχει να κάνει με τον τερματισμό του αλγόριθμου. Έτσι, ο αλγόριθμος μπορεί να τερματίσει (χωρίς να έχει κατασκευαστεί ο επιθυμητός αριθμός κλάσεων) σε περιπτώσεις όπου επανειλημμένες δραστικές αλλαγές στην απόσταση ομαδοποίησης δεν φέρνουν καμία αλλαγή στον αριθμό των κλάσεων.

Τέλος, ενσωματώθηκε στον αλγόριθμο η έννοια της προσέγγισης στον τελικό στόχο. Με αυτό τον τρόπο, όταν ο *HALC* είναι κοντά στον τελικό στόχο (π.χ. το πλήθος των κλάσων διαφέρει μόλις κατά ένα μικρό ποσοστό 3% από το επιθυμητό πλήθος κλάσεων), παύει να

εξετάζει το κριτήριο ποσότητας αφού δεν έχει νόημα πια η ομαδοποίηση πολλών κλάσεων σε κάθε βήμα. Έτσι αποφεύγονται δραστικές αυξήσεις της απόστασης ομαδοποίησης όταν ο αλγόριθμος βρίσκεται κοντά στον αρχικό στόχο.

3.4 Υλοποίηση

3.4.1 Γενικά

Η υλοποίηση ολόκληρου του *CLUE* (και του αλγορίθμου *HALC*) έγινε εξ ολοκλήρου σε *ANSI−C* με τη βοήθεια των εργαλείων *Lex* και *Yacc*. Συγκεκριμένα, το *CLUE* μεταφράστηκε με *gcc (version 2.3.3)* και δοκιμάστηκε σε υπολογιστές *SUN SparcStations*.

Η επιλογή της γλώσσας *C* έγινε για τρεις κυρίως λόγους:

1. Υπήρχε αρκετή εμπειρία στον προγραμματισμό στην γλώσσα αυτή από όλους όσους ασχολήθηκαν με το *CLUE* .

2. Η *C* δίνει τη δυνατότητα για τη συγγραφή γρήγορων και αποδοτικών προγραμμάτων, κάτι το οποίο είναι απαραίτητο όταν κανείς έχει να αντιμετωπίσει τεράστιους όγκους δεδομένων εισόδου.

3. Υπάρχει το διεθνές πρότυπο για τη γλώσσα *C*, οπότε η προσπάθεια μεταφοράς του *CLUE* σε άλλες υπολογιστικές πλατφόρμες δεν θα είναι τόσο δύσκολη.

Η χρήση των *Lex* και *Yacc* βοήθησε πάρα πολύ στην υλοποίηση ολόκληρου του περι−βάλλοντος, αφού απλουστεύτηκε κατά πολύ η λεξικογραφική ανάλυση και η ερμηνεία των εντολών του χρήστη.

3.4.2 Πίνακας Προσπελάσεων

Το πρώτο πρόβλημα που εμφανίστηκε στην αρχική υλοποίηση ήταν το πρόβλημα μνήμης. Συγκεκριμένα, εμφανίζονταν η ανάγκη για μεγάλες περιοχές μνήμης ώστε να χωρέσει ο πίνακας προσπελάσεων με πραγματικά δεδομένα και δεν υπήρχαν διαθέσιμες.

Τα τυπικά μεγέθη ήταν 2,000 τριπλέτες και 50,000 σελίδες της βάσης δεδομένων το οποίο χρειάζεται αν αποθηκευθεί σε έναν συνηθισμένο δισδιάστατο πίνακα περί τα

$$2,000 \times 50,000 \times 8bytes = 800MBytes$$

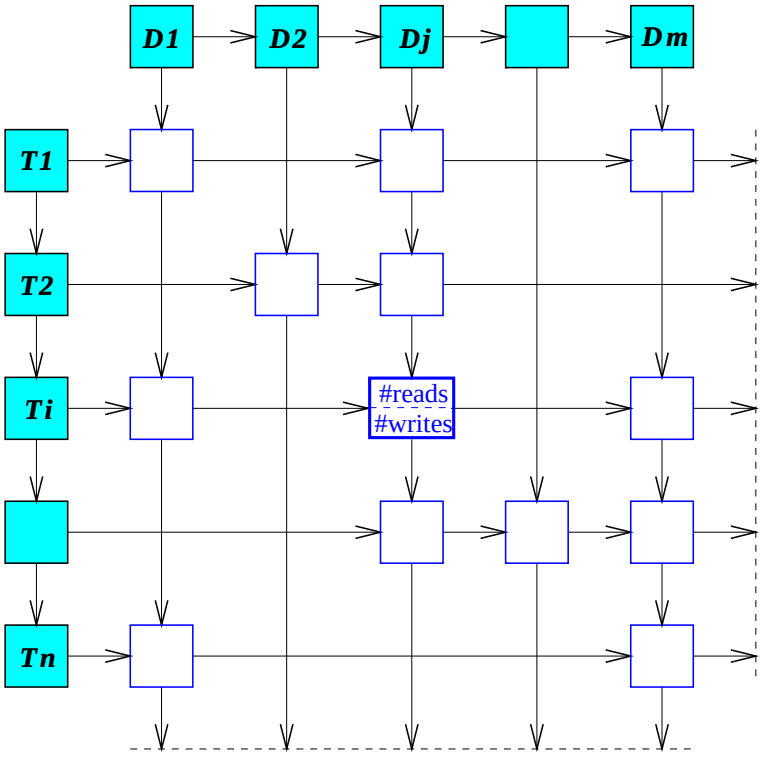
το οποίο είναι εξωφρενικά μεγάλο (τουλάχιστον για να δοθεί με ένα ενιαίο κομμάτι της κύριας μνήμης).

Δύο παρατηρήσεις έδωσαν τη λύση:

- Δεν χρειάζονται όλες οι γραμμές του πίνακα (τριπλέτες) στην κύρια μνήμη του υπολογιστή την ίδια χρονική στιγμή. Αυτό συμβαίνει, γιατί οι όποιες συγκρίσεις γίνονται μέσα στα *υποβήματα*, δηλαδή στα τμήματα εκείνα των βασικών βημάτων ομαδοποίησης, όπου γίνεται η επεξεργασία ενός μόνο διαστήματος τριπλετών.

  Με άλλα λόγια, σε οποιαδήποτε χρονική στιγμή απαιτούνται από τον αλγόριθμο μόνο οι γραμμές που αντιστοιχούν στις τριπλέτες του *διαστήματος υπό επεξεργασία*.

- Τα περισσότερα στοιχεία μιας γραμμής είναι μηδενικά, αφού η κάθε τριπλέτα έχει προσπελάσεις σε ένα πολύ μικρό τμήμα της βάσης. Δηλαδή ο πίνακας είναι *αραιός*.



Σχήμα 3.6: Αραιός Πίνακας Προσπελάσεων

Έτσι υλοποιήθηκε τελικά ο πίνακας προσπελάσεων ως δυναμική δομή δεδομένων με απλά συνδεδμένες λίστες σύμφωνα με τις γραμμές αλλά και σύμφωνα με τις στήλες (σχήμα 3.6).

Το βασικό πλεονέκτημα της προσέγγισης αυτής είναι ότι τώρα πλέον δεν υπάρχει πρόβλημα μνήμης, αφού με το να ‘‘σπάμε’’ τις απαιτήσεις σε μνήμη σε μικρά κομμάτια, ακόμα και αν δεν υπάρχει αρκετή κύρια μνήμη, θα υπάρχει αρκετή δευτερεύουσα μνήμη (swap memory).

Το βασικό μειονέκτημα της προσέγγισης αυτής όμως είναι η μείωση της ταχύτητας πρόσβασης σε οποιοδήποτε σημείο του πίνακα, αφού αφενός θα πρέπει κανείς να διατρέξει

μια λίστα και αφετέρου μπορεί το συγκεκριμένο σημείο να έχει σωθεί στο σκληρό δίσκο (να είναι δηλαδή swapped−out).

Παρόλα αυτά, ακόμα και αν ο συγκεκριμένος αλγόριθμος δεν έχει την αναμενόμενη συμπεριφορά στη διάταξη των προσβάσεων στον πίνακα προσπελάσεων, το κόστος θα είναι μόνο στην ταχύτητα και όχι στην ορθότητα των δεδομένων, αφού απλά τα δεδομένα που δεν υπάρχουν στην κύρια μνήμη θα υπάρχουν στο σκληρό δίσκο.

Τέλος, μια μέση λύση στο πρόβλημα της μειωμένης ταχύτητας είναι η αυτόματη επιλογή της κλασσικής δομής δεδομένων πίνακα σε περίπτωση που οι διαστάσεις του προβλήματος το επιτρέπουν (π.χ. για συνθετικά πειράματα). Με τον τρόπο αυτό η όχι ιδιαίτερα γρήγορη λύση (της υλοποίησης με απλά συνδεδεμένες λίστες) εφαρμόζεται μόνο στην περίπτωση που δεν μπορεί να γίνει διαφορετικά.

3.4.3 Γλώσσα Περιγραφής Πειραμάτων

Από την αρχή ήταν φανερό ότι με το *CLUE* θα έπρεπε να διεξαχθούν πολλά πειράματα για να συλλεγγούν μετρήσεις και να εξαχθούν κάποια αποτελέσματα. Επειδή οι παράμετροι τόσο του *HALC* όσο και του υπόλοιπου περιβάλλοντος ήταν αρκετές, η ιδέα της υλοποίησης μιας γλώσσας περιγραφής πειραμάτων και η χρησιμοποίησή της ήρθε γρήγορα στην επιφάνεια.

Με τη χρήση κάποιας τέτοιας γλώσσας αφενός υπάρχει μεγάλη ευκολία στην αλλαγή παραμέτρων χωρίς την εκ νέου μετάφραση του προγράμματος, και αφετέρου διευκολύνεται η ανάγκη για αυτόματη τεκμηρίωση όσων πειραμάτων γίνονται.

Η γλώσσα περιγραφής πειραμάτων (όπως αυτή υλοποιήθηκε τελικά) δίνει τις εξής δυνατότητες :

- Τον ορισμό όλων των παραμέτρων και του αλγόριθμου ομαδοποίησης που θα χρησιμοποιηθεί.
- Την επιλογή του είδους και της σειράς των βημάτων προεπεξεργασίας.
- Την εκτέλεση μερικών μόνο επαναλήψεων του αλγόριθμου και την αποθήκευση των μερικών αποτελεσμάτων σε κάποιο αρχείο στο δίσκο.
- Την επιλογή της κλάσης βοηθητικών σχολίων που θέλει ο χρήστης να εμφανίζονται κατά την εκτέλεση του προγράμματος (περισσότερες λεπτομέρειες στην επόμενη παράγραφο).
- Την εμφάνιση κάποιων προκαθορισμένων μηνυμάτων για να παρακολουθείται η πορεία εκτέλεσης του πειράματος.

- Τον ορισμό χρονοσφραγίδων (timestamps) με τις οποίες μπορεί να γίνει η χρονομέτρηση των διαφόρων σταδίων του πειράματος.

Για μια πλήρη λίστα των εντολών της γλώσσας περιγραφής πειραμάτων ο αναγνώστης παραπέμπεται στο Παράρτημα Γ.

3.4.4 Βοηθητικά μηνύματα

Αρκετή δουλειά έγινε και στον τομέα των βοηθητικών μηνυμάτων. Εκτός από την συγγραφή μιας παραμετρικής βιβλιοθήκης εμφάνισης μηνυμάτων, υλοποιήθηκε και ο διαχωρισμός σε διάφορες κλάσεις βοηθητικών μηνυμάτων. Σύμφωνα με αυτό το διαχωρισμό έχουν δημιουργηθεί αρκετές κατηγορίες μηνυμάτων ανάλογα με το είδος τους (συγκεκριμένα αν είναι μηνύματα υπολογισμού αποστάσεων, χωρισμού διαστημάτων, ροής του αλγόριθμου, επαναπροσδιορισμού της απόστασης ομαδοποίησης, κ.ο.κ.). Κάθε βοηθητικό μήνυμα ανήκει σε μία από αυτές τις κλάσεις και ανάλογα με το συγκεκριμένο πείραμα (αν δηλαδή έχει οριστεί στη γλώσσα περιγραφής πειραμάτων ότι η συγκεκριμένη κλάση βοηθητικών μηνυμάτων θα πρέπει να τυπώνεται) εμφανίζεται ή όχι, χωρίς βέβαια να χρειαστεί να γίνει εκ νέου μετάφραση του προγράμματος.

3.4.5 Μεταφερσιμότητα

Σημαντική δουλειά, τέλος, έγινε και στον τομέα της μεταφερσιμότητας. Η μεταφερσιμότητα επιτεύχθηκε εν μέρει από την επιλογή της γλώσσας προγραμματισμού και τη χρήση των *Lex* και *Yacc*.

Ιδιαίτερα, η χρήση των *Lex* και *Yacc* διευκολύνουν πολύ την όποια πιθανή αλλαγή στη γλώσσα περιγραφής πειραμάτων, αλλά και τις οποιεσδήποτε μελλοντικές επεκτάσεις.

Εκτός όμως από τη συμμόρφωση με το πρότυπο *ANSI−C* για να μπορούν να μεταφραστούν τα προγράμματα και σε άλλες υπολογιστικές πλατφόρμες χρειάστηκαν κάποιες επιπλέον τροποποιήσεις και κυρίως υπό συνθήκη μεταφραστικοί κανόνες (ifdef statements). Τέλος, απαραίτητη ήτανε η χρήση πρωτοτύπων των συναρτήσεων (function prototypes).

Κεφάλαιο 4

Πειράματα

Σε όλες τις περιπτώσεις ομαδοποίησης φόρτου εργασίας, είτε οι μονάδες που ομαδοποιούνται είναι δοσοληψίες, είτε είναι προγράμματα, είτε είναι διαδικασίες, το κύριο πρόβλημα **δεν** είναι το να γίνει μια ομαδοποίηση (αφού πάντα προκύπτει μια ομαδοποίηση των αρχικών στοιχείων), αλλά ο έλεγχος της εγκυρότητας ή τουλάχιστον η αντικειμενική αξιολόγηση των αποτελεσμάτων.

Στην εργασία αυτή ακολουθήθηκαν δύο κατευθύνσεις όσον αφορά την αξιολόγηση των αποτελεσμάτων. Αφενός δημιουργήθηκε ένα εργαλείο (*TSG*) για την παρασκευή συνθετικών ιχνών συγκεκριμένων προδιαγραφών με σκοπό την επαλήθευση των αποτελεσμάτων και αφετέρου υλοποιήθηκε μια μεθοδολογία αξιολόγησης των αποτελεσμάτων του *CLUE* για πραγματικά δεδομένα.

4.1 Συνθετική παραγωγή ιχνών

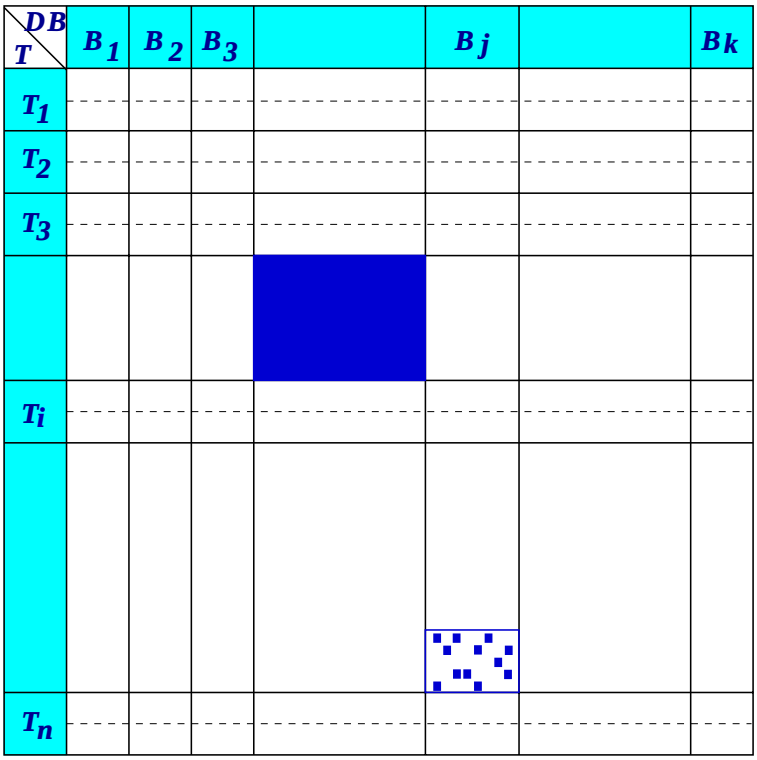
Το *TSG* (από το Test Suite Generator) είναι ένα εργαλείο, το οποίο βάσει μιας απλής γλώσσας προδιαγραφών κατασκευάζει συνθετικά ίχνη, τα οποία χρησιμοποιούνται ως είσοδος στο *CLUE* .

Είναι πολύ σημαντικό γιατί αφενός τα ίχνη από πραγματικές εφαρμογές είναι εξαιρετικά πολύπλοκα, γεγονός που κάνει την αξιολόγηση των αποτελεσμάτων ιδιαίτερα δύσκολη, και αφετέρου τα ίχνη από πραγματικές εφαρμογές είναι συνήθως πολύ μεγάλα, οπότε οι χρόνοι εκτέλεσης είναι αρκετά μεγάλοι (κυρίως για τους άλλους αλγόριθμους και όχι για το *HALC*).

Το *TSG* έχει την ίδια "αντίληψη" με το *CLUE* για τον πίνακα προσπελάσεων που κατασκευάζει. Δηλαδή οι τριπλέτες αντιστοιχούν σε γραμμές και οι σελίδες της βάσης δεδομένων αντιστοιχούν σε στήλες. Χρησιμοποιεί ένα αρχείο περιγραφής συνθετικών ιχνών ως κανόνα για το αρχείο ιχνών που πρόκειται να παράγει.

Τα βασικά στοιχεία που χρειάζεται να ξέρει το *TSG* (και συνήθως είναι τα πρώτα που

προσδιορίζονται στο αρχείο περιγραφής συνθετικών ιχνών) είναι το πλήθος των δοσοληψιών, τερματικών, χρηστών και σελίδων βάσης δεδομένων που θα περιέχει το τελικό ίχνος.



Σχήμα 4.1: Πίνακας Προσπελάσεων στο *TSG*

Κατόπιν, προσδιορίζονται ένα–ένα, παραλληλόγραμμα στον πίνακα προσπελάσεων, τα οποία αντιστοιχούν σε περιοχές που θα πρέπει το *TSG* να "γεμίσει" με προσπελάσεις (βλέπε σχήμα 4.1). Η περιγραφή του παραλληλογράμμου είναι ιδιαίτερα απλή, αφού αρκεί να προσδιορίσει κανείς τον ελάχιστο αριθμό τριπλέτας και το μέγιστο (οπότε προσδιορίζει πλήρως τις γραμμές που θα έχει η περιοχή) και ομοίως τον ελάχιστο και τον μέγιστο αριθμό σελίδων της βάσης.

Τέλος, ιδιαίτερο ρόλο έχουν κάποιες παράμετροι που μπορούν να αλλάξουν κάπως τη συμπεριφορά του *TSG* .

Η πρώτη παράμετρος έχει να κάνει με τον τρόπο που γίνεται η επιλογή ενός αριθμού προσπελάσεων με τον οποίο θα ανατεθεί ένα σημείο στον πίνακα προσπελάσεων. Οι δυνατές επιλογές στην περίπτωση αυτή είναι δύο:

- Να είναι ο αριθμός *σταθερός* για όλο το παραλληλόγραμμο που πρόκειται να γεμίσει.
- Να είναι ο αριθμός *τυχαίος*, και να έχει προσδιοριστεί απλά το διάστημα μέσα στο οποίο

επιτρέπεται να πάρει τιμές.

Η δεύτερη παράμετρος έχει να κάνει με το τρόπο που αντιμετωπίζονται πολλές τιμές για προσπελάσεις στο ίδιο σημείο του πίνακα προσπελάσεων:

- Υπάρχει η περίπτωση κάθε νέα τιμή να αθροίζεται με την τιμή που προϋπήρχε.
- Υπάρχει η περίπτωση κάθε νέα τιμή να εκχωρείται απευθείας στο συγκεκριμένο σημείο του πίνακα, αναιρώντας τις προηγούμενες,

Η τρίτη παράμετρος τέλος, κανονίζει την *πυκνότητα* του παραγόμενου ίχνους. Με τον όρο πυκνότητα εννοείται το πλήθος των μη μηδενικών προσπελάσεων έναντι του συνολικού πλήθους σε μία παραλληλόγραμμη περιοχή. Για παράδειγμα σε μια παραλληλόγραμμη περιοχή διαστάσεων 20×10 και με πυκνότητα 10%, μόλις τα $200 \times 10\% = 20$ στοιχεία θα έχουν μη μηδενική τιμή, έναντι των 2000 στοιχείων κανονικά.

Η παράμετρος της πυκνότητας έχει εισαχθεί για να μπορέσουν να κατασκευαστούν συνθετικά ίχνη όσο το δυνατόν πιο κοντά στα πραγματικά, όπου οι προσπελάσεις είναι εξίσου "αραιές".

4.2 Σύγκριση & αξιολόγηση ομαδοποιήσεων

4.2.1 Σύγκριση ομαδοποιήσεων

Υπάρχουν δύο διαδεδομένοι τρόποι για να συγκρίνει κανείς δύο διαφορετικές ομαδοποιήσεις του ίδιου συνόλου δεδομένων ([And73]).

Η πρώτη μέθοδος βασίζεται στην κατασκευή ενός *πίνακα εγγύτητας* (contingency table). Ο πίνακας αυτός, τη γενική μορφή του οποίου βλέπουμε στο σχήμα 4.1, περιέχει στην κάθε θέση του το πλήθος των όμοιων στοιχείων μεταξύ των δύο κλάσεων. Με άλλα λόγια, το $n_{i,j}$ περιέχει το πλήθος των κοινών σημείων μεταξύ των ομάδων A_i και B_j.

Αφού κατασκευαστεί ο πίνακας εγγύτητας, υπολογίζεται ο έλεγχος του χ^2, όπου για τη γενική περίπτωση είναι:

$$\chi^2 = n_{..} \left[\sum_{i=1}^{p} \sum_{j=1}^{q} \frac{n_{ij}^2}{n_{i.} n_{.j}} - 1 \right]$$

το οποίο όμως αυξάνεται με την αύξηση του $n_{..}$, οπότε δεν έχει όρια. Μια μερική λύση στο πρόβλημα αυτό είναι η χρήση του μέσου τετραγώνου εγγύτητας, το οποίο ορίζεται ως:

$$\Phi^2 = \frac{\chi^2}{n_{..}}$$

Έστω για παράδειγμα δύο διαφορετικές διαμερίσεις ενός συνόλου 15 στοιχείων:

$$\{i_1, i_2, i_3, i_4, i_5, i_6, i_7, i_8, i_9, i_{10}, i_{11}, i_{12}, i_{13}, i_{14}, i_{15}\}$$

B / A	1	2	$\cdots$	q	σύνολα
1	n_{11}	n_{12}	$\cdots$	n_{1q}	$n_{1.}$
2	n_{21}	n_{22}	$\cdots$	n_{2q}	$n_{2.}$
$\vdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
p	n_{p1}	n_{p2}	$\cdots$	n_{pq}	$n_{p.}$
σύνολα	$n_{.1}$	$n_{.2}$	$\cdots$	$n_{.q}$	$n_{..}$

Πίνακας 4.1: Γενική μορφή πίνακα εγγύτητας

Ο πίνακας εγγύτητας για το παράδειγμα αυτό φαίνεται στο σχήμα 4.2, και έχουμε

$$\chi^2 = 15\left(\frac{3^2}{5\times 3}+\frac{2^2}{5\times 2}+\frac{1^2}{4\times 4}+\frac{2^2}{4\times 2}+\frac{1^2}{4\times 4}+\frac{3^2}{3\times 4}+\frac{3^2}{3\times 4}-1\right)=\frac{255}{8}=31.875$$

	$\{i_1,i_2,i_3,i_4,i_5\}$	$\{i_6,i_{10},i_{11},i_{12}\}$	$\{i_7,i_8,i_9\}$	$\{i_{13},i_{14},i_{15}\}$	σύνολα
$\{i_1,i_2,i_3\}$	3	0	0	0	3
$\{i_4,i_5\}$	2	0	0	0	2
$\{i_6,i_7,i_8,i_9\}$	0	1	3	0	4
$\{i_{10},i_{11}\}$	0	2	0	0	2
$\{i_{12},i_{13},i_{14},i_{15}\}$	0	1	0	3	4
σύνολα	5	4	3	3	15

Πίνακας 4.2: Παράδειγμα πίνακα εγγύτητας

Η δεύτερη μέθοδος βασίζεται στο πως "μεταχειρίζονται" οι δύο υπό σύγκριση διαμερίσεις ένα ζευγάρι στοιχείων. Ίση μεταχείριση έχουμε όταν για παράδειγμα και τα δύο στοιχεία του ζευγαριού βρίσκονται στην ίδια ομάδα και στις δύο διαμερίσεις, ή όταν βρίσκονται σε διαφορετικές ομάδες και στις δύο διαμερίσεις. Διαφορετική μεταχείριση έχουμε όταν τα δύο στοιχεία του ζευγαριού βρίσκονται μαζί στη μία διαμέριση και ξεχωριστά στην άλλη διαμέριση. Το μέτρο ομοιότητας των δύο διαμερίσεων ισούται με το πλήθος των ζευγαριών που έχουν όμοια αντιμετώπιση διά του πλήθους των δυνατών συνδυασμών ζευγαριών στοιχείων.

Έστω ένα παράδειγμα συνόλου 6 στοιχείων $\{i_1,i_2,i_3,i_4,i_5,i_6\}$, για το οποίο υπάρχουν δύο διαφορετικές διαμερίσεις:

- $A=\{\{i_1,i_2\},\{i_3,i_4\},\{i_5,i_6\}\}$
- $B=\{\{i_1,i_3\},\{i_4,i_5\},\{i_2,i_6\}\}$

Τα ζευγάρια στοιχείων που έχουν ανόμοια μεταχείριση στις δύο διαμερίσεις A και B είναι τα : (i_1, i_2), (i_2, i_3), (i_2, i_6), (i_4, i_6), και (i_6, i_6). 'Ετσι, το μέτρο της ομοιότητας είναι ίσο με: $\frac{5}{15} = \frac{1}{3} = 66.6\%$.

4.2.2 Αξιολόγηση ομαδοποιήσεων

Ο πιο σίγουρος τρόπος, για να αξιολογήσει κανείς τα αποτελέσματα της ομαδοποίησης δοσοληψιών με βάση τη συγγένεια σε δεδομένα, είναι να τα τροφοδοτήσει σε ένα *πραγματικό σύστημα* κατανεμημένης επεξεργασίας δοσοληψιών και να παρατηρήσει την βελτίωση στις επιδόσεις του συστήματος. Συγκεκριμένα, τα αποτελέσματα της ομαδοποίησης θα πρέπει να χρησιμοποιηθούν αφενός στη φάση της διαμέρισης της βάσης δεδομένων στους διάφορους κόμβους του συστήματος και αφετέρου στους αλγόριθμους δρομολόγησης δοσοληψιών.

Ελλείψει πραγματικού συστήματος, η χρήση προσομοιωτή είναι εξίσου καλή ιδέα. Όσο πιο λεπτομερής η προσομοίωση, τόσο πιο κοντά στο πραγματικό σύστημα θα βρίσκεται και άρα τόσο πιο αξιόπιστα θα είναι τα συμπεράσματα που θα βγάλει κανείς για την ποιότητα των αποτελεσμάτων της ομαδοποίησης. Μία από τις μελλοντικές κατευθύνσεις αυτής της εργασίας είναι και η προσπάθεια χρήσης του προσομοιωτή TPSIM ([MN95a]), ο οποίος είναι ένας προσομοιωτής για κατανεμημένα σύστηματα επεξεργασίας δοσοληψιών.

Η τρίτη προσέγγιση στο θέμα της αξιολόγησης των αποτελεσμάτων της ομαδοποίησης είναι μέσω μιας αντικειμενικής συνάρτησης κόστους, όπως προτάθηκε στο [CMVN92]. Η συνάρτηση αυτή ονομάζεται *Partition Evaluator* και είναι η εξής:

$$PE = E_M^2 + E_R^2$$

όπου E_M^2 είναι το κόστος προσπέλασης άσχετων τοπικών ιδιοτήτων, και
E_R^2 είναι το κόστος προσπέλασης σχετικών απομακρυσμένων ιδιοτήτων.

Δηλαδή στο PE αντανακλάται τόσο η ποσότητα των απομακρυσμένων προσπελάσεων που θα έπρεπε να είναι τοπικές για να βελτιωθεί η επίδοση του συστήματος, όσο και η ποσότητα των δεδομένων που είναι τοπικά, ενώ θα μπορούσαν να είναι απομακρυσμένα.

Τα προβλήματα που παρουσιάζει αυτή η προσέγγιση είναι κυρίως η ανάμειξη πολλών διαφορετικών παραμέτρων σε μία μόνο συνάρτηση κόστους (π.χ. τον τρόπο με τον οποίο μοιράζονται τα δεδομένα στους διάφορους κόμβους, τον τρόπο με τον οποίο μοιράζονται οι δοσοληψίες στους διάφορους κόμβους) κάτι το οποίο σημαίνει ότι μπορεί τα συμπεράσματα να μην ισχύουν για τη γενική περίπτωση.

Τέλος, ένας κλασσικός έλεγχος για την ποιότητα των παραγόμενων ομαδοποιήσεων είναι ο *έλεγχος του* χ^2[JD88]). Βασίζεται στο γεγονός ότι η ελαχιστοποίηση του τετραγωνικού

λάθους μεταξύ των ομάδων (δηλαδή ο υπολογισμός του E_K^2, βλέπε σελίδα 30), συνεπάγεται και μεγιστοποίηση της απόκλισης μεταξύ των ομάδων.

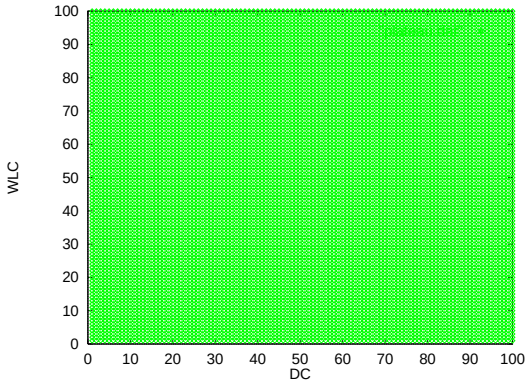
Το βασικότερο πλεονέκτημα της μεθόδου αυτής είναι η ανεξαρτησία της από τις παραμέτρους του συγκεκριμένου προβλήματος, αφού είναι σε θέση να δώσει ένα χαρακτηρισμό της ποιότητας της ομαδοποίησης ανεξάρτητα από τη δρομολόγηση και τον αριθμό των δοσοληψιών, αλλά και από την κατανομή των δεδομένων στους κόμβους του συστήματος. Ο *έλεγχος του* χ^2 ήταν αυτός που προτιμήθηκε και υλοποιήθηκε στη παρούσα εργασία.

4.3 Πειράματα

4.3.1 Συνθετικά κατασκευασμένα ίχνη

Οι βασικές κατηγορίες συνθετικών ιχνών

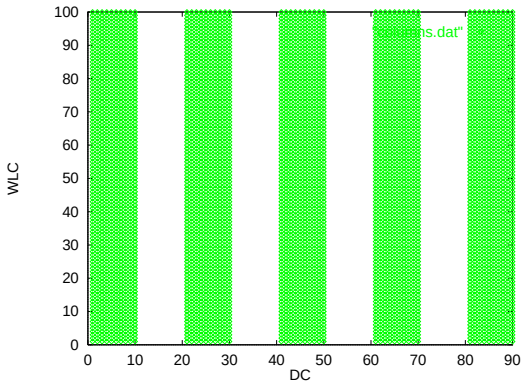
Η στρατηγική που ακολουθήθηκε για τη σχεδίαση συνθετικών ιχνών ήταν η δημιουργία ιχνών που μπορούσαν να αναπαραστήσουν όσο το δυνατόν περισσότερες περιπτώσεις που ενδεχομένως να εμφανιστούν σε ένα πραγματικό ίχνος. Τα συνθετικά ίχνη που κατασκευάστηκαν ήταν τεσσάρων διαφορετικών τύπων.



Σχήμα 4.2: Συνθετικά ίχνη τύπου plateau

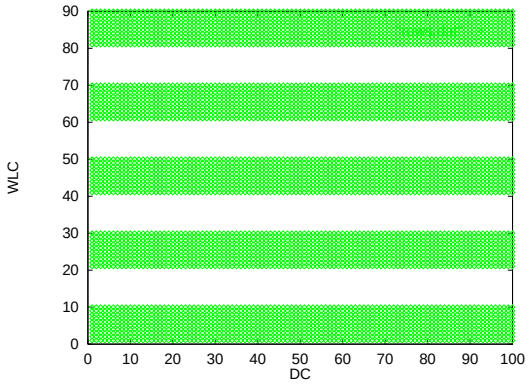
Τύπου *plateau*: (βλέπε σχήμα 4.2)
Αποτελούν την απλούστερη κατηγορία ιχνών αφού αντιστοιχούν στην περίπτωση όλες οι τριπλέτες να προσπελαύνουν όλες τις σελίδες της βάσης δεδομένων. Θα πρέπει η ομαδοποίησή τους να καταλήγει πάντα σε μία κλάση που περιέχει όλες τις τριπλέτες.

Τύπου *columns*: (βλέπε σχήμα 4.3)
Αρκετά απλή περίπτωση κατηγορίας ιχνών, όπου όλες οι τριπλέτες προσπελαύνουν ένα



Σχήμα 4.3: Συνθετικά ίχνη τύπου columns

σταθερό υποσύνολο των σελίδων της βάσης δεδομένων. Θα πρέπει η ομαδοποίησή τους να καταλήγει πάντα σε μία κλάση που περιέχει όλες τις τριπλέτες.



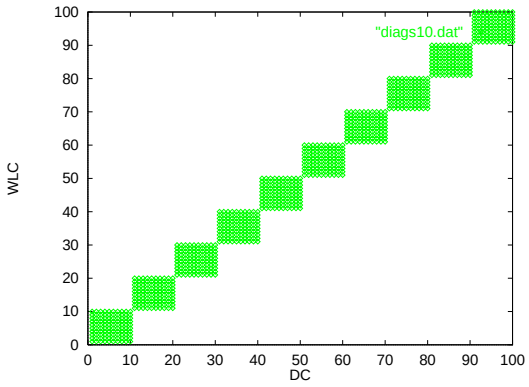
Σχήμα 4.4: Συνθετικά ίχνη τύπου rows

Τύπου *rows*: (βλέπε σχήμα 4.4)

Επίσης αρκετά απλή περίπτωση κατηγορίας ιχνών, όπου κάποιες τριπλέτες προσπελαύνουν όλες τις σελίδες της βάσης δεδομένων. Θα πρέπει η ομαδοποίησή τους να καταλήγει πάντα σε μία κλάση που περιέχει όλες τις τριπλέτες.

Τύπου *diagonalsquares*: (βλέπε σχήμα 4.5)

Η περισσότερο ενδιαφέρουσα περίπτωση κατηγορίας ιχνών, όπου κάποιες τριπλέτες προσπελαύνουν κάποιες σελίδες της βάσης δεδομένων. Οι προσπελάσεις των τριπλετών σε σελίδες της βάσης δεδομένων σχηματίζουν μικρά τετράγωνα πάνω στη διαγώνιο του πίνακα προσπελάσεων. Θα πρέπει η ομαδοποίησή τους να καταλήγει σε τόσες κλάσεις όσα και τα



Σχήμα 4.5: Συνθετικά ίχνη τύπου diagonalsquares

διακριτά τετράγωνα στην διαγώνιο.

Επιπλέον, κατά την κατασκευή των συνθετικών ιχνών, η ανάθεση τιμών στις διάφορες περιοχές του πίνακα προσπελάσεων μπορούσε να είναι *σταθερή*, όπου όλα τα σημεία του πίνακα έπαιρναν την ίδια προκαθορισμένη τιμή, ή *τυχαία*, όπου οι δυνατές τιμές διαλέγονταν με τυχαίο τρόπο από ένα προκαθορισμένο διάστημα.

Πειράματα με τον *HALC*

Για να δοκιμαστεί ο *HALC* εκτελέστηκαν πολλά πειράματα με συνθετικά κατασκευασμένα ίχνη. Χρησιμοποιήθηκαν κυρίως ίχνη του τύπου diagonalsquares γιατί ήταν και τα πιο περίπλοκα. Τα πειράματα που έγιναν αποσκοπούσαν στη μελέτη της ευαισθησίας του *HALC* στις διάφορες παραμέτρους του.

Αποδείχτηκε τελικά ότι τρεις ήτανε οι παράμετροι που επηρεάζαν περισσότερο την ποιότητα των αποτελεσμάτων:

- Το όριο στο κριτήριο ποσότητας, δηλαδή το μέγιστο ποσοστό συμπίεσης των τριπλετών σε ένα βήμα του αλγορίθμου. Τυπικές τιμές του ποσοστού αυτού είναι γύρω στο 10% αλλά και τιμές κοντά στο 20% χρησιμοποιήθηκαν στις περιπτώσεις παραδειγμάτων με σταθερή τιμή προσπελάσεων.

- Το όριο στο κριτήριο ποιότητας (k factor), δηλαδή το ελάχιστο αναμενόμενο ποσοστό συμπίεσης του αριθμού των τριπλετών σε σχέση με το αρχικό ποσοστό συμπίεσης σε ένα βήμα του αλγορίθμου. Τυπικές τιμές του ποσοστού αυτού ήταν γύρω στο 10% – 15%.

- Ο τρόπος επίλυσης του προβλήματος ping–pong, δηλαδή του υπολογισμού της από–στασης ομαδοποίησης στην περίπτωση που αποφασιστεί από τον αλγόριθμο ξαφνική

αλλαγή στην πορεία μεταβολής της. Η καλύτερη προσέγγιση αποδείχτηκε τελικά η χρήση της τελευταίας τιμής (η συντηρητική προσέγγιση δηλαδή).

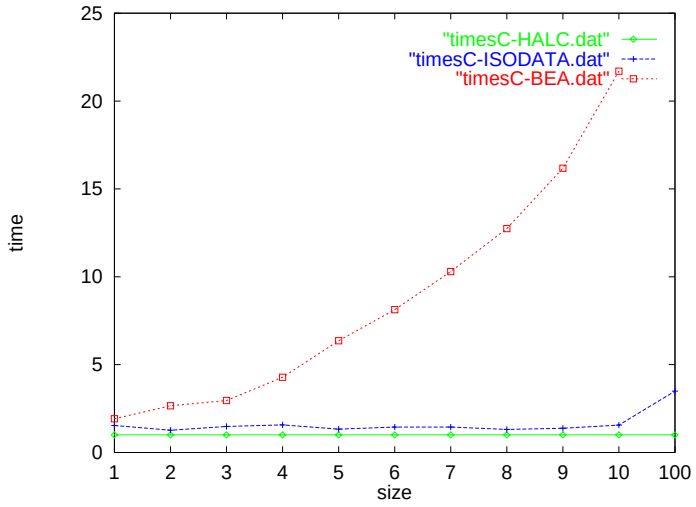
Συγκριτικές μετρήσεις

Για να αξιολογηθεί η επίδοση του *HALC* σε σχέση με τους άλλους δύο αλγορίθμους χρησιμοποιήθηκε η κατηγορία συνθετικών ιχνών diagonalsquares. Κατασκευάστηκαν συνθετικά ίχνη με διαφορετικά μεγέθη $1, 2, \ldots, 10$ και 100 κλάσεων, με σταθερό αλλά και με τυχαίο πλήθος προσπελάσεων και δόθηκαν ως είσοδος στους 3 αλγορίθμους.

Στους πίνακες που ακολουθούν οι χρόνοι t είναι σε δευτερόλεπτα, ενώ οι χρόνοι t_R είναι χρόνοι σχετικοί ως πρός το χρόνο εκτέλεσης του *HALC* (=t_{HALC}), δηλαδή $t_R = \frac{t}{t_{HALC}}$. Τέλος στη στήλη β αναγράφεται το σύνολο των βημάτων του κάθε αλγορίθμου.

Για τα πειράματα αυτά, ο αλγόριθμος ISODATA χρησιμοποιούσε μια σταθερή αρχική ανάθεση σε ομάδες, Όταν δοκιμάστηκε με τυχαία αρχική ανάθεση, τα αποτελέσματα ήταν αποθαρρυντικά, αφού υπήρχαν περιπτώσεις όπου δεν έβρισκε τη σωστή ομαδοποίηση.

Στον πίνακα 4.3 φαίνονται αναλυτικά οι χρόνοι εκτέλεσης για την περίπτωση των σταυερών τιμών προσπελάσεων και στο σχήμα 4.6 υπάρχει ένα διάγραμμα με τους σχετικούς χρόνους εκτέλεσης των ISODATA και BEA ως προς τον $HALC$.

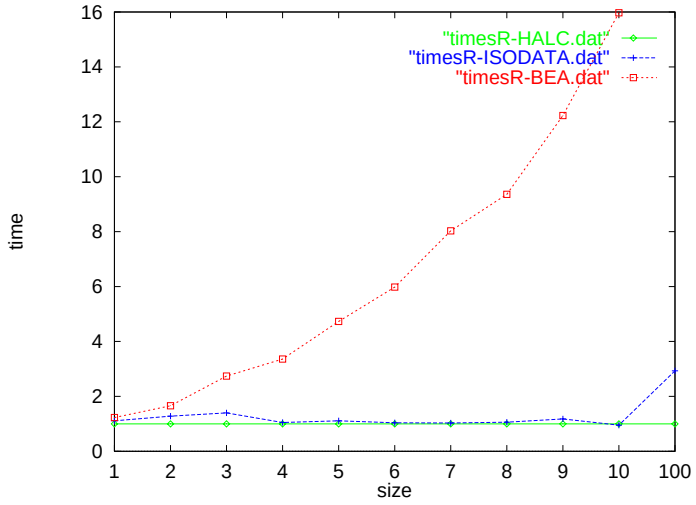


Σχήμα 4.6: Χρόνοι εκτέλεσης για diagonalsquares με σταθερές τιμές προσπελάσεων

Μέγεθος	κλ	E_K^2	HALC		ISODATA			BEA	
			t	β	t	t_R	β	t	t_R
10×10	1	0.0	0.13	4	0.19	1.54	1	0.25	1.92
20×20	2	0.0	0.18	8	0.23	1.27	1	0.48	2.66
30×30	3	0.0	0.25	9	0.37	1.48	1	0.74	2.96
40×40	4	0.0	0.28	9	0.44	1.57	1	1.20	4.28
50×50	5	0.0	0.33	9	0.44	1.33	1	2.10	6.36
60×60	6	0.0	0.37	16	0.54	1.45	1	3.01	8.13
70×70	7	0.0	0.42	22	0.61	1.45	1	4.33	10.30
80×80	8	0.0	0.47	15	0.62	1.31	1	5.99	12.74
90×90	9	0.0	0.50	10	0.69	1.38	1	8.09	16.18
100×100	10	0.0	0.51	15	0.80	1.56	1	11.07	21.70
1000×1000	100	0.0	5.03	19	17.58	3.49	1	7746.4	1540.03

Πίνακας 4.3: Συγκριτικά αποτελέσματα εκτέλεσης των 3 αλγορίθμων για διαγώνια ίχνη με σταθερό πλήθος προσπελάσεων

Στον πίνακα 4.4 φαίνονται αναλυτικά οι χρόνοι εκτέλεσης για την περίπτωση των σταθερών τιμών προσπελάσεων και στο σχήμα 4.7 υπάρχει ένα διάγραμμα με τους σχετικούς χρόνους εκτέλεσης των ISODATA και BEA ως προς τον *HALC*.



Σχήμα 4.7: Χρόνοι εκτέλεσης για diagonalsquares με τυχαίες τιμές προσπελάσεων

Μέγεθος	κλ	E_K^2	HALC		ISODATA			BEA	
			t	β	t	t_R	β	t	t_R
10×10	1	6.79	0.17	5	0.19	1.11	1	0.21	1.23
20×20	2	14.83	0.21	14	0.27	1.28	1	0.35	1.66
30×30	3	23.54	0.27	24	0.38	1.40	1	0.74	2.74
40×40	4	30.65	0.36	29	0.38	1.05	1	1.21	3.36
50×50	5	38.42	0.42	33	0.47	1.11	1	1.99	4.73
60×60	6	46.50	0.50	39	0.52	1.04	1	2.99	5.98
70×70	7	54.43	0.55	40	0.57	1.03	1	4.42	8.03
80×80	8	62.84	0.63	50	0.67	1.06	1	5.90	9.36
90×90	9	70.89	0.65	45	0.77	1.18	1	7.95	12.23
100×100	10	78.99	0.70	37	0.67	0.95	1	11.18	15.97
1000×1000	100	800.34	7.49	68	22.02	2.93	1	8258.0	1102.53

Πίνακας 4.4: Συγκριτικά αποτελέσματα εκτέλεσης των 3 αλγορίθμων για διαγώνια ίχνη με τυχαίο πλήθος προσπελάσεων

Σε όλα τα πειράματα που έγιναν φάνηκε η σχετική υπεροχή του *HALC* έναντι των άλλων δύο αλγορίθμων, αφού ο *HALC* για την ίδια ποιότητα αποτελεσμάτων είχε πάντα το μικρότερο χρόνο εκτέλεσης.

4.3.2 Ίχνη από πραγματικές εφαρμογές

Τα δύο ίχνη από πραγματικές εφαρμογές που χρησιμοποιήθηκαν, πάρθηκαν με ιχνοληψία μιας πραγματικής εφαρμογής βάσης δεδομένων σε ένα παραδοσιακό σύστημα επεξεργασίας δοσοληψιών (BS2000 mainframe, της SNI).

Τα ίχνη που πάρθηκαν χρειάστηκε να μετατραπούν στη μορφή που απαιτεί ως είσοδο το *CLUE* μέσω ενός φίλτρου. Η πιο σημαντική λειτουργία του φίλτρου αυτού ήταν ο συγκερασμός των προσπελάσεων στη βάση δεδομένων όλων των συγκεκριμένων περιπτώσεων (instances) των δοσοληψιών, ώστε να σχηματιστεί ένας συγκεντρωτικός πίνακας προσπελάσεων.

Αξίζει να σημειωθεί ότι και τα δύο ίχνη από πραγματικές εφαρμογές έχουν εξαιρετικά χαμηλή *πυκνότητα προσπελάσεων*, όπου ως πυκνότητα προσπελάσεων ορίζουμε:

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

DOA

Το ίχνος DOA έχει τα ακόλουθα χαρακτηριστικά:

- **474** userIDs, **474** terminalIDs, **103** transactionIDs
- **1984** triplets, **41003** data items
- **180017** μη μηδενικές προσπελάσεις
- πυκνότητα προσπελάσεων: **0,22123%**

Στον πίνακα 4.5 φαίνονται τα αποτελέσματα των 3 αλγορίθμων στο ίχνος DOA, καθώς και η αρχική τιμή του E_K^2, όπου ο αλγόριθμος ISODATA δοκιμάστηκε τόσο με τυχαία αρχική ανάθεση τριπλετών σε ομάδες (ISODATA−R), όσο και με σταθερή αρχική ανάθεση τριπλετών σε ομάδες (ISODATA−C).

αλγόριθμος	βήματα	χρόνος εκτέλεσης	$\frac{t}{t_{HALC}}$	κλάσεις	E_K^2
αρχικά	–	–	–	1984	408.42
HALC	20	1 min 41 sec	1	92	3.56
ISODATA−R	12	159 min 48 sec	94.56	100	34.32
ISODATA−C	13	79 min 55 sec	47.29	100	129.93
BEA	–	$\geq$ 4 days	$\geq$ 3421.78	100	

Πίνακας 4.5: Συγκριτικά αποτελέσματα εκτέλεσης των 3 αλγορίθμων στο ίχνος DOA

PULS

Το ίχνος PULS έχει τα ακόλουθα χαρακτηριστικά:

- **103** userIDs, **110** terminalIDs, **53** transactionIDs
- **280** triplets, **66846** data items
- **178749** μη μηδενικές προσπελάσεις
- πυκνότητα προσπελάσεων: **0,95501%**

Στον πίνακα 4.6 φαίνονται τα αποτελέσματα των 3 αλγορίθμων στο ίχνος PULS, καθώς και η αρχική τιμή του E_K^2, όπου ο αλγόριθμος ISODATA δοκιμάστηκε τόσο με τυχαία αρχική ανάθεση τριπλετών σε ομάδες (ISODATA−R), όσο και με σταθερή αρχική ανάθεση τριπλετών σε ομάδες (ISODATA−C).

αλγόριθμος	βήματα	χρόνος εκτέλεσης	$\frac{t}{t_{HALC}}$	κλάσεις	E_K^2
αρχικά	–	–	–	280	60.19
HALC	20	1 min 25 sec	1	28	0.14
ISODATA−R	10	14 min 5 sec	9.91	30	0.29
ISODATA−C	3	13 min 14 sec	9.32	30	0.96
BEA	–	$\geq$ 3 days	$\geq$ 2566,34	30	

Πίνακας 4.6: Συγκριτικά αποτελέσματα εκτέλεσης των 3 αλγορίθμων στο ίχνος PULS

Συμπεράσματα

Ο αλγόριθμος *HALC* παρουσιάζει πολύ μικρό χρόνο εκτέλεσης (γύρω στο 1,5 min και για τα δύο ίχνη) και πάρα πολύ καλά αποτελέσματα. Ο αλγόριθμος ISODATA παρουσιάζει πολύ μεγαλύτερους χρόνους εκτέλεσης και χειρότερη ποιότητα ομαδοποίησης. Ο αλγόριθμος BEA αναμένεται να έχει την καλύτερη ποιότητα ομαδοποίησης, αλλά ο ανυπόφορα μεγάλος χρόνος εκτέλεσης (λίγες ημέρες) τον καθιστά πρακτικα άχρηστο.

Κεφάλαιο 5

Συμπεράσματα και μελλοντικές κατευθύνσεις

5.1 Συμπεράσματα

Η εργασία αυτή αντιμετώπισε το πρόβλημα του χαρακτηρισμού φόρτου εργασίας με βάση τις προσπελάσεις που κάνουν οι διάφορες δοσοληψίες σε σελίδες της βάσης δεδομένων.

Παρουσιάστηκε ένας απλός και γρήγορος ευρηματικός αλγόριθμος και επιπλέον υλοποιήθηκαν άλλοι δύο γνωστοί για να γίνουν συγκρίσεις. Επίσης κατασκευάστηκε ένα εργαλείο για παρασκευή συνθετικών ιχνών, το οποίο και χρησιμοποιήθηκε για την επαλήθευση των αποτελεσμάτων του *CLUE*. Τέλος, χρησιμοποιήθηκαν ίχνη από πραγματικές εφαρμογές και ο έλεγχος του χ^2 για να συγκριθεί ο *HALC* με τους αλγορίθμους ISODATA και BEA.

Με τις μετρήσεις που έγιναν φάνηκε η υπεροχή του *HALC*, όσον αφορά την ταχύτητα της εκτέλεσης αλλά και την ποιότητα των παραγόμενων ομαδοποιήσεων, αφού σε σύγκριση με τον ISODATA αλγόριθμο είναι πολύ πιο γρήγορος και βγάζει καλύτερα αποτελέσματα, ενώ σε σύγκριση με τον BEA η ποιότητα των αποτελεσμάτων είναι εφάμιλλη, αλλά οι χρόνοι εκτέλεσης του BEA είναι μέχρι και 2000 φορές μεγαλύτεροι από ότι του *HALC*.

Συνολικά μπορούμε να πούμε ότι η ομαδοποίηση δοσοληψιών με βάση τη συγγένεια σε δεδομένα είναι ιδιαίτερα σημαντική, ενώ ειδικότερα ο αλγόριθμος *HALC* εμφανίζεται ως πολύ αξιόλογη λύση στο πρόβλημα.

Στις μελλοντικές κατευθύνσεις αυτής της δουλειάς διακρίνουμε δύο διαφορετικές προσεγγίσεις, όσον αφορά τη σχέση τους με την δουλειά που έχει ήδη γίνει. Η πρώτη κατηγορία κατευθύνσεων, οι επεκτάσεις, ουσιαστικά αποτελεί βελτιώσεις της υπάρχουσας δουλειάς με μικρές ενδεχομένως αλλαγές στο *CLUE*. Αντίθετα, η δεύτερη κατηγορία, οι μελλοντικές κατευθύνσεις, περιέχει θέματα τα οποία μπορούν να αποτελέσουν συνέχιση της παρούσας

δουλειάς, αλλά έχουν σημαντικά περισσότερες δυσκολίες και διαφορές από το *CLUE*.

5.2 Επεκτάσεις του *CLUE*

5.2.1 Αξιολόγηση

Από τις πιο σημαντικές επεκτάσεις αυτής της δουλειάς αποτελεί η σύνδεση του *CLUE* με τον προσομοιωτή TPSIM ([MN95a]), ούτως ώστε να μπορέσει να γίνει μια λεπτομερέστερη αξιολόγηση των αποτελεσμάτων των τριών αλγόριθμων σε πραγματικά ίχνη. Σε αυτή την κατεύθυνση κινείται και η αναζήτηση καινούριων αρχείων ιχνών από πραγματικές εφαρμογές.

Αρκετά λεπτομερής αξιολόγηση των πειραμάτων μπορεί να γίνει και μέσω περισσότερο πολύπλοκων συνθετικών ιχνών. Για να μπορέσει όμως να γίνει κάτι τέτοιο διεξοδικά θα χρειαστούν κάποιες επεκτάσεις στον *TSG* ώστε να μπορέσει να κατασκευάζει ίχνη που έχουν περιγραφεί με μια γλώσσα υψηλότερου επιπέδου.

5.2.2 Άλλοι αλγόριθμοι

Άλλη κατεύθυνση στις επεκτάσεις του *CLUE* είναι η υλοποίηση και άλλων αλγόριθμων ομαδοποίησης. Πιθανοί πρώτοι στόχοι είναι η υλοποίηση κάποιου ιεραρχικού αλγόριθμου, καθώς και η υλοποίηση κάποιου αλγόριθμου που χρησιμοποιεί ασαφή λογική (fuzzy logic).

5.2.3 Διάφορες βελτιώσεις

Επιπλέον, μπορεί να εξεταστεί κατά πόσο η χρήση κάποιων κανόνων και περιορισμών (π.χ. ο εντοπισμός read−only σελίδων μνήμης) είναι σε θέση να βοηθήσει τον αλγόριθμο *HALC* να βελτιώσει τα αποτελέσματά του.

Επίσης, μπορεί να εξεταστεί η πιθανότητα κάποιας επιπρόσθετης επεξεργασίας μετά την εκτέλεση του αλγορίθμου, ώστε να αποφευχθούν οι ακρίτες (ομαδοποιώντας τους για παράδειγμα με την κοντινότερη ομάδα).

Τέλος, θα πρέπει να εξεταστούν αλγόριθμοι για να γίνει η ομαδοποίηση των σελίδων της βάσης δεδομένων, αφού έχει ολοκληρωθεί η ομαδοποίηση των δοσολήψιών. Κάτι τέτοιο θα βοηθήσει στη διαμέριση της βάσης δεδομένων στους κόμβους του συστήματος κατανεμημένης επεξεργασίας δοσοληψιών.

5.3 Μελλοντικές κατευθύνσεις

5.3.1 Μεγάλος όγκος δεδομένων

Ένα από τα σημαντικότερα προβλήματα που αντιμετωπίστηκαν στην εργασία αυτή ήταν ο μεγάλος όγκος των δεδομένων εισόδου. Συγκεκριμένα η ύπαρξη τόσο μεγάλου πλήθους σελίδων της βάσης δεδομένων σε σχέση με το πλήθος των τριπλετών δίνει την εντύπωση της "επιπλέον" πληροφορίας και άρα την ανάγκη για συμπίεσή της.

Στο *CLUE* ακολουθήθηκε η πολιτική της υλοποίησης των δομών δεδομένων με τέτοιο τρόπο ώστε να εκμεταλλεύονται τη μικρή σχετική πυκνότητα των πραγματικών ιχνών και έτσι να μην "υποφέρει" σημαντικά από το μεγάλο όγκο δεδομένων.

Άλλες κατευθύνσεις στην προσπάθεια για τη λύση αυτού του προβλήματος είναι η ομαδοποίηση των σελιδών της βάσης (των στηλών δηλαδή του πίνακα προσπελάσεων) μέσω της χρήσης κάποιων μεθόδων ανάλυσης πρωταρχικών τμημάτων (principal components analysis), ώστε να μειωθεί σημαντικά το πλήθος τους.

Εναλλακτικά, ο μεγάλος όγκος των δεδομένων μπορεί να ελαττωθεί μέσω δειγματοληψίας, χωρίς να υπάρχει σημαντική διαφορά στην ποιότητα της τελικής ομαδοποίησης ([Art78]).

5.3.2 On the fly clustering

Μια άλλη πολύ ενδιαφέρουσα μελλοντική κατεύθυνση της εργασίας αυτής είναι η *ενεργή ομαδοποίηση* (on the fly clustering), όπου οι τριπλέτες ομαδοποιούνται καθώς το σύστημα είναι σε λειτουργία, σε αντίθεση με το *CLUE* όπου η ομαδοποίηση γίνεται off–line. Η ιδέα αυτή μπορεί να φανεί ιδιαίτερα ελκυστική σε περιπτώσεις συστημάτων κατανεμημένης επεξεργασίας δοσοληψιών, όπου ο φόρτος εργασίας αλλάζει πολύ συχνά.

Στην κατεύθυνση αυτή κινείται η προσπάθεια για υλοποίηση *γενετικών αλγορίθμων* ομαδοποίησης καθώς και αλγορίθμων που βασίζονται σε *νευρωνικά δίκτυα*. Τέλος, λόγω των απαιτήσεων για μικρούς χρόνους εκτέλεσης των αλγορίθμων ομαδοποίησης η *παραλληλοποίηση* των αλγορίθμων φαίνεται να αποτελεί μια πιθανή λύση.

5.3.3 Workflow

Ενδιαφέρον επίσης παρουσιάζει η περίπτωση των transactional workflows, όπου κάποιες δοσοληψίες εξαρτώνται από τα αποτελέσματα των προηγούμενων δοσοληψιών, όπως για παράδειγμα οι δοσοληψίες για κρατήσεις ταξιδίων, όπου πρέπει π.χ. να γίνει κράτηση για το αεροπλάνο, για το ξενοδοχείο, για την επιστροφή, κ.ο.κ. και θα πρέπει όλες αυτές οι ξεχωριστές μεταξύ τους δοσοληψίες να ξέρουν η μία για τα αποτελέσματα της άλλης.

Το ενδιαφέρον της περίπτωσης αυτής εστιάζεται στο κατά πόσο η σχετική πληροφορία θα πρέπει να ενσωματώνεται στο χαρακτηρισμό του φόρτου εργασίας ή θα πρέπει να αφήνεται στο τμήμα δρομολόγησης δοσοληψιών ([MN95b]).

5.3.4 Mixed workloads

Η τελευταία (και ίσως η δυσκολότερη) μελλοντική κατεύθυνση είναι αυτή του χαρακτηρισμού του φόρτου εργασίας σε ‘‘ανάμεικτες’’ εφαρμογές, όπου υπάρχουν δοσοληψίες με ανάγκη για μικρούς χρόνους απόκρισης και ταυτόχρονα ερωτήσεις (queries) στη βάση δεδομένων, όπου χρειάζονται αρκετό χρόνο για να εκτελεστούν.

Παράρτημα Α

TSG Reference Input File

```
T testsuiteName
# The header line 'T' has to be the first line. THIS is a comment.
# The header line string must not contain any white-space characters.
# The header line string's maximum length is 30 characters.
# There might be more than one test suite described in a testsuite generator
# input file. The testsuite name will be part of the header lines (type 'H')
# of the workload description files to be generated.
#
# Maximum line length is 80 characters.
# -------1---------2---------3---------4---------5---------6---------7---------8
#
#
#  Some general remarks:
#  =====================
#
#  The test suite generator reads this file until EOF is reached.
#  The records can be classified as follows:
#
#   o COMMENTS: Records beginning with a '#' or a 'C' are comment records.
#         - '#'-records are ignored.
#         - 'C'-records are so called "copy comments". This kind of records are
#           copied to the current output file immedieatedly after the
#           test suite generator has read it in.
#   o OPTIONS: All kinds of records except the record types describing
#         a region (formerly called "rectangle"). Region describing records
#         are described below.
#   o ACTIONS: Records beginning with record type (first character) 'A'.
#   o REGION-DESCRIPTION:  'w', 'd', 'r', and 'u'-records.
```

```
#
#   A key design decision is that options are valid until overwritten
#   by a new value or EOF. An action is performed as soon as the test
#   suite generator reads it ('C'-records may be interpreted as some
#   kind of special action).
#
#   Thus, the test suite gnerator input file is someting like a very
#   very simple "programming language" which enables flexible creation
#   test suites.
#
#   A test suite can consist of a whole bunch of (workload description)
#   files. The (unique) testsuite name is part of the 'H'-line of the
#   created files.
#
# ------------------------------------------------------------------------
#
#
C This is a copy comment. Lines beginning with a C are copied into the current
C output file as soon as processed (thereby turning to a '#'-comment').
#
#
t workloadName
# t-line is 30 characters long. The workload name / testcase name is
# concatenated with the testsuite name and forms the header line of the
# workload description files ('H'-Record).
#
f tsg_output_file
# The 'f' record speifies the name of the current output file. The output
# file is opened as soon this record is read in. There can only be one
# output file at a time.
#
# The following section describes the worklad class related information.
# The number of worcload class elements equals
# no_userids * no_term_ids * no_tran_ids.
# The number of members of each worcload class equals
# no_worload_class_elements div members_per_workloadclass.
number of user IDs = 3
# d no_user_ids = 3
number of terminal IDs = 4
# d no_term_ids = 4
```

```
number of transaction IDs = 5
# d no_tran_ids = 5
#
# So implicitely the number of workload class elements is 3*4*5=60
# in this example. The userid/tranid/termid-strings will be generated
# randomly from a given character set. The character set is
# "ABC...XYZ0123456789" (upper case letters).
# Perharps we should provide an option for the exact number of
# workload class elements, which will default to no_U * no_T * no_X
#
d no_workload_class_elements = 60
d avg_no_workload_class_members = 7
# The above line implies that (60 div 7) = 8 workload classes will
# be created. The first 7 workload classes will have 7 members each.
# Workload class 8 will have 8 + (60 - 7*8) = 8 + 4 = 12 members.
#
#
#
# The following describes the data classes (and related information):
d no_dataclass_elements = 100
d no_data_objects_per_bucket = 3
# the above line implies that data class element #1 ("bucket_1", see
# 'b'-record type on workload description files) covers the data objects
# 0,...,2 (min,...,max), the second covers 3,...,5 etc. There is no need
# to randomize this at the moment. In the current state, it is o.k. to
# assume a dataclass elment_i is a file_i (with one data object min==max==i).
d avg_no_members_per_dataclass = 10
# Each data class has ten members (buckets / data class elements). Implicitely
# 100/10 = 10 data clases have to be generated.
#
density=3%
density=0.03
# This option quantifies the sparsity of the matrix under production.
# The density is used as a cutoff with a combination of a random variable
# in such a way that only in the cases where prob() > density, a point
# in the reference matrix is given a value.
#
O fill_mode = assign/random
# This is a real option type record. the above line means that (until
# stated otherwise...) values are computed randomly and then assigned to
```

```
# the matrix entry. fill_mode can be specified as one of the following
# cases:
#            assign/random
#            assign/constant
#            add/random
#            add/constant
#
# "assign" means that values are directly assigned to the matrix entry.
# "add" means that values are added to the matrix entry instead of
# overwriting it. "random" means that values are computed uniformely
# distributed in some specified range (min-max). "constant" means
# that a constant value is used to be assigned/added to the matrix
# entry.
#
A clear matrix
# "clear matrix" sets all matrix entries to zero.
#
#
# The following couple of lines specify a region in the reference
# matrix and how to compute the number of references to be assigned/aded
# to all matrix entries within the region.
w 4 5       # row_range (workload class #4 to #5)
d 1 3       # col_range (data class #1 to #3)
r 5 100      # read accesses
u 3  50      # update acceses
# The above two lines will be interpreted depending on fill_mode settings.
# If "random" is set, the two integers in each line wil specify
# minimum and maximum of the interval of random numbers to be generated.
# If "constant" is set (max-min)/2 (rounded!) is used as a "constant" to
# be added/assigned to the matrix entriex. In both cases, each 'r' or 'u'
# -record needs to have to integers. This may be a bit inconvenient in case
# of cost, but makes quick changes (constant vs. random) easier.
#
# Many many regions (and options and comments) may follow..
fill mode = add/constant
w 1 3       # row_range (workload class #1 to #3)
d 2 4        # col_range (data class #2 to #4)
r 20 100     # read accesses
u 8  50      # update acceses
#
```

```
A close
# This action closes the current output file.
#
# ... now switch to another output file and do what must be done...
#
# ... you may also put another 'T'-record here (new testsuite!, multiple
# testsuites in one testsuite generator input file!). The testsuite
# generator takes care of memory and recycles most of the data
# structures used before.
#
# Note: The test suite generator runs and runs until it hits EOF.
#
# *************** This is the last line of the testsuite description file ******
```

Παράρτημα Β

CLUE Reference Input File

```
H CLOU_sample_input
# The header line has to be the first line. THIS is a comment.
# The header line string must not contain any white-space characters.
# The header line string's maximum length is 40 characters.
# Maximum line length is 80 characters.
# -------1---------2---------3---------4---------5---------6---------7---------8
#
# Note: The order of the folowing SECTIONS is mandatory. The order of the
# ----- records WITHIN the sections is immaterial.
#
#
# ************************************************************************
# * Global Section (record_type = G):                                    *
# ************************************************************************
# Record Format: "G <what_string> = <howmany_string> [comment]".
#
# The global section is the first section after the header line.
# Within this section,the order of the entries is immaterial.
# All global lines are mandatory.
#
G interval_length = 1000              Length of observation interval (seconds).
G no_user_ids = 3           Number of user-IDs (first component of WL class id).
G no_term_ids = 4      Number of terminal-IDs (second component of WL class id).
G no_tran_ids = 5    Number of transaction IDs (third component of WL class id).
G no_workload_class_elements = 12        Number of WL class elements (triplets).
G no_workload_classes = 3                          Number of workload classes.
G no_data_class_elements = 10 Number of data class elem. (initial data buckets).
G no_data_classes = 10                                Number of data classes.
```

```
#        >>> Note: This list may be extended in the near future!. <<<
#
# ****************************************************************************
# * List of USER_IDs (contains no_user_ids entries, record_type = U)           *
# ****************************************************************************
# Record Format: "U <u_key> <u_id_string> [comment]".
#
# A USER_ID is a character string without white-spaces.
# The maximum length is 8 characters.
# Each USER-ID string is uniquely associated with an integer key.
# The integer key is used to identify "it's" USER_ID string in the
# workload class description (see below). The order of the USER-IDs or
# the integer keys is immaterial. The integer key range must not have gaps.
# Zeroes are not allowed as key values. The number of USER-IDs has to be
# equal to the value of no_user_ids in the global section (this condition
# is checked by the program! Note: ID-Strings are case sensitive!!!
#
# IMPORTANT NOTE: Actually, the key list must be dense and in increasing
# ============== order. This may be changed in future versions of the
#                 modules which read and process this type of input file.
#                 Though it is not really necessary to explicitely represent
#                 the keys in this file (under this restriction), it makes
#                 debugging easier because the file is much easier to read.
#
U 1 UALPHA
U 2 UBRAVO
U 3 UCHARLY
#
#
# ****************************************************************************
# * List of TERM_IDs (contains no_term_ids entries, record_type = T)           *
# ****************************************************************************
# Record Format: "T <t_key> <t_id_string> [comment]".
#
# Specifications and restrictions of user_id list applies (see above).
#
T 1 TDELTA
T 2 TECHO
T 3 TFOXTROT
T 4 TGOLF
```

```
#
#
# ****************************************************************************
# * List of TRAN_IDs (contains no_tran_ids entries, record_type = X (xaction"))*
# ****************************************************************************
# Record Format: "X <x_key> <x_id_string> [comment]".
#
# Specifications and restrictions of user_id list applies (see above).
#
X 1 XINDIA
X 2 XJULIET
X 3 XKILO
X 4 XLIMA
X 5 XMIKE
#
#
#
# ****************************************************************************
# * Description  of Workload Class Elements (no_workload_class_elements, 'e')  *
# ****************************************************************************
# Record Format: "e <element_key> <u_key> <t_key> <x_key> [comment]".
#
# One entry for each occurring Workload Class Member. The members are
# some kind of "refined transaction types" and are uniquely defined
# by the triplet (user_id, term_id, tran_id). The IDs are
# not represented as strings, but as their integer keys as described in the
# above lists (U, T, and X).
#
e 1 1 2 3
e 2 1 2 4
e 3 1 3 1
e 4 1 4 1
e 5 3 3 2
e 6 1 2 5
e 7 3 1 4
e 8 1 4 5
e 9 2 1 5
e 10 2 2 4
e 11 2 3 3
e 12 2 4 2
```

```
#
#
# ********************************************************************************
# * Workload Class List        (contains no_workload_classes entires, rect = W)*
# ********************************************************************************
# Record Formats: "W <wl_class_key> <number_of_elements> [comment]".
#                 "w <element_key1> <...  _5> [comment]".
#
# Each workload class consists of a set of workload class members. The class
# description is introduced with a workload class record (record type 'W'). This
# record is followed by a series of 'w' records. A 'w' record lists up to
# 5 class members. A class member is identified by the element_key from the
# above element list. In an initial input file, each workload class has exactly
# one member.
#
#
# Workload Class Description (Class_1, 3 Members)
W 1 3
w 1
w 2
w 3
# Workload Class Description (Class_2, 4 Members)
W 2 5
w 9
w 10
w 11
w 12
# Workload Class Description (Class_3, 5 Members)
W 3 5
w 4
w 6
w 5
w 7
w 8
#
# ********************************************************************************
# * Description  of Data Class Elements (no_workload_class_elements, 'b')      *
# ********************************************************************************
# Record Format: "b <bucket_key> <min> <max> [comment]".
#
```

```
# A data class member is a continuous sequence ("bucket": min, ... max) of data
# objects accessed by workload class members/elements. A data class may be
# described by physical block numbers or lock names or whatever as long as
# the identifiers can be used to identify the physical location of data.
# Min and Max may have the same value. The data bucket is an atomic unit;
# Normally, two (or more) buckets are NOT "melted together". If two buckets are
# assumed to be similar, they are put into/assumed to be in the same data class!
#
b 1 0 9
b 2 10 19
b 3 20 29
b 4 30 39
b 5 40 49
b 6 50 59
b 7 60 69
b 8 70 79
b 9 80 89
b 10 90 99
#
#
#
#
# ********************************************************************************
# * Data Class List        (contains no_data_classes entires, rect = D)          *
# ********************************************************************************
# Record Formats: "D <d_class_key> <number_of_elements> [comment]".
#                 "d <bucket_key1> <...  _5> [comment]".
#
# Each data class consists of a set of data class members (buckets). The class
# description is introduced with a data class record (record type 'D'). This
# record is followed by a series of 'd' records. A 'd' record lists up to
# 5 class members. A class member is identified by the bucket_key from the
# above element list. In an initial input file, each data class has exactly one
# member.
#
# Data Class Description (Class_1, 1 member)
D 1 1
d 1
# Data Class Description (Class_2, 1 member)
D 2 1
```

```
d 2
# Data Class Description (Class_3, 1 member)
D 3 1
d 3
# Data Class Description (Class_4, 1 member)
D 4 1
d 4
# Data Class Description (Class_5, 1 member)
D 5 1
d 5
# Data Class Description (Class_6, 1 member)
D 6 1
d 6
# Data Class Description (Class_7, 1 member)
D 7 1
d 7
# Data Class Description (Class_8, 1 member)
D 8 1
d 8
# Data Class Description (Class_9, 1 member)
D 9 1
d 9
# Data Class Description (Class_10, 1 member)
D 10 1
d 10
#
#
#
#
# ****************************************************************************
# * Reference Matrix                                                         *
# ****************************************************************************
# Record Formats: "r <workload_class_key> <data_class_key> <#acceses> [comnt]".
#                 "u <workload_class_key> <data_class_key> <#acceses> [comnt]".
#
# This section contains all non_zero entries of the reference matrix. This
# matrix is organized as follows:
#
#  - a row for each workload class
#  - a column for each data class
```

```
#  - each entry consists of two sub-entries:
#    . number of retrieval/read accesses (record type 'r')
#    . number of update/write accesses (record type 'u')
#
# A line in this file describes exactly one sub_entry of the reference matrix.
#
# Though non_zero_entries may be stored here, this is not necessary and therefore
# a waste of disk space.
#
r 1 1 1.000000e+02
u 1 1 101
r 2 2 1000
u 2 2 1001
r 3 3 10000
u 3 3 10001
r 1 4 200
r 2 5 2000
r 3 6 20000
u 1 7 301
u 2 8 3001
u 3 9 30001
u 3 10 300001
#
#
# ********************************************************************************
# ********** This is the last line of the workload description file ******
```

Παράρτημα Γ

SCRIPT Reference Input File

```
# This is a comment
# Maximum line length is 80 characters.
# -------1---------2---------3---------4---------5---------6---------7---------8

#      FILENAME: reference-input.pro
#   DESCRIPTION: Contains a sample protocol script file, and shows all the
#                commands / settings of the script grammar.
# CREATION DATE: July 13rd, 1994
#

# Note: The order of the folowing SECTIONS and the order of each command
# ----- WITHIN the sections is immaterial.
#

# ****************************************************************************
# * Settings Section                                                         *
# ****************************************************************************
#
set maximum number of utilization classes = 15
# Sets the maximum acceptable number of utilization classes
# (i.e. the goal of the clustering algorithm)

set automatic clustering = ON
# Sets whether the clustering performed will re-compute the clustering
# distance automatically or not.  Valid choices are on/true OR off/false.

set metric = PLDM
# Sets the distance metric to be used by the clustering algorithm.
```

```
# For the time being valid choices are :
# PLDM = Pseudo Linear Dependency Metric
# VDM  = Vector Dependency Metric

set clustering distance = 0.2
# OR: set clustering distance = sqrt(2.0)
# OR: set clustering distance = sqrt(2.0) / 15
# Sets the (initial or not) clustering distance for the clustering algorithm.
# Valid parameters are of type double or sqrt(double) or sqrt(double) / integer.

set clustering distance increase ratio = 10%
set clustering distance decrease ratio = 10%
# Sets the ratio that CLOU will use while automatically re-adjusting the
# clustering distance (increase or decrease)

set clustering distance upper bound = 0.0001
set clustering distance lower bound = 1.0
# Sets the range of allowable values for the clustering distance
# Upper bound should definetely be 1 (or less) for PLDM

set maximum clustering distance increase ratio = 1000%
set maximum clustering distance decrease ratio = 90%
# Sets the maximum allowable ratio that CLUE can use while automatically
# re-adjusting the clustering distance (increase or decrease).
# Going off bounds equals to the termination of the algorithm.

set maximum number of ping-pongs = 3
# Sets the maximum number, the algorithm is allowed to "ping-pong", i.e.
# to change its mind about decreasing / increasing the clustering distance.

set max compression rate = 10%
# Sets the maximum acceptable compression rate of the number of PUCs.
# When the compression rates becomes more than that value
# the clustering distance has to be re-adjusted (decreased).
# Valid parameters are of type double, or percentage.

set k factor = 15%
# Sets the minimum acceptable compression of the compression rate of PUCs.
# When the compression rates becomes less than some k% of its original value,
# the clustering distance has to be re-adjusted (increased).
```

```
# Valid parameters are of type double, or percentage.

set goal almost reached at 5%
# Sets the "proximity" to the original goal criterion which is used
# to relax the k_factor criterion, when goal is almost reached.

set unify style = avg
# Sets the behaviour of the unify function regarding the update of the
# reference matrix. For the time being valid choices are :
# ADD: Simply adds up the vector rows of the workload classes to be unified.
# AVG: Calculates the average of the vector rows of the workload classes
#      to be unified.

set stop at no change = 10
# Sets whether to stop the clustering when a modification (i.e. increase) of
# the clustering distance does not lead to a reduction of the number of PUCs
# during a ECS for a certain number of times (10 in the example).
# Valid parameters are of type integer.

set write weight = 1
set read weight = 1
# Set the weight to use when adding up the number of read and the number of
# write acceses. Valid parameters are of type integer.

set maximum intervals coefficient = 299%
set minimum intervals coefficient = 43%
# Set the two coefficients that are used to define the range in which the
# number of rows for each intervals must fall in.



# ****************************************************************************
# * Commands Section                                                         *
# ****************************************************************************
#

store zeroes in column0
# OR: initialize column0
# Fills column 0 entries in the reference matrix with zeroes.

store sums in column0
```

```
# Fills column 0 entries in the reference matrix with the sum of
# the elements for each row.

wipeout zero rows
# Completely eliminates rows with all elements=0

sort
# The 1st preprocessing step: sorting of rows in descending order of references.

store sum squares in column0
# Fills column 0 entries in the reference matrix with the sum of
# the squares of the elements for each row.

downscale
# The 2nd preprocessing step: downscaling of all the reference matrix entries.
# (All matrix entries are divided by the maximum number of accesses.)

normalize
# The 3rd preprocessing step: normalization of the row vectors using the
# maximum vector norm.
# (All matrix entries are divided by the maximum of the maximum vector norms.)

do clustering
# Perform the clustering (used when automatic clustering is on).

do clustering with HALC
# Perform the clustering with HALC (= the default)
# (other possible candidates: BEA, KMEANS, ISODATA).


do clustering step
do 6 clustering steps
# Perform one or more Elementary Clustering Steps.

# ********************************************************************************
# *************** This is the last line of the protocol description file ******
```

Ελληνοαγγλικό Ευρετήριο Όρων

Αισθητήρας	Sensor
Αίτηση	Request
Ακμή	Edge
Ακρίτες	Outliers
Ακύρωση	Invalidation
Ανάδραση	Feedback
Ανάλυση κύριων συνιστωσών	Principal components analysis
Ανεξάρτητη από το σύστημα	System Independent
Ανισορροπία Φόρτου Εργασίας	Load Imbalance
Αντικείμενο	Object
Αντιστοίχιση	Mapping
Αξιολόγηση	Evaluation
Απαρίθμηση	Enumeration
Απομόνωση	Isolation
Ασαφές	Fuzzy
Ασαφή λογική	Fuzzy logic
Ατομικότητα	Atomicity
Βαθμός−εισόδου	In−degree
Βαθμός−εξόδου	Out−degree
Βάση Δεδομένων	Database
Βασικό Βήμα Ομαδοποίησης	Elementary Clustering Step
Γεγονός	Event
Γλώσσα Ορισμού Interface	Interface Definition Language (IDL)
Γράφημα	Graph
Γραφήματα Συμπεριφοράς Χρήστη	User Behaviour Graphs (UBG)
Δειγματοληψία	Sampling
Δενδρόγραμμα	Dendrogram

Διάστημα	Interval
Διαιρετικός	Divisive
Διαλογικός	Interactive
Διαμερίσεις	Partitions
Διαμόρφωση Συστήματος	System Configuration
Διασκορπισμός	Declustering
Διατάσσω	Sort
Διαφάνεια	Transparency
Διαχειριστής	Manager
Διεργασία Εξυπηρέτησης	Server
Διεργασία Πελάτης	Client
Δίκτυο	Network
Δοσοληψία	Transaction
Δοσοληψία Χρέωσης/Πίστωσης	Debit/Credit Transaction
Δρομολόγηση Δοσοληψιών	Transaction Routing
Δυναμικός	Dynamic
Εγγύτητα	Contingency, Proximity
Είσοδος/Έξοδος	Input/Output (I/O)
Εκτελέσιμο	Executable
Έλεγχος Συνδρομικότητας	Concurrency Control
Εμφάνιση	Occurence
Ενέργεια Δεσμού	Bond Energy
Έννοια	Concept
Εννοιολογικό Πλαίσιο	Conceptual Framework
Ενταμιευτής	Buffer
Εντολή	Command
Εξισορρόπηση Φόρτου Εργασίας	Load Balancing (LB)
Επανάληψη	Iteration
Επαναφορά	Recovery
Επεξεργασία Δοσοληψιών	Transaction Processing (TP)
Επίβλεψη	Supervision
Επίδοση	Performance
Επικάλυψη	Overlap
Εργαλείο Παρακολούθησης Επίδοσης	Performance Monitor
Εργασία	Task

Ερώτηση	Query
Ετερογενής	Heterogeneous
Ευρηματικός	Heuristic
Εύρος Μεταγωγής Δεδομένων	I/O Bandwidth
Ζευγνύον Δένδρο	Spanning Tree
Ιδιότητα	Attribute
Ιστορία Ζωής	Life History
Ισχυρή Σύνδεση	Tightly Coupled
Ιχνοληψία	Tracing
Ίχνος	Trace
Κάθετη Τμηματοποίηση	Vertical Fragmentation
Καθολικός	Global
Κανονικοποίηση	Normalization
Κατανεμημένο Σύστημα	Distributed System
Κατασκευή	Construct
Κεντρικοποιημένος	Centralized
Κέντρο	Center
Κέντρο Βάρους	Centroid
Κλιμάκωση	Scaling
Κοινόχρηστος	Shared
Κόμβος	Node
Λειτουργικό Σύστημα	Operating System
Λογισμικό	Software
Μεταβίβαση Αίτησης Προσπέλασης	Database Request Shipping
Μεταφερσιμότητα	Portability
Μη Ομοιόμορφη Προσπέλαση Μνήμης	Non Uniform Memory Access (NUMA)
Μικρό επιπλέον κόστος	Low overhead
Μονάδες Εργασίας	Units of work
Μονιμότητα	Durability
Μονοθετικός	Monothetic
Νήμα	Thread
Ομάδα	Cluster
Ομαδοποίηση	Clustering
Ομαδοποίηση Φόρτου Εργασίας	Workload Clustering
Ομοιόμορφη Προσπέλαση Μνήμης	Uniform Memory Access (UMA)

Οπτικοποίηση	Visualization
Παλινδρόμηση	Regression
Παραγωγικό	Generative
Παράλληλος	Parallel
Παροχή	Throughput
Περίπτωση	Instance
Περιορισμοί Ακεραιότητας	Integrity Constraints
Περιορισμοί Σειράς Εκτέλεσης	Precedence Constraints
Πιθανοτικός	Probabilistc
Πίνακας προσπελάσεων	Reference matrix
Πλαίσιο	Framework
Πολυεπεξεργαστής	Multiprocessor
Πολυθετικός	Polythetic
Πόρος	Resource
Προσανατολισμένο	Directed
Προσανατολισμένος σε Στόχους Επίδοσης	Goal Oriented
Προσομοίωση	Simulation
Προσπέλαση	Access
Πρότυπη μορφή	Pattern
Πρότυπο στοιχείο	Prototype
Ροή	Flow
Σάρωση	Scan
Σειριακός	Sequential
Σειριακός	Serial
Σημασιολογική	Semantic
Σημείο Παρατήρησης	Viewpoint
Στατικός	Static
Συγγένεια	Affinity
Συγγένεια Δεδομένων	Data Affinity
Συγχωνευτικός	Agglomerative
Σύζευξη	Coupling
Συμφόρηση	Contention
Σύνδεση	Link
Συνδιάλεξη	Session
Συνδυαστική έκρηξη	Combinatorial explosion

Συνέπεια	Consistency
Συνεχής Ακολουθία	Stream
Συστήματα Μαζικής Επεξεργασίας	Batch Systems
Συσχέτιση	Correlation
Σχεσιακός	Relational
Ταξινόμηση	Classification
Ταυτόχρονος	Concurrent
Τεκμηρίωση	Documentation
Τετραγωνικό Σφάλμα	Square Error
Τμήμα	Segment
Τυπική Απόκλιση	Variance
Υλικό	Hardware
Υπερχείλιση	Overflow
Υπεύθυνος Διαχείρισης Συστήματος	System Administrator
Φόρτος Εργασίας	Workload
Φωλιασμένος	Nested
Χαλαρή Σύνδεση	Loosely Coupled
Χαρακτηρισμός Φόρτου Εργασίας	Workload Characterization
Χαρακτηριστικά	Features
Χρόνος Απόκρισης	Response Time
Χρονοσφραγίδα	Timestamp
Χωρίς Απομακρυσμένες Προσπελάσεις Μνήμης	NO Remote Memory Access (NORMA)

Αγγλοελληνικό Ευρετήριο Όρων

Access	Προσπέλαση
Affinity	Συγγένεια
Agglomerative	Συγχωνευτικός
Atomicity	Ατομικότητα
Attribute	Ιδιότητα
Batch Systems	Συστήματα Μαζικής Επεξεργασίας
Bond Energy	Ενέργεια Δεσμού
Buffer	Ενταμιευτής
Center	Κέντρο
Centralized	Κεντρικοποιημένος
Centroid	Κέντρο Βάρους
Classification	Ταξινόμηση
Client	Διεργασία Πελάτης
Cluster	Ομάδα
Clustering	Ομαδοποίηση
Combinatorial explosion	Συνδυαστική έκρηξη
Command	Εντολή
Concept	Έννοια
Conceptual Framework	Εννοιολογικό Πλαίσιο
Concurrency Control	Έλεγχος Συνδρομικότητας
Concurrent	Ταυτόχρονος
Consistency	Συνέπεια
Construct	Κατασκευή
Contention	Συμφόρηση
Contingency	Εγγύτητα
Correlation	Συσχέτιση
Coupling	Σύζευξη

Data Affinity ..... Συγγένεια Δεδομένων

Database Request Shipping ..... Μεταβίβαση Αίτησης Προσπέλασης

Database ..... Βάση Δεδομένων

Debit/Credit Transaction ..... Δοσοληψία Χρέωσης/Πίστωσης

Declustering ..... Διασκορπισμός

Dendrogram ..... Δενδρόγραμμα

Directed ..... Προσανατολισμένο

Distributed System ..... Κατανεμημένο Σύστημα

Divisive ..... Διαιρετικός

Documentation ..... Τεκμηρίωση

Durability ..... Μονιμότητα

Dynamic ..... Δυναμικός

Edge ..... Ακμή

Elementary Clustering Step ..... Βασικό Βήμα Ομαδοποίησης

Enumeration ..... Απαρίθμηση

Evaluation ..... Αξιολόγηση

Event ..... Γεγονός

Executable ..... Εκτελέσιμο

Features ..... Χαρακτηριστικά

Feedback ..... Ανάδραση

Flow ..... Ροή

Framework ..... Πλαίσιο

Fuzzy logic ..... Ασαφή λογική

Fuzzy ..... Ασαφές

Generative ..... Παραγωγικό

Global ..... Καθολικός

Goal Oriented ..... Προσανατολισμένος σε Στόχους Επίδοσης

Graph ..... Γράφημα

Hardware ..... Υλικό

Heterogeneous ..... Ετερογενής

Heuristic ..... Ευρηματικός

I/O Bandwidth ..... Εύρος Μεταγωγής Δεδομένων

In−degree ..... Βαθμός−εισόδου

Input/Output (I/O) ..... Είσοδος/Έξοδος

Instance ..... Περίπτωση

Integrity Constraints ..... Περιορισμοί Ακεραιότητας

Interactive ..... Διαλογικός

Interface Definition Language (IDL) ..... Γλώσσα Ορισμού Interface

Interval ..... Διάστημα

Invalidation ..... Ακύρωση

Isolation ..... Απομόνωση

Iteration ..... Επανάληψη

Life History ..... Ιστορία Ζωής

Link ..... Σύνδεση

Load Balancing (LB) ..... Εξισορρόπηση Φόρτου Εργασίας

Load Imbalance ..... Ανισορροπία Φόρτου Εργασίας

Loosely Coupled ..... Χαλαρή Σύνδεση

Low overhead ..... Μικρό επιπλέον κόστος

Manager ..... Διαχειριστής

Mapping ..... Αντιστοίχιση

Monothetic ..... Μονοθετικός

Multiprocessor ..... Πολυεπεξεργαστής

NO Remote Memory Access (NORMA) ..... Χωρίς Απομακρυσμένες Προσπελάσεις Μνήμης

Nested ..... Φωλιασμένος

Network ..... Δίκτυο

Node ..... Κόμβος

Non Uniform Memory Access (NUMA) ..... Μη Ομοιόμορφη Προσπέλαση Μνήμης

Normalization ..... Κανονικοποίηση

Object ..... Αντικείμενο

Occurence ..... Εμφάνιση

Operating System ..... Λειτουργικό Σύστημα

Out−degree ..... Βαθμός−εξόδου

Outliers ..... Ακρίτες

Overflow ..... Υπερχείλιση

Overlap ..... Επικάλυψη

Parallel ..... Παράλληλος

Partitions ..... Διαμερίσεις

Pattern ..... Πρότυπη μορφή

Performance Monitor ..... Εργαλείο Παρακολούθησης Επίδοσης

Performance ..... Επίδοση

Polythetic ..... Πολυθετικός

Portability ..... Μεταφερσιμότητα

Precedence Constraints ..... Περιορισμοί Σειράς Εκτέλεσης

Principal components analysis ..... Ανάλυση κύριων συνιστωσών

Probabilistc ..... Πιθανοτικός

Prototype ..... Πρότυπο στοιχείο

Proximity ..... Εγγύτητα

Query ..... Ερώτηση

Recovery ..... Επαναφορά

Reference matrix ..... Πίνακας προσπελάσεων

Regression ..... Παλινδρόμηση

Relational ..... Σχεσιακός

Request ..... Αίτηση

Resource ..... Πόρος

Response Time ..... Χρόνος Απόκρισης

Sampling ..... Δειγματοληψία

Scaling ..... Κλιμάκωση

Scan ..... Σάρωση

Segment ..... Τμήμα

Semantic ..... Σημασιολογική

Sensor ..... Αισθητήρας

Sequential ..... Σειριακός

Serial ..... Σειριακός

Server ..... Διεργασία Εξυπηρέτησης

Session ..... Συνδιάλεξη

Shared ..... Κοινόχρηστος

Simulation ..... Προσομοίωση

Software ..... Λογισμικό

Sort ..... Διάταξη

Spanning Tree ..... Ζευγνύον Δέντρο

Square Error ..... Τετραγωνικό Σφάλμα

Static ..... Στατικός

Stream ..... Συνεχής Ακολουθία

Supervision ..... Επίβλεψη

System Administrator ..... Υπεύθυνος Διαχείρισης Συστήματος

System Configuration	Διαμόρφωση Συστήματος
System Independent	Ανεξάρτητη από το σύστημα
Task	Εργασία
Thread	Νήμα
Throughput	Παροχή
Throughput	Παροχή
Tightly Coupled	Ισχυρή Σύνδεση
Timestamp	Χρονοσφραγίδα
Trace	Ίχνος
Tracing	Ιχνοληψία
Transaction Processing (TP)	Επεξεργασία Δοσοληψιών
Transaction Routing	Δρομολόγηση Δοσοληψιών
Transaction	Δοσοληψία
Transparency	Διαφάνεια
Uniform Memory Access (UMA)	Ομοιόμορφη Προσπέλαση Μνήμης
Units of work	Μονάδες Εργασίας
User Behaviour Graphs (UBG)	Γραφήματα Συμπεριφοράς Χρήστη
Variance	Τυπική Απόκλιση
Vertical Fragmentation	Κάθετη Τμηματοποίηση
Viewpoint	Σημείο Παρατήρησης
Visualization	Οπτικοποίηση
Workload Characterization	Χαρακτηρισμός Φόρτου Εργασίας
Workload Clustering	Ομαδοποίηση Φόρτου Εργασίας
Workload	Φόρτος Εργασίας

Βιβλιογραφία

[ACRS94] A. K. Agrawala, M. Calzarossa, P. Rossaro, and G. Serazzi. "Workload Modelling in Multi−Window Environments", 1994.

[AH90] P. Arabie and L. Hubert. "The Bond Energy Algorithm Revisited". *IEEE Transactions on Systems, Man, and Cybernetics*, 20(1), Jan/Feb 1990.

[And73] M. Anderberg. *"Cluster Analysis for Applications"*. Academic Press, New York, 1973.

[Art78] H. Artis. "Capacity Planning for MVS computer systems". In D. Ferrari, editor, *Performance of Computer Installations*. North−Holland, 1978.

[BDE+95] E. Born, T. Delica, W. Erhl, L. Richter, and R. Riedl. "Characterization of Workloads for Distributed Processing: Methodology and Guide". Deliverable WP2/T.2.1/D4 of project LYDIA (available online at ftp.ics.forth.gr:/lydia/deliverables/4thDel/SNI_deliverable.ps), June 1995.

[Ber90] P. Bernstein. "Transaction Processing Monitors". *Communications of the ACM*, 33(11):75−−86, November 1990.

[Bla90] D. Black. *"Scheduling and Resource Management Techniques for Multiprocessors"*. PhD thesis, Carnegie Mellon University, 1990. CMU−CS−90−152.

[Boc94] H. Bock. "Classification and Clustering: Problems for the Future". In E. Diday, Y. Lechevallier, M. Schader, P. Bertrand, and B. Burtsch, editors, *"New Approaches in Classification and Data Analysis"*, pages 3−−24. Springer−Verlag, 1994.

[CDY86] D. W. Cornell, D. M. Dias, and P. S. Yu. "On Multisystem Coupling through Function−Request Shipping". *IEEE Transactions on Software Engineering*, 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.

[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.

[CK88] T. Casavant and J. Kuhl. ''A Taxonomy of Scheduling in General Purpose Distributed Computing Systems''. *IEEE Transactions on Software Engineering*, 14(2):141−−154, February 1988.

[CM94] M. Calzarossa and L. Massari. ''Measurement−Based Approach to Workload Characterization''. In R. Marie, G. Haring, and G. Kotsis, editors, *Proc. of the 7th International Conference on Modeling Techniques and Tools for Computer Perfomance Evaluation*, pages 123−−147, Vienna, 1994. Springer−Verlag.

[CMM+94] M. Calzarossa, L. Massari, A. Merlo, M. Pantano, and D. Tessera. ''MEDEA: A Tool for Workload Characterization of Parallel Systems '', 1994.

[CMVN92] S. Chakravarthy, J. Muthutaj, R. Varadarajan, and S. Navathe. ''An Objective Function for Vertically Partitioning Relations in Distributed Databases and its Analysis.''. Technical Report UF−CIS−TR−92−045, University of Florida, Computer and Information Sciences, 1992.

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

[DYC94] A. Dan, P. S. Yu, and J. Chung. ''Characterization of Database Access Pattern for Analytic Prediction of Buffer Hit Probability''. *VLDB Jounal*, 4(1):127−−154, 1994.

[Eve74] B. Everitt. *''Cluster Analysis''*. John Wiley & Sons, New York, 1974.

[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.

[FMS+94] R. Friedrich, J. Martinka, T. Sienknecht, M. Chelliah, and A. Ranganathan. ''A Distributed Performance Measurement System for Large−Scale Heterogeneous Environments''. Technical Report NSA−93−031, Hewlett−Packard Company, 1994.

[FNGD93] D. Ferguson, C. Nikolaou, L. Georgiadis, and K. Davies. ''Satisfying Response Time Goals in Transaction Processing Systems''. In *Proc. of 2nd Int. Conf. on Parallel and Distributed Information Systems*, January 1993.

[FNY88] D. Ferguson, C. Nikolaou, and Y. Yemini. "Microeconomic Algorithms for Dynamic Load Balancing in Distributed Computer Systems". In *Proc. of 8th Int. Conf. on Distributed Computing Systems*, June 1988.

[FNY93] D. Ferguson, C. Nikolaou, and Y. Yemini. "An Economy for Managing Replicated Data in Autonomous Distributed Systems". In *Proc. of Int. Symp. on Autonomous Decentralized Systems*, number Research Report RC18164, 1993.

[FSZ83] D. Ferrari, G. Serazzi, and A. Zeigner. *"Measurement and Tuning of Computer Systems"*. Prentice−Hall, New Jersey, 1983.

[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).

[GLJ76] D. P. Gaver, S. S. Lavenberg, and T. G. Price Jr. "Exploratory Analysis of Access Path Length Data for a Data Base Management System". *IBM Journal on Research and Development*, 20:449−−464, 1976.

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

[Gra93] J. Gray. *"The Benchmark Handbook for Database and Transaction Processing Systems"*. Morgan Kaufmann Publishers, second edition edition, 1993.

[Har83] G. Haring. "On Stochastic Models of Interactive Workloads". In A. K. Agrawala and S. K. Tripathi, editors, *Perfomance '83*, pages 133−−152. North−Holland, 1983.

[HS93] S. Haldar and D. K. Subramanian. "An affinity−based dynamic load balancing protocol for distributed transaction processing systems". *Performance Evaluation*, 17(1):53−−71, January 1993.

[JD88] A. Jain and R. Dubes. *"Algorithms for Clustering Data"*. Prentice−Hall, New Jersey, 1988.

[KD89] J. Kearns and S. DeFazio. "Diversity in Database Reference Behaviour". *Performance Evaluation*, 17(1):11−−19, May 1989.

[Kla92] O. Klaassen. ''Modeling Database Reference Behaviour''. In G. Balbo and G. Serrazi, editors, *Computer Perfomance Evaluation: Modelling Techniques and Tools*, pages 47−−60. North−Holland, 1992.

[LS76] P. A. Lewis and G. S. Shedler. ''Statistical Analysis of Non−stationary Series of Events in a Data Base System''. *IBM Journal on Research and Development*, 20:465−−482, 1976.

[MHC94] B. Miller, J. Hollingsworth, and M. Callaghan. ''The Paradyn Parallel Performance Tools and PVM''. Technical Report TR−1240, University of Wisconsin − Madison, 1994.

[MN95a] M. Marazakis and C. Nikolaou. Design and implementation of a simulator for transaction processing systems, 1995.

[MN95b] M. Marazakis and C. Nikolaou. ''Towards Adaptive Scheduling of Tasks in Transactional Workflows (to appear)''. In *Winter Simulation Conference*, 1995.

[MSW72] W. McCormick, P. Schweitzer, and T. White. ''Problem Decomposition and Data Reorganization by a Clustering Technique''. *Oper. Res.*, 20, Sep/Oct 1972.

[NCWD84] S. Navathe, S. Ceri, G. Wiederhold, and J. Dou. ''Vertical Partitioning Algorithms for Database Design''. *ACM Transactions on Database Systems*, 9(4), December 1984.

[NS95] C. Nikolaou and T. Saridakis. Arraytracer (in preparation), 1995.

[OV91] M. Ozsu and P. Valduriez. *''Principles of Distributed Database Systems''*. Prentice Hall, 1991.

[Raa93] K. Raatikainen. ''Cluster Analysis and Workload Classification''. *Performance Evaluation Review*, 20(4):24−−30, May 1993.

[RVH94] S. V. Raghavan, D. Vasukiammaiyar, and G. Haring. ''Generative Networkload Models for a Single Server Environment''. In *Proceedings of the 1994 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, pages 118−−127. ACM Publications, 1994.

[SAB+94] S. Sistare, D. Allen, R. Bowker, K. Jourdenais, J. Simons, and R. Title. ''A Sclalable Debugger for Massively Parallel Message−Passing Programs''. In *IEEE Parallel and Distributed Technology*, pages 50−−56, 1994.

[SR95] B. Schiemann and M. Rosenberger. ''IDL for Load Balancing''. Deliverable WP4/T.4.1/D6 of project LYDIA, June 1995.

[Val93] P. Valduriez. ''Parallel Database Systems: Open Problems and New Issues''. *Distributed and Parallel Databases*, 1:137−−165, 1993.

[VR92] N. Venkateswarlu and P. Raju. ''Fast ISODATA Clustering Algorithms''. *Pattern Recognition*, 25(3):335−−342, 1992.

[WH94] H. Wabnig and G. Haring. ''PAPS: The Parallel Program Performance Prediction Toolset''. In R. Marie, G. Haring, and G. Kotsis, editors, *Proc. of the 7th International Conference on Modeling Techniques and Tools for Computer Perfomance Evaluation*, volume 794, pages 283−−304, Vienna, 1994. Springer−Verlag.

[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 Press, 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.

[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.

[YD92] P. S. Yu and A. Dan. ''Impact of Wokload 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.

[YD94] 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, October 1994.

[YL91] P. S. Yu and A. Leff. ''On Robust Transaction Routing and Load Sharing''. *ACM Trans. Database Syst.*, 16(3):476−−512, September 1991.

[Zah71] C. Zahn. ‘‘Graph Theoretical Methods for Determining and Describing Gestalt Clusters’’. *IEEE Transactions on Computers*, C−20(1):68−−86, 1971.