Κεφάλαιο 8

Δυναμική βελτιστοποίηση διαχείρισης μνήμης για τις εφαρμογές πολυμέσων

Όπως εισάγεται ήδη στα προηγούμενα κεφάλαια αυτού του βιβλίου, πρέπει να αναπτυχθούν νέες μεθοδολογίες διαχείρισης μνήμης επιπέδου συστήματος για τις εφαρμογές πολυμέσων, λόγω της αυξανόμενης πολυπλοκότητας και της δραστικής ανόδου στις απαιτήσεις μνήμης. Στο παρελθόν, οι περισσότερες εφαρμογές που ήταν στις ενσωματωμένες πλατφόρμες, παρέμειναν κυρίως στην κλασική περιοχή της επεξεργασίας σήματος, και απέφευγαν ενεργά τους αλγορίθμους που χρησιμοποιούν τα δυναμικά στοιχεία αποεκχώρησης στο χρόνο εκτέλεσης, που αποκαλείται επίσης και ως δυναμική μνήμη (dynamic memory, DM από εδώ και στο εξής). Πρόσφατα, τα πολυμέσα και οι ασύρματες εφαρμογές δικτύων για να μεταφερθούν στα ενσωματωμένα συστήματα έχουν δεχτεί μια πολύ γρήγορη αύξηση της ποικιλίας, πολυπλοκότητας και λειτουργίας. Αυτές οι εφαρμογές (πχ. MPEG4 ή νέα πρωτόκολλα δικτύων) εξαρτώνται, με λίγες εξαιρέσεις, από το DM για τις διαδικασίες τους, λόγω της έμφυτης προβλεψιμότητας των δεδομένων εισόδου. Ο σχεδιασμός των τελικών ενσωματωμένων συστημάτων για το (στατικό) αποτύπωμα μνήμης στη χειρότερη περίπτωση[1] θα οδηγούσε να μην μπορεί να ικανοποιήσει τις απαιτήσεις αυτών των εφαρμογών. Ακόμα κι αν χρησιμοποιούνται οι μέσες τιμές των πιθανών εκτιμήσεων του αποτυπώματος μνήμης, αυτές οι στατικές λύσεις θα οδηγήσουν σε υψηλότερους αριθμούς αποτυπώματος μνήμης (δηλαδή, περίπου 25%) από τις λύσεις DM [77]. Επιπλέον, αυτές οι ενδιάμεσες στατικές λύσεις δεν λειτουργούν σε ακραίες περιπτώσεις δεδομένων εισόδου, ενώ οι υλοποιήσεις με DM μπορούν να το κάνουν, δεδομένου ότι μπορούν να κλιμακώσουν το απαραίτητο αποτύ-

[1]από όλα τα στοιχεία που τοποθετούνται στη μνήμη και που μετριούνται σε bits.

πωμα μνήμης. Κατά συνέπεια, πρέπει να χρησιμοποιηθούν στο σχεδιασμό ενσωματωμένων συστημάτων για δυναμικές εφαρμογές διαχειριστές δυναμικής μνήμης. Σήμερα, είναι πολλές οι γενικές διαθέσιμες πολιτικές διαχείρισης DM, και οι εφαρμογές τους, οι οποίες παρέχουν σχετικά καλή εκτέλεση και χαμηλό κατακερματισμό για συστήματα γενικής χρήσης [56]. Εντούτοις, για τα ενσωματωμένα συστήματα, τέτοιοι διαχειριστές πρέπει να εφαρμοστούν μέσα στο περιορισμένο λειτουργικό σύστημά τους (ΛΣ) και να λάβουν υπόψιν τους περιορισμένους διαθέσιμους πόρους. Κατά συνέπεια, τα πρόσφατα ενσωματωμένα ΛΣ (πχ. [78]) χρησιμοποιούν τους διαχειριστές DM σύμφωνα με την βασική ιεραρχία μνήμης και το είδος εφαρμογών που θα τρέξουν σε αυτούς.

Συνήθως, οι εξατομικευμένοι διαχειριστές DM έχουν ως σκοπό να βελτιώσουν την απόδοση [79], [56], αλλά μπορούν επίσης να χρησιμοποιηθούν για να βελτιστοποιήσουν σε μεγάλο βαθμό το αποτύπωμα μνήμης έναντι των διαχειριστών γενικού σκοπού DM, το οποίο επηρεάζει την τελική κατανάλωση ενέργειας και την απόδοση στα ενσωματωμένα συστήματα, καθώς πρέπει να μοιραστούν την περιορισμένη διαθέσιμη μνήμη, εντός ολοκληρωμένου κυκλώματος με πολλές ταυτόχρονες δυναμικές εφαρμογές. Για παράδειγμα, στους τρισδιάστατους οπτικούς αλγορίθμους, ένας κατάλληλα εξατομικευμένος σχεδιασμένος διαχειριστής DM μπορεί να βελτιώσει το αποτύπωμα μνήμης κατά 45% περίπου πέρα από τους συμβατικούς γενικής χρήσης διαχειριστές DM [77]. Πόσο μάλλον, όταν χρησιμοποιούνται οι εξατομικευμένοι διαχειριστές DM, οι σχεδιασμοί των οποίων πρέπει να βελτιστοποιηθούν χειρωνακτικά από τον υπεύθυνο για την ανάπτυξη, εξετάζοντας μόνο έναν περιορισμένο αριθμό εναλλακτικών λύσεων σχεδιασμού και εφαρμογής. Αυτές καθορίζονται βασισμένες στην εμπειρία και την έμπνευση του σχεδιαστή. Αυτή η εξερεύνηση είναι περιορισμένη λόγω της έλλειψης συστηματικών μεθοδολογιών, για να ερευνήσουν με συνέπεια το χώρο σχεδιασμού της διαχείρισης DM. Κατά συνέπεια, οι σχεδιαστές πρέπει να καθορίσουν, να κατασκευάσουν και να αξιολογήσουν χειρωνακτικά τις νέες εξατομικευμένες εφαρμογές των στρατηγικών DM, οι οποίες έχουν αποδειχθεί πολύ χρονοβόρες. Ο υπεύθυνος για την ανάπτυξη βρίσκεται αντιμέτωπος ακόμα, με τον καθορισμό της δομής του διαχειριστή DM και πώς να σχεδιάσει το περίγραμμα ανά περίπτωση, ακόμα κι αν το ενσωματωμένο ΛΣ προσφέρει την ιδιαίτερη υποστήριξη για τις τυποποιημένες γλώσσες, όπως η C ή C++.

Σε αυτό το κεφάλαιο, παρουσιάζουμε μια μεθοδολογία που επιτρέπει στους υπεύθυνους για την ανάπτυξη, να σχεδιάσουν τους εξατομικευμένους μηχανισμούς DM διαχείρισης, με το μειωμένο αποτύπωμα μνήμης που απαιτείται για αυτά τα δυναμικά πολυμέσα και τις ασύρματες εφαρμογές δικτύων. Καταρχήν, αυτή η μεθοδολογία οριοθετεί το σχετικό χώρο σχεδιασμού των διαχειριστικών αποφάσεων DM για ένα ελάχιστο αποτύπωμα μνήμης, στις δυναμικές ενσωματωμένες εφαρμογές. Μετά από αυτό, έχουμε μελετήσει τη σχετική επιρροή κάθε

απόφασης του χώρου σχεδιασμού για το αποτύπωμα μνήμης, και έχουμε καθορίσει μια κατάλληλη διάταξη για να προσπελάσουμε αυτό το χώρο σχεδιασμού σύμφωνα με τη συμπεριφορά DM αυτών των δυναμικών εφαρμογών. Κατά συνέπεια, οι κύριες συνεισφορές της μεθοδολογίας μας είναι διπλές: (1) ο καθορισμός μιας συνεπούς ορθογωνιοποίησης του χώρου σχεδιασμού της διαχείρισης DM για τα ενσωματωμένα συστήματα και (2) ο καθορισμός μιας κατάλληλης διάταξης των αποφάσεων σχεδιασμού, για τα δυναμικά πολυμέσα και τις ασύρματες εφαρμογές δικτύων (και οποιουδήποτε άλλου τύπου ενσωματωμένων εφαρμογών, που κατέχει τα ίδια δυναμικά χαρακτηριστικά αποεκχώρησης) για να ενισχυθούν οι σχεδιαστές και για να δημιουργήσουν προσαρμοσμένους διαχειριστές DM σύμφωνα με τη συγκεκριμένη δυναμική συμπεριφορά κάθε εφαρμογής.

Αυτό το κεφάλαιο οργανώνεται ως εξής. Στην Ενότητα 8.1, περιγράφουμε τη σχετική εργασία στα πλαίσια της διαχείρισης μνήμης για τις εφαρμογές πολυμέσων. Στην ενότητα 8.2 παρουσιάζουμε το χώρο σχεδιασμού σχετικής διαχείρισης DM των αποφάσεων, για ένα μειωμένο αποτύπωμα μνήμης στις δυναμικές εφαρμογές πολυμέσων. Στη ενότητα 8.3 καθορίζουμε τη διάταξη για να προσπελάσουμε αυτό το χώρο σχεδιασμού και να ελαχιστοποιήσουμε το αποτύπωμα μνήμης της υπό ανάλυση εφαρμογής. Στη ενότητα 8.4 περιγράφουμε τη σφαιρική ροή σχεδιασμού, που προτείνεται στην μεθοδολογία μας για να ελαχιστοποιήσει το αποτύπωμα μνήμης στις δυναμικές ενσωματωμένες εφαρμογές. Στη ενότητα 8.5 εισάγουμε τις περιπτωσιολογικές μελέτες μας και παρουσιάζουμε λεπτομερώς τα πειραματικά αποτελέσματα που επιτεύχθηκαν. Τέλος, στη ενότητα 8.6 συνάγουμε τα συμπεράσματά μας.

8.1 Σχετική Εργασία

Οι βάσεις ήδη καθιερώνονται αρκετά στην επιστημονική κοινότητα για την αποδοτική χρήση της δυναμικής μνήμης (dynamic memory, DM). Η διαθέσιμη βιβλιογραφία είναι αρκετή για τις γενικής χρήσης εφαρμογές και τις πολιτικές διαχειριστικού λογισμικού DM[56]. Προσπαθώντας να επαναχρησιμοποιηθεί αυτή η εκτενής διαθέσιμη βιβλιογραφία, στα ενσωματωμένα συστήματα, όπου η ομάδα των εφαρμογών που εκτελούνται είναι ιδιαίτερα απαιτητικές (πχ. καταναλωτικές συσκευές) τείνουν να χρησιμοποιήσουν τις παραλλαγές DM της γνωστής κατάστασης προόδου γενικής χρήσης. Παραδείγματος χάριν, η χρήση συστημάτων βασισμένη σε linux έχει ως βάση τον Lea DM διαχειριστή [79] [56] και επίσης τα βασισμένα σε Windows συστήματα (και σε κινητά, αλλά και υπολογιστές) περιλαμβάνουν τις ιδέες του διαχειριστή DM Kingsley [80], [81], [56]. Τέλος, κάποια ΛΣ πραγματικού χρόνου ενσωματωμένων συστημάτων (πχ. [78]) υποστηρίζουν DM αποεκχώρησης μέσω των εξατομικευμένων διαχειριστών DM που βασίζονται στους απλούς κατανεμητές περιοχών [82] με ένα λογικό επί-

πεδο απόδοσης, για τα συγκεκριμένα χαρακτηριστικά γνωρίσματα πλατφορμών. Όλες αυτές οι προσεγγίσεις προτείνουν τις βελτιστοποιήσεις εξετάζοντας τα συστήματα γενικής χρήσης, όπου η σειρά των εφαρμογών είναι πολύ ευρεία και απρόβλεπτη στο χρόνο σχεδιασμού. Αντίθετα, η προσέγγισή μας εκμεταλλεύεται τη πρόσθετη συμπεριφορά DM των πολυμέσων και των ασύρματων εφαρμογών δικτύων για να δημιουργηθούν οι προσαρμοσμένοι και αποδοτικοί διαχειριστές DM για τα ενσωματωμένα συστήματα.

Αργότερα, εμφανίστηκε η έρευνα για τους εξατομικευμένους διαχειριστές DM που λαμβάνουν συγκεκριμένη συμπεριφορά εφαρμογής για να βελτιώσουν την απόδοση [79], [83], [56]. Επίσης, για τη βελτίωση της ταχύτητας στα ιδιαίτερα περιορισμένα ενσωματωμένα συστήματα [84], προτείνεται να χωριστεί ο DM σε σταθερά block και να τοποθετηθεί σε έναν ενιαίο συνδεδεμένο κατάλογο με μια απλή (αλλά γρήγορη) κατάλληλη στρατηγική, πχ. την πρώτη ή επόμενη τοποθέτηση [56]. Επιπλέον, είναι διαθέσιμες κάποιες μερικώς διαμορφώσιμες βιβλιοθήκες διαχειριστών DM για να παρέχουν το χαμηλό υπερυψωμένο και υψηλό επίπεδο απόδοσης αποτυπώματος μνήμης, για τις χαρακτηριστικές συμπεριφορές μιας ορισμένης εφαρμογής (πχ. Obstacks) [56], ο οποίος είναι ένας εξατομικευμένος διαχειριστής DM που βελτιστοποιείται για μια συμπεριφορά σωρού εκχώρησης/αποεκχώρησης). Ομοίως, στο [83] προτείνεται ένας διαχειριστής DM που επιτρέπεται να καθορίσει τις πολλαπλάσιες περιοχές στη μνήμη με διάφορες καθορισμένες από το χρήστη λειτουργίες, για τη μνήμη αποεκχώρησης. Επιπλέον, δεδομένου ότι η χρήση των αντικειμενοστραφών γλωσσών στο σχεδιασμό ενσωματωμένων συστημάτων με την υποστήριξη για την αυτόματη ανακύκλωση των νεκρών (μη χρησιμοποιούμενων) αντικειμένων σε DM (συνήθως αποκαλούμενο ως συλλογή απορριμάτων), όπως η Java, έχει συντελεστεί αρκετή εργασία για να προτείνει αυτόματα τα περιορισμένα γενικά έξοδα συλλογής απορριμάτων χρησιμοποιώντας έναν αλγόριθμο σχετικό με την απόδοση, η οποία μπορεί να χρησιμοποιηθεί σε πραγματικού χρόνου συστήματα [85], [86]. Σε αυτό το πλαίσιο, επίσης οι επεκτάσεις υλικού έχουν προταθεί για να εκτελέσουν αποτελεσματικότερα τη συλλογή απορριμάτων [87]. Η κύρια διαφορά αυτών των προσεγγίσεων, έναντι των δικών μας είναι ότι στοχεύουν κυρίως στις βελτιστοποιήσεις απόδοσης και προτείνουν τις ειδικές λύσεις χωρίς να καθορίζουν έναν ολοκληρωμένο χώρο σχεδίασης και εξερεύνησης σχεδιασμού για τα δυναμικά ενσωματωμένα συστήματα, όπως προτείνουμε σε αυτό το κεφάλαιο.

Επιπλέον, έχει γίνει έρευνα για να παρέχει την αποδοτική υποστήριξη υλικού για τη διαχείριση DM. Στο [88] παρουσιάζεται μια μονάδα διαχειριστικής επέκτασης αντικειμένου για να χειριστεί την αποεκχώρηση των block μνήμης στο υλικό, χρησιμοποιώντας έναν αλγόριθμο που είναι μια παραλλαγή του κλασικού δυαδικού συστήματος buddy. Στο [89] προτείνεται μια ενότητα υλικού αποκαλούμενη SoCDMMU (δηλαδή, δυναμική διαχειριστική μονάδα μνή-

μης σε συστήματα εντός ολοκληρωμένου κυκλώματος, System On chip Dynamic Memory Management Unit), που αντιμετωπίζει τη σφαιρική μνήμη αποεκχώρησης εντός ολοκληρωμένου κυκλώματος για να επιτύχει έναν αιτιατό τρόπο, ώστε να διαιρεθεί η μνήμη μεταξύ των στοιχείων επεξεργασίας των σχεδίων SOC. Εντούτοις, το ΛΣ εκτελεί ακόμα τη διαχείριση της μνήμης που διατίθεται σε έναν αποκλειστικό επεξεργαστή εντός ολοκληρωμένου κυκλώματος. Όλες αυτές οι προτάσεις είναι πολύ σχετικές για τα ενσωματωμένα συστήματα όπου το υλικό μπορεί ακόμα να αλλάξει, ενώ η εργασία μας θεωρείται για τις σταθερές ενσωματωμένες αρχιτεκτονικές σχεδιασμού, όπου η προσαρμογή μπορεί να γίνει μόνο στο ΛΣ ή το επίπεδο λογισμικού.

Τέλος, πολύ έρευνα έχει εκτελεσθεί στις τεχνικές βελτιστοποίησης για το αποτύπωμα μνήμης, τη κατανάλωση ενέργειας, την απόδοση και άλλους σχετικούς παράγοντες, συμπληρωματικά με την εργασία μας και για τη χρήση στα στατικά στοιχεία που συνήθως είναι επίσης παρόντα στις δυναμικές εφαρμογές που εξετάζουμε.

8.2 Σχετικός χώρος σχεδιασμού για τη δυναμική διαχείριση μνήμης στα δυναμικά πολυμέσα και τις ασύρματες εφαρμογές δικτύων

Στην κοινωνία λογισμικού η βιβλιογραφία που είναι διαθέσιμη είναι πολύ για τις πιθανές επιλογές σχεδιασμού για τους μηχανισμούς DM διαχείρισης [56], αλλά καμία από τις προηγούμενες εργασίες δεν παρέχει ένα πλήρες χώρο σχεδιασμού, χρήσιμο για τις συστηματικές εφαρμογές εξερεύνησης σε λογισμικό και στα ασύρματα δίκτυα για τα ενσωματωμένα συστήματα. Ως εκ τούτου, στην Ενότητα 8.2.1 απαριθμούμε αρχικά το σύνολο των σχετικών αποφάσεων στο χώρο σχεδιασμού της διαχείρισης DM, για ένα μειωμένο αποτύπωμα μνήμης στις ασύρματες εφαρμογές δικτύων και στα δυναμικά πολυμέσα. Κατόπιν, στην Ενότητα 8.2.2 συνοψίζουμε εν συντομία τις αλληλεξαρτήσεις που παρατηρούνται μέσα σε αυτό το χώρο σχεδιασμού, οι οποίες μας επιτρέπουν μερικώς να διατάξουμε αυτόν τον απέραντο χώρο σχεδιασμού των αποφάσεων. Τέλος, στην Ενότητα 8.2.3 εξηγούμε πώς να δημιουργήσουμε σφαιρικούς DM διαχειριστές για τα δυναμικά πολυμέσα και τις ασύρματες εφαρμογές δικτύων.

8.2.1 Διαχειριστής δυναμικού χώρου μνήμης για μειωμένο αποτύπωμα μεγέθους

Η συμβατική διαχείριση DM αποτελείται βασικά από δύο χωριστούς στόχους, δηλαδή τη εκχώρηση και την αποεκχώρηση [56]. Πιο αναλυτικά, η εκχώ-

ρηση είναι ο μηχανισμός ο οποίος αναζητεί ένα block αρκετά μεγάλο για να ικανοποιήσει το αίτημα μιας δεδομένης εφαρμογής και η αποεκχώρηση είναι ο μηχανισμός που επιστρέφει αυτό το block στη διαθέσιμη μνήμη του συστήματος και επαναχρησιμοποιείται αργότερα από ένα άλλο αίτημα.

Στις πραγματικές εφαρμογές, τα block μπορούν να ζητηθούν και να επιστραφούν σε οποιαδήποτε διάταξη, δημιουργώντας κατά συνέπεια 'τις τρύπες' μεταξύ των χρησιμοποιημένων block. Αυτές οι τρύπες είναι γνωστές ως κατακερματισμός μνήμης. Αφ' ενός, ο εσωτερικός κατακερματισμός εμφανίζεται όταν επιλέγεται να ικανοποιηθεί ένα μεγαλύτερο block από αυτό που απαιτεί ένα αίτημα. Απ΄την άλλη όμως, εάν η ελεύθερη μνήμη για να ικανοποιήσει ένα αίτημα μνήμης είναι διαθέσιμη, αλλά μη συνεχής (έτσι δεν μπορεί να χρησιμοποιηθεί για εκείνο το αίτημα), καλείται εξωτερικός κατακερματισμός. Επομένως, πάνω από την κατανομή μνήμης και την αποεκχώρηση, ο διαχειριστής DM πρέπει επίσης να φροντίσει τα ζητήματα τεμαχισμού. Αυτό γίνεται από τα διαιρούμενα και συνεχόμενα ελεύθερα block για να κρατήσει τον τεμαχισμό μνήμης όσο το δυνατόν μικρότερο. Τέλος, για να υποστηρίξουν όλους αυτούς τους μηχανισμούς, οι πρόσθετες δομές δεδομένων πρέπει να συνταχθούν για να παρακολουθήσουν όλα τα ελεύθερα και χρησιμοποιημένα block, και τους μηχανισμούς ανασυγκρότησης. Κατά συνέπεια, για να δημιουργήσουμε έναν αποδοτικό διαχειριστή DM, πρέπει συστηματικά να ταξινομήσουμε τις αποφάσεις σχεδιασμού που μπορούν να ληφθούν, για να χειριστούν όλους τους πιθανούς συνδυασμούς των παραγόντων που έχουν επιπτώσεις στο υποσύστημα DM (πχ. τεμαχισμός, γενικά έξοδα των πρόσθετων δομών δεδομένων, κλπ).

Έχουμε ταξινομήσει όλες τις σημαντικές επιλογές σχεδιασμού, που αποτελούν τον χώρο σχεδιασμού του διαχειριστή DM στα διαφορετικά ορθογώνια δέντρα απόφασης. Το ορθογώνιο σημαίνει ότι οποιαδήποτε απόφαση σε οποιοδήποτε δέντρο μπορεί να συνδυαστεί με οποιαδήποτε απόφαση σε ένα άλλο δέντρο, και το αποτέλεσμα πρέπει να είναι ένας ενδεχόμενος έγκυρος συνδυασμός, καλύπτοντας κατά συνέπεια ολόκληρο το πιθανό χώρο σχεδιασμού. Κατόπιν, η σχετικότητα μιας ορισμένης λύσης σε κάθε συγκεκριμένο σύστημα εξαρτάται από τους περιορισμούς σχεδιασμού της, το οποίο υπονοεί ότι μερικές λύσεις μπορούν να μην συναντήσουν όλους τους περιορισμούς συγχρονισμού και δαπανών για εκείνο το συγκεκριμένο σύστημα. Επιπλέον, οι αποφάσεις στα διαφορετικά ορθογώνια δέντρα μπορούν να διαταχτούν διαδοχικά κατά τέτοιο τρόπο ώστε τα δέντρα να μπορούν να διαβούν χωρίς επαναλήψεις, εφ' όσον διαδίδονται οι κατάλληλοι περιορισμοί από ένα επίπεδο απόφασης σε όλα τα επόμενα επίπεδα. Βασικά, όταν ληφθεί μια απόφαση σε κάθε δέντρο, καθορίζεται ένας εξατομικευμένος διαχειριστής DM (στη σημείωση, τον ατομικό διαχειριστή DM μας) για ένα συγκεκριμένο σχεδιασμό συμπεριφοράς DM (συνήθως, μια από τις φάσεις συμπεριφοράς της εφαρμογής DM). Κατά αυτόν τον τρόπο μπορούμε να αναδημιουργήσουμε οποιουσδήποτε διαθέσιμους DM διαχειριστές

γενικού σκοπού [56] ή να δημιουργήσουμε ιδιαίτερα DM διαχειριστές.

Τα δέντρα έχουν ομαδοποιηθεί στις κατηγορίες σύμφωνα με τα διαφορετικά κύρια μέρη που μπορούν να διακριθούν στη διαχείριση DM [56]. Μια επισκόπηση των σχετικών κατηγοριών αυτού του χώρου σχεδιασμού για ένα μειωμένο αποτύπωμα μνήμης παρουσιάζεται στην Εικόνα 8.1. Αυτή η προσέγγιση επιτρέπει την μείωση της πολυπλοκότητας του σφαιρικού σχεδιασμού των διαχειριστών DM στα μικρότερα υπό-προβλήματα που μπορούν να αποφασιστούν τοπικά, καθιστώντας εφικτό τον καθορισμό μιας κατάλληλης διάταξης για να προσπελαστούν Έπειτα περιγράφουμε τις πέντε κύριες κατηγορίες και τα δέντρα απόφασης που παρουσιάζονται στην Εικόνα 8.1. Για κάθε έναν από αυτούς, εστιάζουμε στα δέντρα απόφασης που είναι σημαντικά για τη δημιουργία των διαχειριστών DM, με ένα μειωμένο αποτύπωμα μνήμης.

- Α. Η δημιουργία των δομών block, που διαχειρίζεται τις δομές δεδομένων block δημιουργείται και χρησιμοποιείται αργότερα από τους διαχειριστές DM για να ικανοποιήσει τα αιτήματα μνήμης. Πιο συγκεκριμένα, το δέντρο δομών block καθορίζει τα διαφορετικά block του συστήματος και των δομών εσωτερικού ελέγχου. Σε αυτό το δέντρο, περιλαμβάνουμε όλους τους πιθανούς συνδυασμούς δυναμικών τύπων στοιχείων (από τώρα και στο εξής αποκαλούμενων DDTs), που απαιτούνται για να αντιπροσωπεύσουν και να κατασκευάσουν οποιαδήποτε δυναμική αντιπροσώπευση [68], [77] στοιχείων, που χρησιμοποιούνται σε έναν διαχειριστή DM. Από την άλλη, το δέντρο μεγεθών block αναφέρεται στα διαφορετικά μεγέθη των βασικών διαθέσιμων block για τη διαχείριση DM, η οποία μπορεί να καθοριστεί ή όχι. Τρίτον, οι ετικέτες block και τα καταγραμμένα δέντρα πληροφοριών διευκρινίζουν τους πρόσθετους τομείς που απαιτούνται μέσα στο block για να αποθηκεύσουν τις πληροφορίες που χρησιμοποιούνται από τον DM διαχειριστή. Τέλος, το εύκαμπτο σε μέγεθος διαχειριστικό δέντρο block αποφασίζει, εάν οι διαιρεμένοι και συγχωνευμένοι μηχανισμοί ενεργοποιούνται ή αν η πρόσθετη μνήμη ζητείται από το σύστημα. Αυτό εξαρτάται από τη διαθεσιμότητα του μεγέθους του block μνήμης που έχει ζητηθεί.

- Β. Το τμήμα pool βασίζεται στο κριτήριο, το οποίο εξετάζει τον αριθμό των pool (δεξαμενές) που υπάρχουν (ή περιοχών μνήμης) στον DM διαχειριστή και τους λόγους για τους οποίους δημιουργούνται. Το δέντρο μεγέθους σημαίνει ότι τα pool μπορούν να υπάρξουν είτε περιέχοντας εσωτερικά στα block διάφορων μεγεθών, είτε αυτά μπορούν να διαιρεθούν έτσι ώστε μια pool να υπάρχει ανά διαφορετικό μέγεθος block. Επιπλέον, το δέντρο δομών pool διευκρινίζει τη σφαιρική δομή ελέγχου για τις διαφορετικές ομάδες του συστήματος. Σε αυτό το δέντρο περιλαμβάνουμε όλους τους πιθανούς συνδυασμούς DDTs που απαιτούνται για να

αντιπροσωπεύσουν και να κατασκευάσουν οποιαδήποτε δυναμική αντιπροσώπευση στοιχείων των pool μνήμης [68], [77].

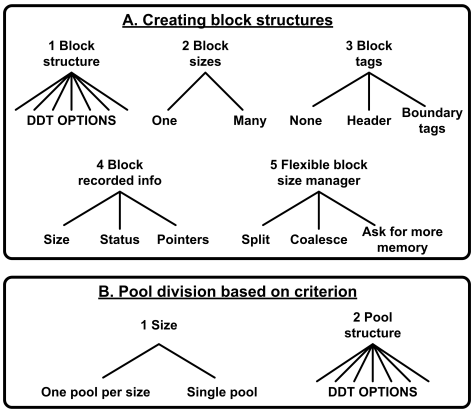
- Γ. Κατανεμημένα block, τα οποία εξετάζουν τις πραγματικές ενέργειες που απαιτούνται σε διαχείριση DM, για να ικανοποιήσουν τα αιτήματα μνήμης και να τα συνδέσουν με έναν ελεύθερο block μνήμης. Εδώ περιλαμβάνουμε όλες τις διαθέσιμες σημαντικές επιλογές προκειμένου να επιλέξουμε ένα block από έναν κατάλογο ελεύθερων block [56]. Σημειώστε ότι μια απελευθέρωση εμποδίζει την κατηγορία με τα ίδια δέντρα, όπως αυτή η κατηγορία μπόρεσε να δημιουργηθεί, αλλά δεν την περιλαμβάνουμε στην Εικόνα 8.1 για να αποφύγουμε την πολυπλοκότητα, όχι απαραίτητα στο χώρο σχεδιασμού διαχείρισης DM. Το γεγονός είναι ότι η κατηγορία διάθεσης block κατέχει περισσότερη επιρροή για το αποτύπωμα μνήμης από την πρόσθετη κατηγορία block. Επιπλέον, αυτές οι δύο κατηγορίες συνδέονται τόσο στενά σχετικά με το αποτύπωμα μνήμης της τελικής λύσης, ώστε οι αποφάσεις να λαμβάνονται μόνον όταν ακολουθούνται σε ένα άλλο. Κατά συνέπεια, η απελευθερωμένη κατηγορία block καθορίζεται εντελώς αφού επιλέξει τις επιλογές αυτής της κατηγορίας διάθεσης block.

- Δ. Συνεχή block, τα οποία συσχετίζονται με τις ενέργειες που εκτελούνται από τους DM διαχειριστές για να εξασφαλίσουν ένα χαμηλό ποσοστό της εξωτερικής μνήμης τεμαχισμού, ενώνοντας δύο μικρότερα block σε έναν μεγαλύτερο. Κατ' αρχάς, ο αριθμός ανώτατου δέντρου μεγέθους block καθορίζει τα νέα μεγέθη block που επιτρέπονται μετά από συγχώνευση δύο διαφορετικών συνεχόμενων block. Το δέντρο καθορίζει πόσο συχνά πρέπει να εκτελεσθεί η συγχώνευση.

- Ε. Διαιρεμένα block, τα οποία αναφέρονται στις ενέργειες που εκτελούνται από τους διαχειριστές DM για να εξασφαλίσουν ένα χαμηλό ποσοστό του εσωτερικού τεμαχισμού μνήμης, δηλαδή χωρίζοντας έναν μεγαλύτερο block σε δύο μικρότερα. Κατ' αρχάς, ο αριθμός ελάχιστου δέντρου μεγέθους block, καθορίζει νέα μεγέθη block που επιτρέπονται μετά από τον διαχωρισμό ενός φραγμού σε μικρότερους. Και τότε καθορίζει το δέντρο πόσο συχνά πρέπει να εκτελεσθεί ο διαχωρισμός (αυτά τα δέντρα δεν παρουσιάζονται στο πλήρες στοιχείο στην Εικόνα 8.1, επειδή οι επιλογές είναι οι ίδιες όπως στα δύο δέντρα της συγχωνευμένης κατηγορίας).

8.2.2 Αλληλεξαρτήσεις μεταξύ των ορθογώνιων δέντρων

Μετά από αυτόν τον καθορισμό των κατηγοριών και των δέντρων απόφασης, σε αυτό το τμήμα προσδιορίζουμε τις πιθανές αλληλεξαρτήσεις τους. Επι-



Σχήμα 8.1: Ο χώρος σχεδιασμού διαχείρισης DM των ορθογώνιων αποφάσεων για το μειωμένο αποτύπωμα μνήμης.

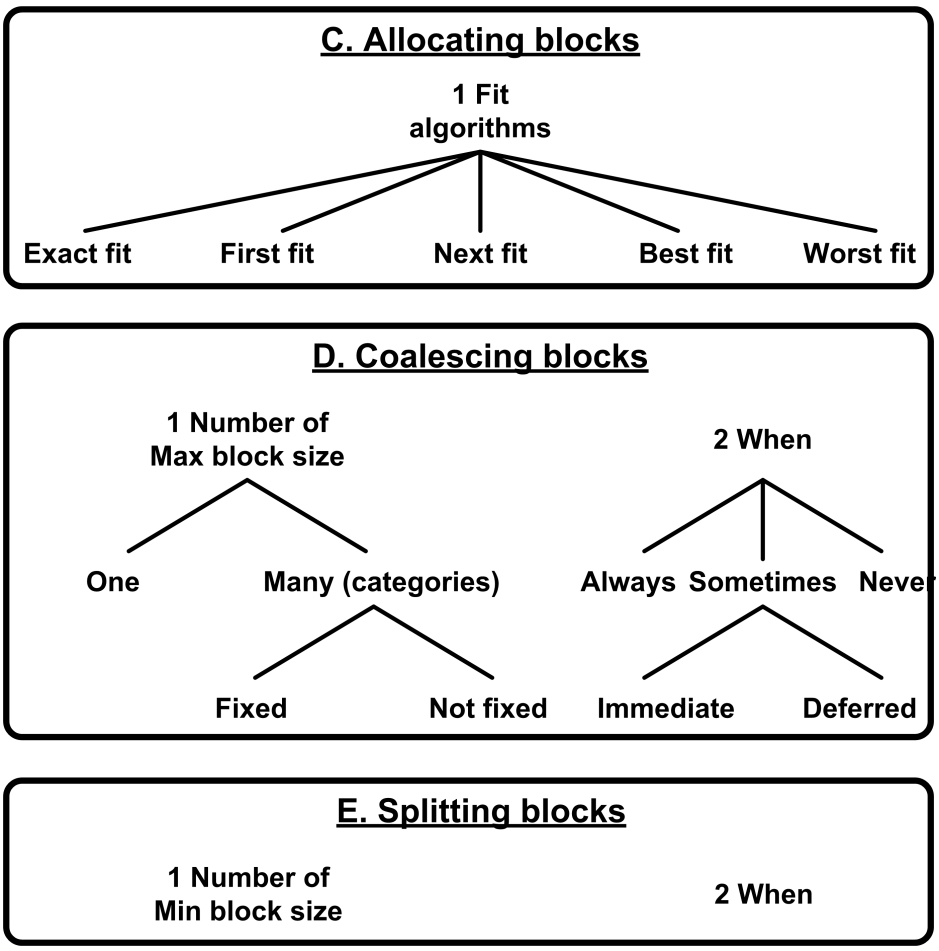
βάλλεται μια μερική διάταξη στο χαρακτηρισμό των διαχειριστών DM. Τα δέντρα απόφασης είναι ορθογώνια, αλλά μη ανεξάρτητα. Επομένως, η επιλογή ορισμένων φύλλων σε μερικά δέντρα έχει μεγάλες επιπτώσεις στις συναφείς αποφάσεις σε άλλες (δηλαδή, αλληλεξαρτήσεις), όταν σχεδιάζεται ένας ορισμένος διαχειριστής DM. Ολόκληρο το σύνολο αλληλεξαρτήσεων για το χώρο σχεδιασμού παρουσιάζεται στην Εικόνα 8.2. Αυτές οι αλληλεξαρτήσεις μπορούν να ταξινομηθούν σε δύο κύριες ομάδες. Κατ' αρχάς, χρήση άλλων δέντρων ή κατηγοριών (που σύρονται ως πλήρη βέλη στην Εικόνα 8.2). Δεύτερον, οι αλληλεξαρτήσεις που έχουν επιπτώσεις σε άλλα δέντρα ή κατηγορίες, λόγω των συνδεόμενων σκοπών τους (που παρουσιάζονται ως βέλη στην Εικόνα 8.2). Τα βέλη δείχνουν ότι η αρχική πλευρά έχει επιπτώσεις στο λαμβάνοντα και πρέπει να αποφασιστεί πρώτα.

Φύλλα ή δέντρα που εμποδίζουν τη χρήση άλλων στο χώρο σχεδιασμού

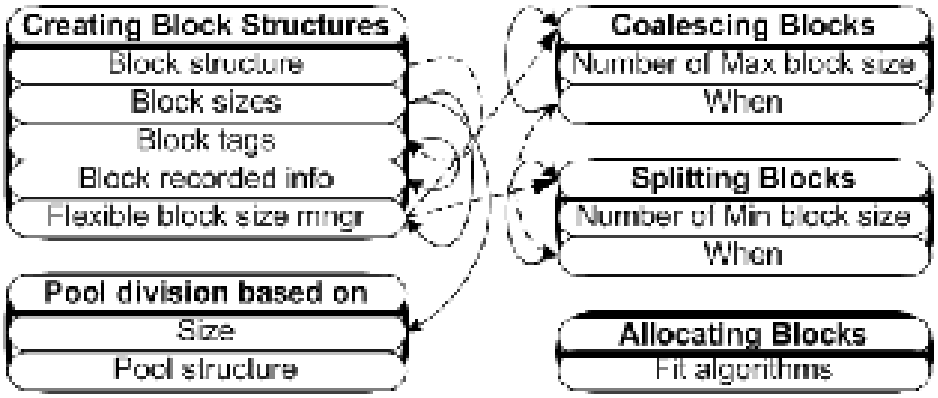
Αυτές οι αλληλεξαρτήσεις εμφανίζονται και οφείλονται στην ύπαρξη των αντίθετων φύλλων και των δέντρων στο χώρο σχεδιασμού. Απεικονίζονται στην Εικόνα 8.2 και αντιπροσωπεύονται από τα πλήρη τόξα:

- Κατ' αρχάς, μέσα στη κατηγορία δημιουργίας δομών block, εάν κανένα φύλλο από το δέντρο ετικετών block δεν επιλεχθεί, τότε το καταγεγραμμένο block του δέντρου πληροφοριών, δεν μπορεί να χρησιμοποιηθεί: δεν θα υπήρχε κανένας χώρος μνήμης για να αποθηκεύσει τις καταγεγραμμένες πληροφορίες μέσα στο block. Αυτή η αλληλεξάρτηση παρουσιάζεται γραφικά για παράδειγμα στην Εικόνα 8.3.

- Το ένα φύλλο από το δέντρο μεγεθών block αποκλείει τη χρήση του δέ-



Σχήμα 8.2: Αλληλεξαρτήσεις μεταξύ των ορθογώνιων δέντρων στο χώρο σχεδιασμού.



Σχήμα 8.3: Παράδειγμα της αλληλεξάρτησης μεταξύ δύο ορθογώνιων δέντρων του διαστήματος διαχείρισης σχεδιασμού DM.

ντρου μεγέθους στο τμήμα pool που εδρεύει στην κατηγορία κριτηρίου και του εύκαμπτου δέντρου διαχειριστών μεγέθους block δημιουργώντας τη κατηγορία δομών block. Αυτό εμφανίζεται επειδή το ένα φύλλο μεγέθους block δεν επιτρέπει να καθορίσει οποιαδήποτε νέα μεγέθη.

Φύλλα ή δέντρα που περιορίζουν τη χρήση άλλων στο χώρο σχεδιασμού

Αυτές οι αλληλεξαρτήσεις υπάρχουν, δεδομένου ότι τα φύλλα πρέπει να συνδυαστούν για να δημιουργήσουν ολόκληρους συνεπείς σχεδιασμούς DM. Παραδείγματος χάριν, οι διαιρεμένοι και συγχωνευμένοι μηχανισμοί συσχετίζονται αρκετά και οι αποφάσεις σε μια κατηγορία πρέπει να βρουν ισοδύναμες σε άλλες κατηγορίες. Αυτές οι αλληλεξαρτήσεις αντιπροσωπεύονται με τα τόξα στην Εικόνα 8.2

Κατ' αρχάς, το εύκαμπτο δέντρο διαχειριστών μεγέθους block επηρεάζει πολύ όλα τα δέντρα μέσα στα συγχωνευμένα block και τις διαιρεμένες κατηγορίες block. Κατά συνέπεια, σύμφωνα με το επιλεγμένο φύλλο για έναν ορισμένο ατομικό διαχειριστή DM (δηλαδή, διάσπαση ή συνεχή φύλλο), ο διαχειριστής DM πρέπει να επιλέξει μερικά φύλλα των δέντρων που περιλαμβάνονται ή δεν περιλαμβάνονται σε εκείνες τις αποφάσεις. Παραδείγματος χάριν, εάν το διασπασμένο φύλλο επιλέγεται, ο διαχειριστής DM δεν θα χρησιμοποιήσει τη συνεχή κατηγορία και τις λειτουργίες των αντίστοιχων εσωτερικών δέντρων τους, που έχει ο ατομικός διαχειριστής DM. Εντούτοις, μπορεί να χρησιμοποιηθεί σε έναν άλλο ατομικό διαχειριστή DM, σε μια διαφορετική φάση της DM εκχώρησης, κατά συνέπεια ο τελικός σφαιρικός διαχειριστής περιέχει και τους δύο (δείτε την Ενότητα 8.2.3 για περισσότερες λεπτομέρειες).

Εντούτοις, το κύριο κόστος της επιλογής που γίνεται στο εύκαμπτο δέντρο

διαχειριστών μεγέθους block, χαρακτηρίζεται από το κόστος των κατηγοριών των συγχωνευμένων block όπως και των διαιρεμένων block. Αυτό σημαίνει ότι ένας λεπτομερής εκτιμητής της επιρροής δαπανών πρέπει να είναι διαθέσιμος για να πάρει την απόφαση στο προαναφερθέν εύκαμπτο δέντρο διαχειριστών μεγέθους block. Στην πραγματικότητα, αυτός ο εκτιμητής είναι απαραίτητος, εάν οι παραγόμενες αποφάσεις σε άλλες κατηγορίες έχουν μια επιρροή στο τελικό κόστος μικρότερη από την επιρροή του δέντρου που αποφασίζει να τους χρησιμοποιήσει [90]. Αυτό δεν συμβαίνει στο δέντρο το οποίο ορίζει και έτσι κανένας εκτιμητής δεν απαιτείται.

- Δεύτερον, η απόφαση που λαμβάνεται στο δέντρο δομών pool έχει σημαντικές επιπτώσεις σε ολόκληρο το τμήμα pool που εδρεύει στην κατηγορία κριτηρίου. Αυτό συμβαίνει επειδή μερικές δομές δεδομένων περιορίζουν ή δεν επιτρέπουν στη pool να διαιρεθεί με τους σύνθετους τρόπους που τα κριτήρια αυτής της κατηγορίας προτείνουν.

- Τρίτον, το δέντρο δομών block μέσα στη κατηγορία δομών block η οποία δημιουργείται, επηρεάζει έντονα την απόφαση στο δέντρο ετικετών block της ίδιας κατηγορίας, επειδή ορισμένες δομές δεδομένων απαιτούν τους πρόσθετους τομείς για τη συντήρησή τους. Παραδείγματος χάριν, οι ενιαίοι συνδεόμενοι κατάλογοι απαιτούν έναν επόμενο τομέα και ένας κατάλογος όπου διάφορα μεγέθη block επιτρέπονται και πρέπει να περιλάβει έναν τομέα επιγραφών και μέσα το μέγεθος κάθε ελεύθερου φραγμού. Διαφορετικά το κόστος για να διαβεί τον κατάλογο και να βρει ένα κατάλληλο block για το ζητούμενο μέγεθος κατανομής, είναι υπερβολικά μεγάλο [68].

- Τέλος, τα αντίστοιχα When δέντρα από τις διαιρεμένες και συγχωνευμένες κατηγορίες block μαζί με ένα διπλό βέλος στην Εικόνα 8.3 συσχετίζονται πολύ στενά ο ένας με τον άλλον και μια διαφορετική απόφαση σε κάθε δέντρο, δεν φαίνεται να παρέχει οποιοδήποτε είδος οφέλους στην τελική λύση. Αντίθετα, σύμφωνα με τη μελέτη και τα πειραματικά αποτελέσματά μας, αυξάνει συνήθως το κόστος στο αποτύπωμα μνήμης της τελικής λύσης διαχειριστών DM. Εντούτοις, αυτό το διπλό βέλος απαιτείται επειδή δεν είναι δυνατό να αποφασιστεί ποια κατηγορία έχει περισσότερη επιρροή στην τελική λύση χωρίς μελέτη των ειδικών παραγόντων της επιρροής για ορισμένο μετρικό, για να βελτιστοποιήσει (πχ. αποτύπωμα μνήμης, κατανάλωση ενέργειας, απόδοση, κλπ). Κατά συνέπεια, η απόφαση για την οποία η κατηγορία πρέπει να αποφασιστεί, πρώτα πρέπει να αναλυθεί για κάθε ιδιαίτερη συνάρτηση κόστους ή είναι μετρικά απαραίτητο να βελτιστοποιηθεί στο σύστημα.

8.2.3 Κατασκευή των σφαιρικών δυναμικών διαχειριστών μνήμης

Τα σύγχρονα πολυμέσα και οι ασύρματες εφαρμογές δικτύων περιλαμβάνουν διαφορετικούς σχεδιασμούς συμπεριφοράς DM, τα οποία συνδέονται με τις λογικές φάσεις τους (δείτε το Τμήμα 8.5 για τα πραγματικά παραδείγματα). Συνεπώς, η μεθοδολογία μας πρέπει να εφαρμοστεί σε κάθε μια από αυτές τις διαφορετικές φάσεις προκειμένου να δημιουργηθεί χωριστά ένας εξατομικευμένος ατομικός διαχειριστής DM για κάθε μια από αυτές. Κατόπιν, ο σφαιρικός διαχειριστής DM της εφαρμογής είναι ο συνυπολογισμός όλων αυτών των ατομικών διαχειριστών DM σε ένα. Για αυτόν τον λόγο, έχουμε αναπτύξει τη βιβλιοθήκη C++ που εδρεύει στις αφηρημένες κατηγορίες ή τα πρότυπα που καλύπτουν όλες τις πιθανές αποφάσεις στο χώρο σχεδιασμού DM και μας επιτρέπουν την κατασκευή της τελικής σφαιρικής εφαρμογής των εξατομικευμένων διαχειριστών DM με έναν απλό τρόπο μέσω της σύνθεσης C++ των στρωμάτων [91].

8.3 Διάταξη για το μειωμένο δυναμικό αποτύπωμα μνήμης στα δυναμικά πολυμέσα και τις ασύρματες εφαρμογές δικτύων

Μόλις καθοριστεί ολόκληρος ο χώρος σχεδιασμού για τη διαχείριση DM στα δυναμικά ενσωματωμένα συστήματα και ταξινομηθεί στο προηγούμενο τμήμα, η διάταξη για τους διαφορετικούς τύπους εφαρμογών μπορεί να καθοριστεί σύμφωνα με τη συμπεριφορά και το κόστος συναρτήσεων DM που βελτιστοποιείται. Σε αυτήν την περίπτωση, το υποσύστημα DM βελτιστοποιείται για να επιτύχει τις λύσεις με ένα μειωμένο αποτύπωμα DM. Επομένως, πρώτα στην Ενότητα 8.3.1 συνοψίζουμε τους παράγοντες της επιρροής για το αποτύπωμα DM. Κατόπιν, στο Τμήμα 8.3.2 περιγράφουμε εν συντομία τα χαρακτηριστικά γνωρίσματα που επιτρέπουν σε μας να ομαδοποιήσουμε τις διαφορετικές δυναμικές εφαρμογές και να εστιάσουμε στα κοινά (και ιδιαίτερα) χαρακτηριστικά γνωρίσματα των πολυμέσων και τις ασύρματες εφαρμογές δικτύων, οι οποίες μας επιτρέπουν να συγκεντρώσουμε αυτές τις εφαρμογές και να καθορίσουμε μια κοινή διάταξη εξερεύνησης του χώρου σχεδιασμού. Τέλος, στην Ενότητα 8.3.2 παρουσιάζουμε την κατάλληλη διάταξη εξερεύνησης για αυτά τα πολυμέσα και τις ασύρματες εφαρμογές δικτύων για να επιτευχθούν οι μειωμένες λύσεις διαχείρισης αποτυπώματος μνήμης DM.

8.3.1 Παράγοντες της επιρροής για τη δυναμική εξερεύνηση αποτυπώματος μνήμης

Οι κύριοι παράγοντες που έχουν επιπτώσεις στο μέγεθος μνήμης είναι δύο: η υπερυψωμένη οργάνωση και τα απόβλητα μνήμης τεμαχισμού.

1. Τα γενικά έξοδα οργάνωσης είναι τα γενικά έξοδα που παράγονται από τους υποβοηθούμενους τομείς και τις δομές δεδομένων, τα οποία συνοδεύουν κάθε block και pool αντίστοιχα. Αυτή η οργάνωση είναι ουσιαστική να διαθέσει, να απελευθερώσει και να χρησιμοποιήσει τα block μνήμης μέσα στις pool, και εξαρτάται από τα ακόλουθα μέρη:

    - Οι τομείς (πχ. επιγραφές, υποσημειώσεις, κλπ) μέσα στα block μνήμης χρησιμοποιούνται για να αποθηκεύσουν τα στοιχεία σχετικά με το συγκεκριμένο block και είναι συνήθως μερικά bytes μακριά. Η χρήση αυτών των τομέων ελέγχεται από την κατηγορία Α (που δημιουργεί τις δομές block) στο χώρο σχεδιασμού.
    - Οι υποστηριζόμενες δομές δεδομένων, παρέχουν την υποδομή για να οργανώσουν την pool και για να χαρακτηρίσουν τη συμπεριφορά της. Μπορούν να χρησιμοποιηθούν για να αποτρέψουν τον τεμαχισμό με τον καταναγκασμό των block, για να διατηρήσουν τη μνήμη σύμφωνα με το μέγεθός τους, χωρίς απαραίτητα να πρέπει να διαιρεθούν και να συγχωνευτούν. Η χρήση αυτών των δομών δεδομένων ελέγχεται από την κατηγορία Β (το τμήμα pool βασίζεται στο κριτήριο αυτό). Σημειώστε ότι οι ίδιες επικουρικές δομές δεδομένων βοηθούν στο να αποτρέψουν τον τεμαχισμό, αλλά σιωπηρά παράγουν κάποια γενικά έξοδα. Τα γενικά έξοδα που παράγουν μπορούν να είναι συγκρίσιμα με τον τεμαχισμό που παράγεται από τις μικρές δομές δεδομένων. Εν τούτοις αυτός ο αρνητικός παράγοντας ξεπερνιέται από τη δυνατότητά τους να αποτρέψουν τα προβλήματα τεμαχισμού, ο οποίος είναι ένας πιο σχετικός αρνητικός παράγοντας για το αποτύπωμα μνήμης [56]. Η ίδια επίδραση στην πρόληψη τεμαχισμού είναι επίσης παρούσα στο δέντρο C1. Αυτό συμβαίνει επειδή ανάλογα με τον κατάλληλο αλγόριθμο που επιλέγεται, είναι δυνατό να μειωθεί ο εσωτερικός τεμαχισμός του συστήματος. Παραδείγματος χάριν, εάν διατίθεται ένα τμήμα 12 bytes χρησιμοποιώντας έναν επόμενο κατάλληλο αλγόριθμο και ο επόμενος φραγμένος μέσα στη pool είναι 100 bytes, 88 bytes χάνονται στον εσωτερικό τεμαχισμό ενώ μια καλή εφαρμογή αλγορίθμου, πιθανώς δεν θα χρησιμοποιήσει ποτέ οποιοδήποτε block για το αίτημα. Επομένως, αποτρέποντας τον τεμαχισμό, η κατηγορία Γ είναι εξίσου σημαντική με τη Β.

2. Τα απόβλητα μνήμης τεμαχισμού προκαλούνται από τον εσωτερικό και εξωτερικό τεμαχισμό, που συζητήθηκε νωρίτερα σε αυτό το κεφάλαιο, το οποίο εξαρτάται από τα εξής:

    - Ο εσωτερικός τεμαχισμός θεραπεύεται συνήθως από την κατηγορία Ε (διαιρεμένα block). Αυτό έχει επιπτώσεις συνήθως στις μικρές δομές δεδομένων. πχ. εάν έχετε μόνο 100 bytes block μέσα στις pool σας και εσείς θέλετε να διαθέσετε 20 bytes block, θα ήταν σοφό να χωριστεί κάθε block σε 100 bytes μέσα στη pool σας, σε 5 block 20 bytes για να αποφευχθεί ένα μεγάλο μέρος εσωτερικού τεμαχισμού.
    - Ο εξωτερικός τεμαχισμός θεραπεύεται συνήθως από την κατηγορία Δ (συγχωνευμένα block). Έχει επιπτώσεις συνήθως στα μεγάλα αιτήματα στοιχείων. Παραδείγματος χάριν, εάν ένα block 50KB πρέπει να διατεθεί, αλλά υπάρχουν μόνο 500 bytes block μέσα στις pool, θα ήταν απαραίτητο να συγχωνευτούν 100 bytes block για να παραχθεί το ζητούμενο ποσό μνήμης.

Σημειώστε τη διάκριση μεταξύ των κατηγοριών Δ και Ε, που προσπαθούν να εξετάσουν τον τεμαχισμό, σε αντιδιαστολή με την κατηγορία Β και Γ που προσπαθούν να τον αποτρέψουν.

8.3.2 Ανάλυση αποεκχώρησης των χαρακτηριστικών των δυναμικών ενσωματωμένων πολυμέσων και των ασύρματων εφαρμογών δικτύων

Τα δυναμικά πολυμέσα και οι ασύρματες ενσωματωμένες εφαρμογές δικτύου περιλαμβάνουν διάφορες μη−κατανεμημένες φάσεις (ή οι σχεδιασμοί) για τις δομές δεδομένων τους, οι οποίες αντιπροσωπεύουν συνήθως τις διαφορετικές φάσεις στη λογική της ίδιας εφαρμογής. Έχουμε ταξινομήσει αυτούς τους διαφορετικούς σχεδιασμούς εκχώρησης DM σε τρία ορθογώνια συστατικά [56], δηλαδή, το κυρίαρχο μέγεθος block κατανομής, τον κύριο σχεδιασμό και το ποσοστό των εντατικών φάσεων μνήμης αποεκχώρησης.

Όπως έχουμε παρατηρήσει εμπειρικά, χρησιμοποιώντας αυτήν την προηγούμενη ταξινόμηση βασισμένη στα συστατικά, τα δυναμικά πολυμέσα και οι ασύρματες εφαρμογές δικτύων, μοιράζονται τον κεντρικό αγωγό που χαρακτηρίζει την εκτίμηση του αποτυπώματος DM. Κατ' αρχάς, τα κυρίαρχα μεγέθη block κατανομής είναι λίγα (όπως οι περιπτωσιολογικές μελέτες μας δείχνουν, 6 ή 7 μπορούν να αποτελέσουν 70-80 % από τα συνολικά αιτήματα κατανομής), αλλά αυτά τα μεγέθη επιδεικνύουν μια μεγάλη παραλλαγή δεδομένου ότι μερικοί απ' αυτούς, μπορεί να είναι ακριβώς μερικά bytes, ενώ άλλες μπορούν να είναι αρκετά KB. Επομένως, οι κατάλληλοι διαχειριστές DM δεν είναι αρμόδιοι

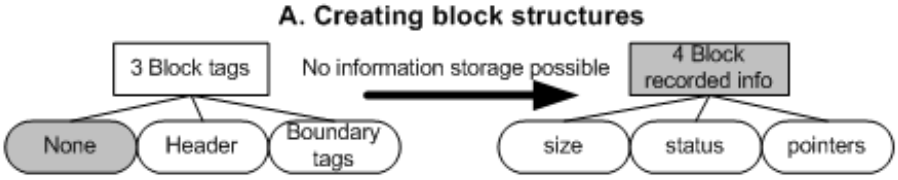
μόνο για τα μεγάλα ή πολύ μικρά μεγέθη, και οι συνδυασμοί λύσεων και για τους δύο τύπους κατανομών απαιτούνται σε αυτές τις εφαρμογές. Δεύτερον, το κύριο τμήμα σχεδίων, που καθορίζει το κυρίαρχο σχεδιασμό μη-κατανομής, πχ. κεκλιμένη ράμπα, αιχμές, οροπέδια, (δείτε [56] για περισσότερες λεπτομέρειες), δείχνει ότι πολύ συχνά τα δυναμικά πολυμέσα και οι ασύρματες εφαρμογές δικτύων κατέχουν τις πολύ ενεργές φάσεις, όπου λίγες δομές δεδομένων διατίθενται δυναμικά με έναν πολύ γρήγορο και μεταβλητό τρόπο (δηλαδή οξύνει), ενώ άλλες δομές δεδομένων αυξάνονται συνεχώς (δηλαδή κεκλιμένες ράμπες) και παραμένουν σταθερές σε ένα ορισμένο σημείο για μεγάλο διάστημα, προτού να ελευθερωθούν. Σύμφωνα με τις παρατηρήσεις μας, αυτές οι φάσεις δημιουργίας/καταστροφής του συνόλου των δομών δεδομένων με ορισμένα μεγέθη κατανομής επηρεάζονται συνήθως από τη λογική δομή των φάσεων που καθορίζονται από τους σχεδιαστές στον κώδικα εφαρμογής (πχ. που δίνει τις φάσεις) ή τα πρόσθετα γεγονότα (πχ. άφιξη των πακέτων στις ασύρματες εφαρμογές δικτύων). Τρίτον, το ποσοστό των εντατικών φάσεων αποεκχώρησης μνήμης καθορίζει πόσο συχνά πρέπει να αλλάξει τη δομή του διαχειριστή DM, για να εγκαταστήσει τα νέα μεγέθη αποεκχώρησης και το σχεδιασμό της δυναμικής εφαρμογής. Σε αυτήν την περίπτωση έχουμε παρατηρήσει επίσης τα πανομοιότυπα χαρακτηριστικά γνωρίσματα και στα δυναμικά πολυμέσα και στις ασύρματες εφαρμογές δικτύων, επειδή η συμπεριφορά χρόνου εκτέλεσης του συστήματος τείνει να ακολουθήσει μια προκαθορισμένη διάταξη για το πώς να χειριστεί τις διαφορετικές φάσεις και τα γεγονότα. Παραδείγματος χάριν, στα τρισδιάστατα παιχνίδια η ακολουθία για να συντηρήσει τα νέα γεγονότα, ως μετακίνηση της φωτογραφικής μηχανής από τους παίκτες (πχ. αναπροσαρμογή των ορατών αντικειμένων, απόδοση του νέου υποβάθρου, κλπ) καθορίζεται πάντα ανάλογα με τον τύπο αντικειμένου. Ομοίως, καθορίζεται και ο τρόπος να αντιμετωπιστεί η άφιξη των νέων πακέτων στις ασύρματες εφαρμογές δικτύων. Κατά συνέπεια, όλα αυτά τα προηγούμενα χαρακτηριστικά γνωρίσματα επιτρέπουν σε μας για να συγκεντρώσουμε αυτούς τους δύο αρχικά διαφορετικούς τομείς των εφαρμογών, εξετάζοντας τα κοινά χαρακτηριστικά DM τους και να καθορίσουν μια κοινή διάταξη για να προσπελαστεί ο χώρος σχεδιασμού.

8.3.3 Διάταξη των δέντρων για το μειωμένο αποτύπωμα μνήμης στα δυναμικά πολυμέσα και τις ασύρματες εφαρμογές δικτύων

Σε αυτήν την υποενότητα συζητάμε τη σφαιρική διάταξη μέσα στις ορθογώνιες κατηγορίες χώρου διαχείρισης του σχεδιασμού DM σύμφωνα με τους προαναφερθέντες παράγοντες της επιρροής για ένα μειωμένο αποτύπωμα μνήμης, για τα δυναμικά πολυμέσα και τις ασύρματες εφαρμογές δικτύων. Έχουμε

καθορίσει αυτήν την σφαιρική διάταξη μετά από την εκτενή δοκιμή (δηλαδή, περισσότερο από 6 μήνες πειραμάτων, με τις διαφορετικές διαταγές και εφαρμογές των διαχειριστών DM) χρησιμοποιώντας ένα αντιπροσωπευτικό σύνολο δέκα πραγματικών δυναμικών ενσωματωμένων πολυμέσων και ασύρματων εφαρμογών δικτύων, με διαφορετικά μεγέθη κώδικα (δηλαδή, από 1000 γραμμές έως 700 χιλιάδες γραμμές C++ κώδικα), συμπεριλαμβανομένου: εξελικτική τρισδιάστατη απόδοση όπως [92] ή MPEG 4 κωδικοποιητής (VTC) [40], οι τρισδιάστατοι αλγόριθμοι [38], τρισδιάστατα παιχνίδια [41], [42] και αποθήκευση, εφαρμογές [93] αναδημιουργίας εικόνας δικτύων και δρομολόγησης.

Η εμπειρία προτείνει ότι το πλήθος των περιπτώσεων του κατακερματισμού δεν μπορεί να αποφευχθεί, μόνο με μια κατάλληλη οργάνωση pool [56]. Τα πειράματά μας δείχνουν, ότι αυτό ισχύει ιδιαίτερα για τα ενσωματωμένα δυναμικά πολυμέσα και τις ασύρματες εφαρμογές δικτύων. Επομένως, οι κατηγορίες Δ και Ε τοποθετούνται στη σειρά απόφασης πριν από τις κατηγορίες Γ και Β. Η τελική σειρά είναι η ακόλουθη: το δέντρο Α2 τοποθετείται πρώτα για να καθορίσει εάν ένα ή περισσότερα μεγέθη block χρησιμοποιούνται και το Α5 είναι τοποθετημένο δεύτερο, για να αποφασίσει τη σφαιρική δομή των block. Έπειτα, οι κατηγορίες που εξετάζουν τον τεμαχισμό, δηλαδή, οι κατηγορίες Δ και Ε, αποφασίζονται επειδή, όπως έχουμε προαναφέρει, είναι σημαντικότερες από τις κατηγορίες που προσπαθούν να αποτρέψουν τον τεμαχισμό (δηλαδή, Γ και Β). Κατόπιν, το υπόλοιπο των γενικών εξόδων οργάνωσης πρέπει να αποφασιστεί για αυτά τα αιτήματα block. Κατά συνέπεια, αποφασίζονται τα υπόλοιπα δέντρα στην κατηγορία Α (δηλαδή, Α1, Α3 και Α4). Μετά, λαμβάνοντας υπόψη τις αλληλεξαρτήσεις, η τελική σφαιρική διάταξη είναι η ακόλουθη: Α2->Α5->Ε2->D2->E1->D1->B4->B1- >C1->A1->A3->A4.



Σχήμα 8.4: Παράδειγμα της σωστής διάταξης μεταξύ δύο ορθογώνιων δέντρων.

Εάν η διάταξη που προτείναμε πριν δεν ακολουθείται, οι περιττοί περιορισμοί διαδίδονται στα επόμενα δέντρα απόφασης. Ως εκ τούτου, οι καταλληλότερες αποφάσεις δεν μπορούν να ληφθούν στα υπόλοιπα ορθογώνια δέντρα. Ένα παράδειγμα αυτού παρουσιάζεται στην Εικόνα 8.4. Υποθέστε ότι η διάταξη ήταν Α3 και έπειτα Ε2 και D2. Με απόφαση του σωστού φύλλου για Α3, προφανής θα ήταν η επιλογή για να σώσει το αποτύπωμα μνήμης, να μην επιλεχτεί κανένα φύλλο, το οποίο δείχνει ότι δεν πρέπει να χρησιμοποιηθεί κανένας το-

μέας επιγραφών οποιουδήποτε τύπου. Αυτό φαίνεται λογικό εκ πρώτης όψεως επειδή οι τομείς επιγραφών, θα απαιτούσαν ένα σταθερό ποσό πρόσθετης μνήμης για κάθε block που πρόκειται να διατεθεί. Μετά από την απόφαση για κανένα φύλλο στο δέντρο A3, αποφασίζονται τα φύλλα που χρησιμοποιούνται για τα δέντρα E2 και D2. Τώρα, είμαστε υποχρεωμένοι για να επιλέξουμε το never φύλλο επειδή μετά διαδίδουμε τους περιορισμούς A3, τα block δεν μπορούν να είναι διαιρεμένα κατάλληλα ή συγχωνευμένα χωρίς αποθήκευση των πληροφοριών για το μέγεθός τους. Ως εκ τούτου, ο τελικός διαχειριστής DM χρησιμοποιεί τη λιγότερη μνήμη ανά block, αλλά δεν μπορεί να εξετάσει τον εσωτερικό ή εξωτερικό τεμαχισμό από τα διαιρεμένα ή συγχωνευμένα block. Αυτή η λύση θα μπορούσε να φανεί ως σωστή απόφαση για μια εφαρμογή όπου το κυρίαρχο μέγεθος block, καθορίζεται και έτσι κανένα σοβαρό αποτέλεσμα δεν υπάρχει λόγω του εσωτερικού ή εξωτερικού τεμαχισμού. Εντούτοις, εάν η εφαρμογή περιλάβει ένα μεταβλητό ποσό μεγεθών κατανομής (χαρακτηριστική συμπεριφορά σε πολλά δυναμικά πολυμέσα και ασύρματες εφαρμογές δικτύων), το πρόβλημα τεμαχισμού (εσωτερικό και εξωτερικό) πρόκειται να καταναλώσει περισσότερη μνήμη από τους πρόσθετους τομείς επιγραφών που απαιτούνται για τη συγχώνευση και το διαχωρισμό δεδομένου ότι τα ελευθερωμένα block δεν θα είναι σε θέση να επαναχρησιμοποιηθούν. Επομένως, είναι απαραίτητο να αποφασίσει για τα E2 και D2 δέντρα πρώτα, και να διαδοθούν έπειτα οι προκύπτοντες περιορισμοί στο δέντρο A3. Για να καταδειχτεί η ακρίβεια αυτής της εξήγησης, στις ακόλουθες παραγράφους παρουσιάζονται τα πειραματικά αποτελέσματα του αποτυπώματος μνήμης για μια εφαρμογή πολυμέσων με ένα ποικίλο κυρίαρχο μέγεθος block που χρησιμοποιεί κάθε μια από τις πιθανές διαταγές, δηλαδή ένας τρισδιάστατος αλγόριθμος με τα εξελικτικά πλέγματα [92]. Αυτός περιγράφεται αργότερα λεπτομερέστερα στις περιπτωσιολογικές μελέτες και τα πειραματικά αποτελέσματά μας 8.5. Τα αποτελέσματα που παρουσιάζονται είναι μέσες τιμές, μετά από 10 τρεξίματα για κάθε πείραμα και τις ομάδες 10 εκρήξεων. Καταρχήν, η εφαρμογή διαιρείται σε διάφορες εκρήξεις 1.000 (από)εκχωρήσεων με ένα περιορισμένο ποσό block (10 διαφορετικά μεγέθη που κυμαίνονται σε bytes από 20 μέχρι 500).

Πίνακας 8.1: Παράδειγμα δύο διαχειριστών DM με διαφορετική διάταξη στο χώρο σχεδίασης.

Διαχειριστές DM	Ίχνη Μνήμης (KB)	Χρόνος Εκτέλεσης
(1) A3-D2/E2	2.392*10^2	7.005
(2) D2/E2-A3	4.682*10	11.687
Σύγκριση 2-1	19.56	

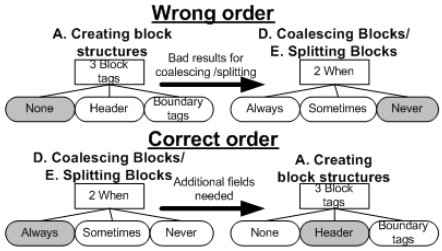
Αυτός είναι ένας χαρακτηριστικός τρόπος με τον οποίο μπορεί να τρέξει μια εφαρμογή πολυμέσων με διάφορες φάσεις. Στις εφαρμογές δικτύων η σειρά των

μεγεθών μπορεί να ποικίλει ακόμα περισσότερο, λόγω της αβεβαιότητας στα πακέτα φόρτου εργασίας και εισαγωγής. Έχουμε εφαρμόσει δύο διαφορετικές εκδόσεις του ίδιου διαχειριστή DM, οι οποίες περιλαμβάνουν μια ενιαία pool όπου τα ελεύθερα και χρησιμοποιημένα block συνδέονται διπλά μέσα σε έναν κατάλογο. Η μόνη διαφορά μεταξύ αυτών των δύο διαχειριστών DM είναι η προαναφερθείσα εξηγημένη διάταξη μεταξύ A3 και D2/E2.

Στην πρώτη λύση, αποφασίζεται το A3 και έπειτα D2 και E2. Ως εκ τούτου, καμία συνεχής και διαιρεμένη υπηρεσία δεν παρέχεται στο διαχειριστή DM, αλλά απαιτούνται και λιγότερο υπερυψωμένα block. Στη δεύτερη λύση αποφασίζονται πρώτα οι D2 και E2 και έπειτα καθορίζεται ο A3. Κατά συνέπεια, αυτός ο διαχειριστής DM περιλαμβάνει και τις δύο υπηρεσίες (δηλαδή, συγχώνευση και διαχωρισμό), αλλά περιλαμβάνει γενικά έξοδα 7 bytes ανά block στον κατάλογο ως πρόσθετη επιγραφή για τη συντήρηση: 2 bytes για το μέγεθος, κάθε σύνδεση 2 bytes (2 απαιτούνται) και 1 byte (δηλαδή, ένας τομέας bool) για να αποφασίσουν εάν χρησιμοποιείται. Τα αποτελέσματα που επιτυγχάνονται και στις δύο περιπτώσεις παρουσιάζονται στον Πίνακα 8.1. Παρουσιάζεται σαφώς ότι τα γενικά έξοδα των πρόσθετων επιγραφών είναι λιγότερο σημαντικά από τα γενικά έξοδα λόγω του τεμαχισμού, όπου απαιτείται, η 5 φορές λιγότερη μνήμη (δείτε τη σύγκριση 2-1 γραμμών στον Πίνακα 8.1). Επομένως, όπως προτείνουμε στη σφαιρική διάταξη εξερεύνησης μας για το μειωμένο αποτύπωμα μνήμης (δείτε το τμήμα 8.3.3 για περισσότερες λεπτομέρειες), ο δεύτερος διαχειριστής DM παράγει τα καλύτερα αποτελέσματα. Σε αυτόν τον δεύτερο διαχειριστή, πρώτα τα D2 και E2 δέντρα αποφασίζονται και έπειτα χρησιμοποιούνται οι περιορισμοί τους για να επιλέξουν το A3 φύλλο δέντρων. Επίσης σημειώστε τη διαφορά στο χρόνο εκτέλεσης: ο πρώτος διαχειριστής DM έχει έναν πολύ χαμηλό συνολικό χρόνο εκτέλεσης, επειδή δεν παρέχει τις συγχωνευμένες ή διαιρεμένες υπηρεσίες, ενώ ο δεύτερος έχει πρόσθετα γενικά έξοδα στο χρόνο εκτέλεσης 66.83%. Αυτό δείχνει ότι η απαίτηση απόδοσης του συστήματος υπό ανάπτυξη, πρέπει να λάβει υπόψη όταν αποφασίζονται οι επιλογές στο χώρο σχεδιασμού. Ως εκ τούτου, εάν ο σχεδιαστής συστημάτων απαιτεί ένα ορισμένο επίπεδο απόδοσης, το οποίο δεν επιτυγχάνεται από την ακραία μειωμένη λύση αποτυπώματος μνήμης που παρουσιάζεται στον αριθμό 2, μια άλλη λύση μπορεί να σχεδιαστεί πιο κοντά στη λύση με αριθμό 1 (αλλά με μια ισορροπημένη ανταλλαγή μεταξύ του αποτυπώματος μνήμης και της απόδοσης). Όπως έχει αναφερθεί πιο πριν, έχουμε εκτελέσει ένα μεγάλο ποσό παρόμοιων πειραμάτων για τις υπόλοιπες και παραγόμενες διαταγές απόφασης που προτίθενται. Το κύριο συμπέρασμα από τις μελέτες μας είναι ότι με την προτεινόμενη σφαιρική διάταξη, η μεθοδολογία εξερεύνησης δεν είναι απαραίτητα περιορισμένη και μόνο οι απαραίτητοι περιορισμοί διαβιβάζονται στα επόμενα δέντρα απόφασης, η οποία δεν είναι δυνατή με άλλες διατάξεις για αυτούς τους τύπους δυναμικών εφαρμογών με τον μη περιορισμό του μεγέθους αυτών.

8.4 Επισκόπηση της σφαιρικής ροής στη δυναμική διαχειριστική μεθοδολογία μνήμης

Ο κύριος στόχος της προτεινόμενης μεθοδολογίας σχεδιασμού DM διαχείρισης, είναι να παρασχεθούν στους υπεύθυνους για την ανάπτυξη μια ροή σχεδιασμού των εξατομικευμένων διαχειριστών DM για τα δυναμικά πολυμέσα και τα ασύρματα συστήματα δικτύων. Αυτή η ροή σχεδιασμού διαιρείται σε τέσσερις κύριες φάσεις, όπως υποδεικνύεται στην Εικόνα 8.5. Η πρώτη φάση της μεθοδολογίας λαμβάνει και ταξινομεί τις αναλυτικές πληροφορίες για το DM και τον χρόνο με τον οποίο εκτελείται κάθε δυναμικό αποεκχώρησης, όπως απεικονίζεται στην πρώτη ωοειδή μορφή της Εικόνας 8.5. Αυτή η φάση είναι βασισμένη στη σκιαγράφηση της χρήσης του DM στην εξεταζόμενη εφαρμογή επειδή, σύμφωνα με την εμπειρία μας, είναι η μόνη ρεαλιστική επιλογή για τα σημερινά δυναμικά πολυμέσα και τις ασύρματες εφαρμογές [91], [77] δικτύων. Η σκιαγράφηση πραγματοποιείται ως εξής. Κατ' αρχάς, παρεμβάλλουμε στον κώδικα της εφαρμογής υψηλού επιπέδου σχεδιάζοντας το περίγραμμα πλαισίου μας, το οποίο είναι βασισμένο στη βιβλιοθήκη C++ της σκιαγράφησης των αντικειμένων και των πρόσθετων γραφικών εργαλείων [77]. Κατόπιν, σχεδιάζουμε αυτόματα το περίγραμμα ένα αντιπροσωπευτικό σύνολο περιπτώσεων εισαγωγής της εφαρμογής, γενικά μεταξύ 10 και 15 διαφορετικών εισαγωγών, συμπεριλαμβανομένων των ακραίων περιπτώσεων min και max.



Σχήμα 8.5: Σφαιρική επισκόπηση της προτεινόμενης ροής μεθοδολογίας σχεδιασμού διαχείρισης DM.

Στο αποτύπωμα μνήμης μεθοδολογίας, που αντιπροσωπεύεται ως δεύτερη ορθογώνια μορφή της Εικόνας 8.5, χαρακτηρίζεται η συμπεριφορά DM της εφαρμογής. Οι φάσεις κατανομής προσδιορίζονται και συνδέονται με τις λογικές φάσεις της εφαρμογής (πχ. ψαλίδισμα, φωτισμός, κλπ σε τρισδιάστατη εμφάνιση [94]) που χρησιμοποιούν τα εργαλεία μας, για να αναλύσουν τις πληροφορίες σχεδιασμού περιγράμματος που λαμβάνονται στην προηγούμενη φάση. Αυτά τα εργαλεία προσδιορίζουν τις κύριες παραλλαγές στις διαφορετικές γραφικές παραστάσεις και τους πίνακες και τα δυναμικά διατιθέμενα με-

γέθη. Κατόπιν, το χώρο σχεδιασμού για τη διαχείριση DM και τη διάταξη εξερεύνησης για τα δυναμικά πολυμέσα και τις ασύρματες εφαρμογές δικτύων, που προτείνονται χρησιμοποιώντας στις Ενότητες 8.2 και 8.3, αντίστοιχα, κατάλληλα σύνολα υποψηφίων εξατομικευμένων διαχειριστών DM οι οποίοι επιλέγονται για κάθε αίτηση. Αυτή η δεύτερη φάση μπορεί να πάρει μέχρι μια εβδομάδα για τις πολύ σύνθετες εφαρμογές, δηλαδή με περισσότερες από 20 φάσεις εκχώρησης στην ίδια εφαρμογή. Έπειτα, στην τρίτη φάση της μεθοδολογίας διαχείρισης DM, εφαρμόζουμε τους υποψηφίους διαχειριστές DM και διερευνάμε εξαντλητικά τη συμπεριφορά χρόνου εκτέλεσής τους στην αίτηση κάτω από τη μελέτη, όπως η κατανάλωση αποτυπώματος μνήμης, τεμαχισμού, απόδοσης και ενέργειας. Για αυτόν το λόγο, όπως έχουμε εξηγήσει στο Τμήμα 8.2.3, έχουμε αναπτύξει μια βελτιστοποιημένη C++ βιβλιοθήκη, που καλύπτει όλες τις αποφάσεις στο χώρο σχεδιασμού μας και επιτρέπει την κατασκευή και τη σκιαγράφηση των εφαρμογών εξατομικευμένων διαχειριστών DM με έναν απλό τρόπο μέσω της σύνθεσης C++ των στρωμάτων [91]. Κατά συνέπεια, ολόκληρη η εξερεύνηση εφαρμογής γίνεται αυτόματα, χρησιμοποιώντας την προαναφερθείσα βιβλιοθήκη και τα πρόσθετα εργαλεία, η οποία εκτελεί τους πολλαπλάσιους χρόνους για την εφαρμογή, με τους διαφορετικούς DM διαχειριστές για να αποκτηθεί ο πλήρης χρόνος εκτέλεσης που σχεδιάζει το περίγραμμα και τις πληροφορίες. Αυτή η τρίτη φάση διαρκεί μεταξύ 3 και 4 ημερών στην περίπτωση σύνθετων εφαρμογών. Στην τέταρτη και τελική φάση της ροής σχεδιασμού διαχείρισης DM, χρησιμοποιούμε τα εργαλεία μας για να αξιολογήσουμε αυτόματα τη σκιαγράφηση που παράγεται στην προηγούμενη φάση και για να καθορίσουμε τα τελικά χαρακτηριστικά γνωρίσματα της εφαρμογής του εξατομικευμένου διαχειριστή DM για την εφαρμογή, όπως ο τελικός αριθμός ανώτατων μεγεθών block και ο αριθμός μεγεθών μνήμης εκχώρησης. Αυτή η τέταρτη φάση διαρκεί μεταξύ 2 ή 3 ημερών. Συνολικά, ολόκληρη η τυπική ροή για τους διαχειριστές DM απαιτεί μεταξύ 2 και 3 εβδομάδων για τις πραγματικές εφαρμογές.

8.5 Περιπτωσιολογικές μελέτες και πειραματικά αποτελέσματα

Έχουμε εφαρμόσει την προτεινόμενη μεθοδολογία σε τρεις ρεαλιστικές περιπτωσιολογικές μελέτες που αντιπροσωπεύουν τα διαφορετικά πολυμέσα και τις περιοχές εφαρμογής δικτύων: η πρώτη περιπτωσιολογική μελέτη είναι ένα τρισδιάστατο σύστημα αναδημιουργίας εικόνας και ο τρίτος είναι ένα τρισδιάστατο δεδομένο σύστημα βασισμένο στα εξελικτικά πλέγματα. Στις ακόλουθες υποενότητες περιγράφουμε εν συντομία, τη συμπεριφορά των τριών περιπτω-

σιολογικών μελετών και τη προτεινόμενη μεθοδολογία που εφαρμόζεται για να σχεδιάσει τους εξατομικευμένους διαχειριστές που ελαχιστοποιούν το αποτύπωμα μνήμης. Όλα τα αποτελέσματα που παρουσιάζονται είναι οι μέσες τιμές μετά από ένα σύνολο 10 προσομοιώσεων για κάθε εφαρμογή διαχειριστών που χρησιμοποιεί 10 πρόσθετες πραγματικές εισαγωγές από αυτές που χρησιμοποιούνται στο σχεδιάγραμμα και σχεδιάζουν τους εξατομικευμένους διαχειριστές DM. Όλες οι τελικές τιμές στις προσομοιώσεις ήταν πανομοιότυπες (παραλλαγές λιγότερες από 2%).

8.5.1 Μέθοδος που εφαρμόζεται σε ένα προγραμματισμένο δίκτυο εφαρμογής

Η πρώτη περιπτωσιολογική μελέτη που παρουσιάζεται είναι το έλλειμμα γύρω από την εφαρμογή του Robin (DRR) που λαμβάνεται από τη NetBench αξιολογώντας την ακολουθία [93]. Είναι ένας αλγόριθμος που εφαρμόζεται σε πολλούς δρομολογητές. Στην πραγματικότητα, οι παραλλαγές του DRR αλγορίθμου χρησιμοποιήθηκαν από το Cisco για τα εμπορικά προϊόντα του σημείου πρόσβασης και εν τέλει στις ευρυζωνικές συσκευές πρόσβασης. Χρησιμοποιώντας τον αλγόριθμο DRR ο δρομολογητής προσπαθεί να ολοκληρώσει μια έκθεση σχεδιάζοντας με την άδεια του ίδιου ποσού στοιχείων, για να περαστεί και να σταλεί από κάθε εσωτερική σειρά αναμονής. Στον αλγόριθμο DRR, ο χρονοπρογραμματιστής επισκέπτεται κάθε εσωτερική σειρά αναμονής με αντικείμενα, αυξάνοντας το μεταβλητό «έλλειμμα» από την αξία «κβάντο» και καθορίζει τον αριθμό bytes στο πακέτο στην κορυφή της σειράς αναμονής. Εάν το μεταβλητό «έλλειμμα» είναι λιγότερο από το μέγεθος του πακέτου στην κορυφή της σειράς αναμονής (δηλαδή, δεν έχει αρκετές πιστώσεις αυτήν τη στιγμή), ο χρονοπρογραμματιστής κινείται για να συντηρηθεί η επόμενη σειρά αναμονής. Εάν το μέγεθος του πακέτου στην κορυφή της σειράς αναμονής είναι λιγότερο ή ίσο προς το μεταβλητό «έλλειμμα», τότε εκείνη η μεταβλητή μειώνεται από τον αριθμό bytes στο πακέτο και το πακέτο διαβιβάζεται στο λιμένα παραγωγής. Ο χρονοπρογραμματιστής συνεχίζει αυτήν την διαδικασία, που αρχίζει από την πρώτη σειρά αναμονής, κάθε φορά που διαβιβάζεται ένα πακέτο. Εάν μια σειρά αναμονής δεν έχει άλλα πακέτα, καταστρέφεται. Τα πακέτα άφιξης περιμένουν στη σειρά στον κατάλληλο κόμβο, εάν κανένας τέτοιος κόμβος δεν υπάρχει, δημιουργείται. Έχουν χρησιμοποιηθεί δέκα πραγματικά ίχνη κυκλοφορίας δικτύων διαδικτύου μέχρι 10MB/sec [95] για να τρέξουν τις ρεαλιστικές προσομοιώσεις DRR.

Για να δημιουργήσουμε τον εξατομικευμένο διαχειριστή DM έχουμε ακολουθήσει βήμα προς βήμα τη μεθοδολογία μας. Κατά συνέπεια, προκειμένου να καθοριστούν οι λογικές φάσεις της εφαρμογής και του ατομικού διαχειριστή

DM, σχεδιάζουμε αρχικά το περίγραμμα της συμπεριφοράς DM με τις δυναμικές δομές δεδομένων μας, που σχεδιάζουν το περίγραμμα κατά προσέγγιση [77]. Κατόπιν, εφαρμόζουμε την εξερεύνηση σχεδιασμού της διαχείρισης DM. Κατ' αρχάς, λαμβάνουμε την απόφαση στο δέντρο A2 (μεγέθη block), να έχουμε δηλαδή πολλά μεγέθη block για να αποτρέψουμε τον εσωτερικό τεμαχισμό. Αυτό γίνεται επειδή τα block μνήμης που ζητούνται από την εφαρμογή DRR, ποικίλλουν πολύ στο μέγεθος (για να αποθηκεύσουν τα πακέτα των διαφορετικών μεγεθών) και εάν μόνο ένα μέγεθος block χρησιμοποιείται για όλα τα ζητούμενα διαφορετικά μεγέθη block, τότε ο εσωτερικός θρυμματισμός θα μπορούσε να είναι μεγάλος. Κατόπιν, στο δέντρο A5 (εύκαμπτος διαχειριστής block μεγέθους) επιλέγουμε να χωρίσουμε ή να συγχωνεύσουμε, έτσι ώστε κάθε φορά που ζητείται ένα block μνήμης με ένα μεγαλύτερο ή μικρότερο μέγεθος από τον τρέχοντα block, να επικαλούνται οι συγχωνευμένοι ή διαιρεμένοι μηχανισμοί. Στα δέντρα E2 και D2 (when) το επιλέγουμε πάντα, επιπλέον η δοκιμή ανασυγκροτείται μόλις εμφανίζεται ο τεμαχισμός. Κατόπιν, στα δέντρα E1 και D1 (αριθμός ανώτατου/ελάχιστου μεγέθους block), επιλέγουμε πολλές κατηγορίες και σταθερές, επειδή θέλουμε να πάρουμε τη μέγιστη επίδραση από τους συγχωνευμένους και διαιρεμένους μηχανισμούς με τον μη περιορισμό του μεγέθους αυτών των νέων block. Μετά από αυτό, στα δέντρα B1 (το τμήμα pool βάσει μεγέθους) και B2 (δομή pool), επιλέγεται η απλούστερη πιθανή εφαρμογή pool, η οποία είναι μια ενιαία pool, επειδή εάν δεν υπάρχει κανένα σταθερό μέγεθος block, κατόπιν καμία πραγματική χρήση δεν υπάρχει για τις σύνθετες δομές pool, για να επιτύχει ένα μειωμένο αποτύπωμα μνήμης.

Κατόπιν, στο δέντρο C1 (κατάλληλοι αλγόριθμοι), επιλέγουμε την ακριβή τακτοποίηση, για να αποφύγουμε όσο το δυνατόν περισσότερο την απώλεια της μνήμης στον εσωτερικό τεμαχισμό. Έπειτα, στο δέντρο A1 (δομή block), επιλέγουμε το απλούστερο DDT που επιτρέπεται, το οποίο είναι ένας διπλά συνδεόμενος κατάλογος. Κατόπιν, στα δέντρα A3 (ετικέτες block) και A4 (το block που κατέγραψε τις πληροφορίες), επιλέγουμε έναν τομέα επιγραφών για να προσαρμόσουμε τις πληροφορίες για το μέγεθος και τη θέση κάθε φραγμού για να υποστηρίξουμε τους διαιρεμένους και συγχωνευμένους μηχανισμούς. Τέλος, μετά από αυτό, πρέπει να παρθούν αυτές οι αποφάσεις μετά από τη διάταξη που περιγράφεται στο τμήμα 8.3, σύμφωνα με την προτεινόμενη ροή σχεδιασμού μπορούμε να καθορίσουμε εκείνες τις αποφάσεις του τελικού εξατομικευμένου διαχειριστή DM, που εξαρτάται από την ιδιαίτερη συμπεριφορά χρόνου εκτέλεσής της στην εφαρμογή (πχ. τελικός αριθμός ανώτατων μεγεθών block) μέσω της προσομοίωσης με την εξατομικευμένη C++ βιβλιοθήκη και τα εργαλεία μας [91] (δείτε στο τμήμα 8.4 για περισσότερες λεπτομέρειες). Κατόπιν, το εφαρμόζουμε και συγκρίνουμε την εξατομικευμένη λύση με τους πολύ γνωστούς βελτιστοποιημένης κατάστασης προόδου, γενικής χρήσης διαχειριστές, δηλαδή Lea v2.7.2 [96] και Kingsley [56]. Ο διαθέτης Lea είναι ένας από τους καλύτερους

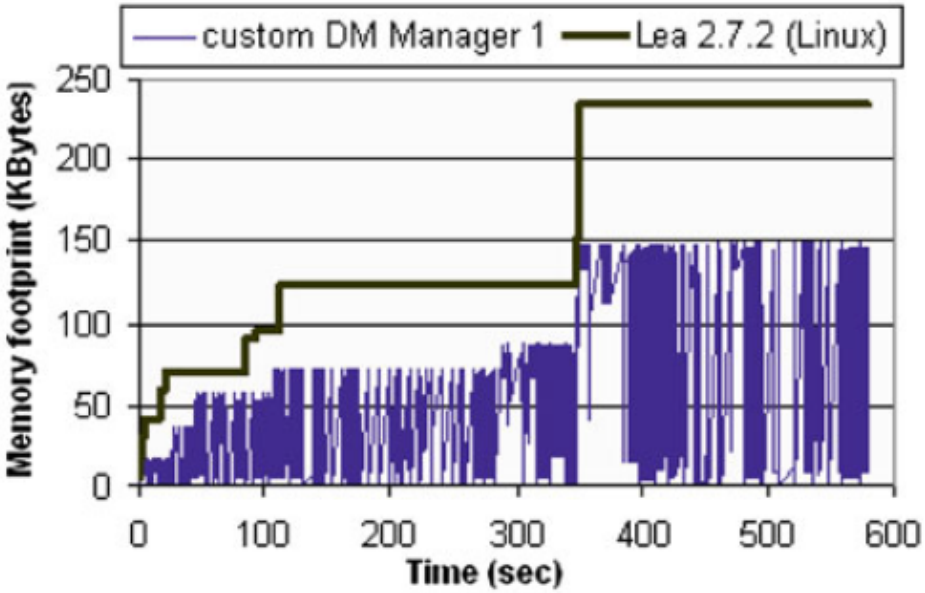
γενικούς διαχειριστές (σε επίπεδο του συνδυασμού αποτυπώματος ταχύτητας και μνήμης) [79] και διάφορες παραλλαγές είναι ενσωματωμένες στις διαφορετικές διανομές του GNU Linux ΛΣ. Είναι ένας υβριδικός διαχειριστής DM, που περιλαμβάνει τις διαφορετικές συμπεριφορές για τα διαφορετικά μεγέθη αντικειμένου. Για τα μικρά αντικείμενα χρησιμοποιεί κάποιους γρήγορους καταλόγους [56], για τα μέσου μεγέθους αντικείμενα εκτελεί την κατά προσέγγιση καλή εφαρμογή κατανομής [56] και για τα μεγάλα αντικείμενα χρησιμοποιεί την αφιερωμένη μνήμη (που διατίθεται άμεσα με τη λειτουργία mmap ()). Επίσης, συγκρίνουμε την προσέγγισή μας με μια βελτιστοποιημένη έκδοση του Kingsley [56] DM διαχειριστή, που χρησιμοποιεί ακόμα τη δύναμη να διαχωρίζει δύο κατάλληλους καταλόγους [56] για να επιτύχει τις γρήγορες κατανομές, αλλά βελτιστοποιείται για αντικείμενα μεγαλύτερα από 1.024 bytes, τα οποία παίρνουν τη μνήμη τους από έναν ταξινομημένο συνδεδεμένο κατάλογο, θυσιάζοντας οριστικά την τακτοποίηση ταχύτητας. Μια παρόμοια τεχνική εφαρμογής χρησιμοποιείται αρκετά βασισμένη στα παράθυρα ΛΣ [80] [81].

Επιπλέον, έχουμε συγκρίνει την εξατομικευμένη λύση DM με μια σχεδιασμένη χειρωνακτικά εφαρμογή των σημασιολογικών περιοχών διαχειριστών [82] που μπορούν να βρεθούν σε κάποιο ενσωματωμένο ΛΣ (πχ. [78] Στην Εικόνα 8.6 παρουσιάζονται οι χρήσεις των εξατομικευμένων διαχειριστών DM με λιγότερη μνήμη από το Lea 2.7.2 (Linux ΛΣ), το Kingsley και τους διαχειριστές DM περιοχών. Αυτό οφείλεται στο γεγονός ότι ο εξατομικευμένος διαχειριστής DM δεν έχει καθορίσει τα μεγέθη και προσπαθεί να συγχωνευτεί και να χωρέσει όσο το δυνατόν περισσότερο, το οποίο είναι μια καλύτερη επιλογή στις δυναμικές εφαρμογές με τα μεγέθη και με τη μεγάλη παραλλαγή. Επιπλέον, δεν χρησιμοποιείται όταν τεράστια κομμάτια της μνήμης συγχωνεύονται, και έτσι επιστρέφονται πίσω στο σύστημα για άλλες εφαρμογές. Το Lea και το Kingsley δημιουργούν τους τεράστιους ελεύθερους καταλόγους αχρησιμοποίητων block (σε περίπτωση που επαναχρησιμοποιούνται αργότερα), συγχωνεύονται και χωρίζουν σπάνια (Lea) ή ποτέ (Kingsley) και τελικά, έχουν καθορίσει τα μεγέθη block. Αυτό μπορεί να παρατηρηθεί στην Εικόνα 8.7, όπου παρουσιάζουμε γραφικές παραστάσεις χρήσης DM των εξατομικευμένων διαχειριστών DM και Lea, και Εικόνα 8.8, όπου τα αιτήματα DM της εφαρμογής παρουσιάζονται κατά τη διάρκεια που οργανώνεται το ένα εκ των δύο. Στην περίπτωση του διαχειριστή DM περιοχών, καταναλώνεται περισσότερο αποτύπωμα μνήμης από το Lea λόγω του τεμαχισμού, δεδομένου ότι κανένας συγχωνευμένος/διαιρεμένος μηχανισμός δεν εφαρμόζεται για να επαναχρησιμοποιήσει τα block μνήμης.

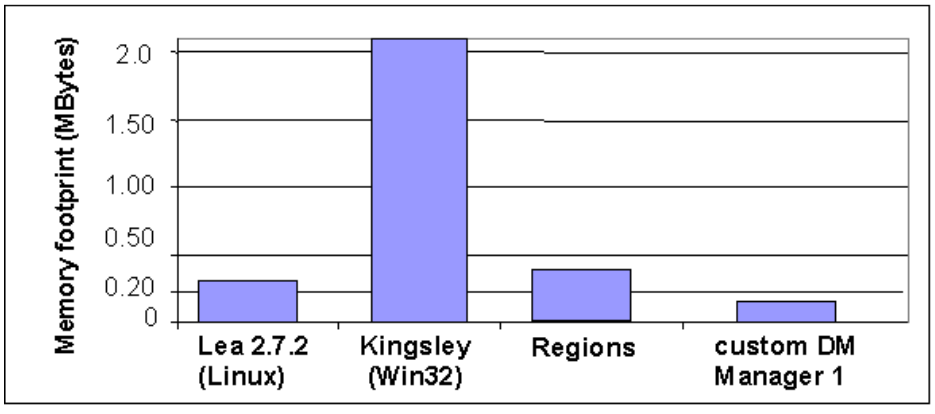
Σχετικά με την απόδοση των διαφόρων διαχειριστών DM (δείτε τον Πίνακα 8.2 για περαιτέρω περίληψη με τα αποτελέσματα όλων των περιπτωσιολογικών μελετών), μπορούμε να παρατηρήσουμε ότι σε όλες τις περιπτώσεις 600 το δευτερόλεπτο, είναι ο χρόνος που χρησιμοποιείται για να σχεδιάσει τα προγραμματισμένα ίχνη των 600 δευτερολέπτων στην κίνηση του διαδικτύου. Αυτό

DM manager	memory footprint (KBytes)	execution time (secs)
(1) [illegible]	2.392×10^2	7.005
(2) [illegible]	4.692×10^1	11.687
Comparison 2-1	19.56%	166.83%

Σχήμα 8.6: Μέγιστα αποτελέσματα αποτυπώματος μνήμης στην εφαρμογή DRR.



Σχήμα 8.7: Συμπεριφορά του αποτυπώματος μνήμης του Lea διευθυντή DM έναντι του εξατομικευμένου διαχειριστή DM.



Σχήμα 8.8: Αιτήματα εκχώρησης DM στην εφαρμογή DRR.

σημαίνει ότι όλοι οι διαχειριστές DM που μελετώνται ικανοποιούν σε πραγματικό χρόνο τις απαιτήσεις της εφαρμογής DRR. Επιπλέον, ο διαχειριστής DM βελτιώνει εκτενώς το αποτύπωμα μνήμης που χρησιμοποιείται από τους άλλους διαχειριστές DM, αλλά κανένα πρόσθετο χρονικό γενικό έξοδο δεν παρατηρείται, λόγω των εσωτερικών διαδικασιών συντήρησής του. Τέλος, έχουμε αξιολογήσει τη συνολική κατανάλωση ενέργειας του τελικού ενσωματωμένου συστήματος με κάθε έναν από τους μελετημένους διαχειριστές DM, χρησιμοποιώντας έναν ακριβή κυκλικό βραχίονα βασισμένο στον προσομοιωτή που περιλαμβάνει ένα πλήρες πρότυπο εκτίμησης ενέργειας–καθυστέρησης για τεχνολογία τρανζίστορ 0.13 μm [97]. Τα αποτελέσματα δείχνουν ότι ο εξατομικευμένος διαχειριστής DM επιτυγχάνει τα πολύ καλά αποτελέσματα για την ενέργεια, όταν συγκρίνεται με τους διαχειριστές DM της κατάστασης προόδου. Αυτό οφείλεται στο γεγονός ότι οι περισσότερες από τις προσβάσεις μνήμης αποδίδονται εσωτερικά από τους διαχειριστές στις σύνθετες δομές διαχείρισης και δεν απαιτούνται στον εξατομικευμένο διαχειριστή, το οποίο χρησιμοποιεί τις απλούστερες εσωτερικές δομές δεδομένων που βελτιστοποιούνται για την εφαρμογή στόχων. Κατά συνέπεια, ο διαχειριστής DM μειώνει τις τιμές κατανάλωσης ενέργειας από 12%, ενώ το Kingsley, και το Lea κατά 15% και 16% αντίστοιχα (Εικόνα 8.9). Είναι σημαντικό να αναφερθεί ότι ακόμα κι αν ο Kingsley έχει ένα μικρότερο ποσό διαχειριστικών προσβάσεων DM, δεδομένου ότι δεν εκτελεί τις διαιρεμένες ή συγχωνευμένες διαδικασίες, αυτό πάσχει από μια μεγάλη ποινική ρήτρα του αποτυπώματος μνήμης. Αυτό μεταφράζεται στις πολύ ακριβές προσβάσεις μνήμης, επειδή πρέπει να χρησιμοποιηθούν οι μεγαλύτερες μνήμες. Συνεπώς, για τις δυναμικές εφαρμογές δικτύων όπως DRR, η μεθοδολογία μας επιτρέπει να σχεδιάσουμε τους πολύ προσαρμοσμένους διαχειριστές DM που εκθέτουν το λιγότερο τεμαχισμό από το Lea, οι περιοχές ή Kingsley και απαιτούν έτσι τη λιγότερη μνήμη. Δεδομένου ότι αυτή η μείωση στο αποτύπωμα μνήμης συνδυάζεται με μια απλούστερη εσωτερική διαχείριση του DM, το τελικό σύστημα καταναλώνει λιγότερη ενέργεια.

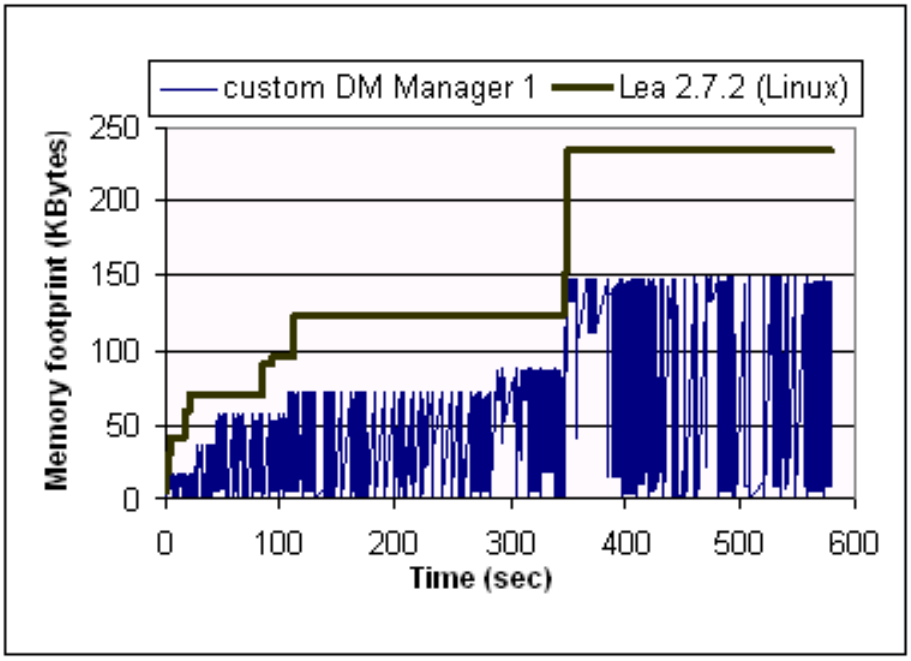
8.5.2 Εφαρμογή της μεθοδολογίας μας σε ένα τρισδιάστατο σύστημα αναδημιουργίας εικόνας

Οι δεύτερες μορφές περιπτωσιολογικής μελέτης, όπου ένας από τους υποαλγορίθμους αναδημιουργίας μιας τρισδιάστατης εφαρμογής [38] που λειτουργεί όπως την τρισδιάστατη αντίληψη στα ζωντανά όντα, όπου η σχετική μετατόπιση μεταξύ διάφορων 2D προβολών χρησιμοποιείται για να αναδημιουργήσει τη 3D διάσταση. Η ενότητα λογισμικού που χρησιμοποιείται ως αίτηση των οδηγών μας, είναι μια από τις βασικές δομικές μονάδες σε πολλούς τρέχοντες τρισδιάστατους αλγορίθμους οράματος: επιλογή χαρακτηριστικών γνωρισμά-

Πίνακας 8.2: Μέγιστο αποτύπωμα μνήμης, ο χρόνος εκτέλεσης και η κατανάλωση ενέργειας του κάθε Διαχειριστή DM στις περιπτωσιολογικές μελέτες μας

Δυναμικοί Διαχειριστές Μνήμης	Διαφοροποιημένος DRR	ανοικοδόμηση 3D εικόνας	εξελικτική απόδοση 3D
Kingsley-Windows (Bytes)	2.09×10^6	2.26×10^6	3.96×10^6
Χρόνος εκτέλεσης (s)	600	1.52	6.05
Κατανάλωση ενέργειας	7.98×10^9	4.04×10^6	11.22×10^6
Lea-Linux (Bytes)	2.34×10^5	2.11×10^6	1.86×10^6
Χρόνος εκτέλεσης (s)	600	2.01	8.12
Κατανάλωση ενέργειας(nJ)	8.18×10^9	4.11×10^6	11.43×10^6
περιφέρειες(bytes)	2.47×10^5	2.08×10^6	2.03×10^6
Χρόνος εκτέλεσης (s)	600	1.87	7.03
Κατανάλωση ενέργειας(nJ)	8.25×10^9	4.01×10^6	11.61×10^6
Obstacks (Bytes)	–	–	1.55×10^6
Χρόνος εκτέλεσης (s)	–	–	6.55
Κατανάλωση ενέργειας(nJ)	–	–	10.93×10^6
ο DM διαχειριστής μας (bytes)	1.48×10^5		1.07×10^6
Χρόνος εκτέλεσης (s)	600		6.91
Κατανάλωση ενέργειας(nJ)	7.11×10^9		9.67×10^6

των και συσχέτιση. Έχει εξαχθεί από τον αρχικό κώδικα του τρισδιάστατου συστήματος αναδημιουργίας εικόνας (δείτε [45] για τον πλήρη κώδικα του αλγορίθμου με 1.75 εκατομμύρια γραμμές υψηλού επιπέδου C++), και δημιουργεί τη μαθηματική αφαίρεση από τα σχετικά πλαίσια που χρησιμοποιείται στο σφαιρικό αλγόριθμο. Περιλαμβάνει ακόμα 600.000 γραμμές C++ κώδικα, ο οποίος καταδεικνύει την πολυπλοκότητα των εφαρμογών που μπορούμε να εξετάσουμε την προσέγγισή μας και την ανάγκη της υποστήριξης εργαλείων για τις φάσεις παραγωγής ανάλυσης και της εξερεύνησης του κώδικα στη γενική προσέγγισή μας (δείτε την Εικόνα 8.4).



Σχήμα 8.9: Σχετική ενεργειακή βελτίωση του εξατομικευμένου διαχειριστή DM στην εφαρμογή DRR.

Αυτή η εφαρμογή ταιριάζει με τις γωνίες [38] που ανιχνεύονται σε 2 επόμενα πλαίσια και στηρίζεται σε μεγάλο ποσοστό στο DM, λόγω της μη προβλεψιμότητας των χαρακτηριστικών γνωρισμάτων των εικόνων εισαγωγής στον χρόνο σύνταξης (πχ. ο αριθμός πιθανών γωνιών που ταιριάζουν ποικίλλει σε κάθε εικόνα). Επιπλέον, οι διαδικασίες γίνονται στις εικόνες με εντατικές ανάγκες σε μνήμη. Παραδείγματος χάριν, κάθε εικόνα με ένα μέγεθος 640 × 480 pixels πέρα από 1MB. Επομένως, τα γενικά έξοδα DM (πχ. ο εσωτερικός και εξωτερικός τεμαχισμός [56]) αυτής της εφαρμογής πρέπει να ελαχιστοποιηθούν για να είναι χρησιμοποιήσιμα για τις ενσωματωμένες συσκευές, όπου τρέχουν ταυτόχρονα περισσότερες εφαρμογές. Τέλος, σημειώστε ότι οι προσβάσεις του αλγορίθμου

στις εικόνες τυχαιοποιούνται. Αυτό έχει ως συνέπεια ότι οι κλασικές βελτιστοποιήσεις πρόσβασης εικόνας δε μπορούν να μειώσουν περαιτέρω το αποτύπωμα DM. Για αυτήν την περιπτωσιολογική μελέτη, η δυναμική συμπεριφορά της δείχνει, ότι η εφαρμογή παρουσιάζει ελάχιστη ποικιλία σε δυναμικούς τύπους [77] και συγκεκριμένα 8 διαφορετικά μεγέθη κατανομής.

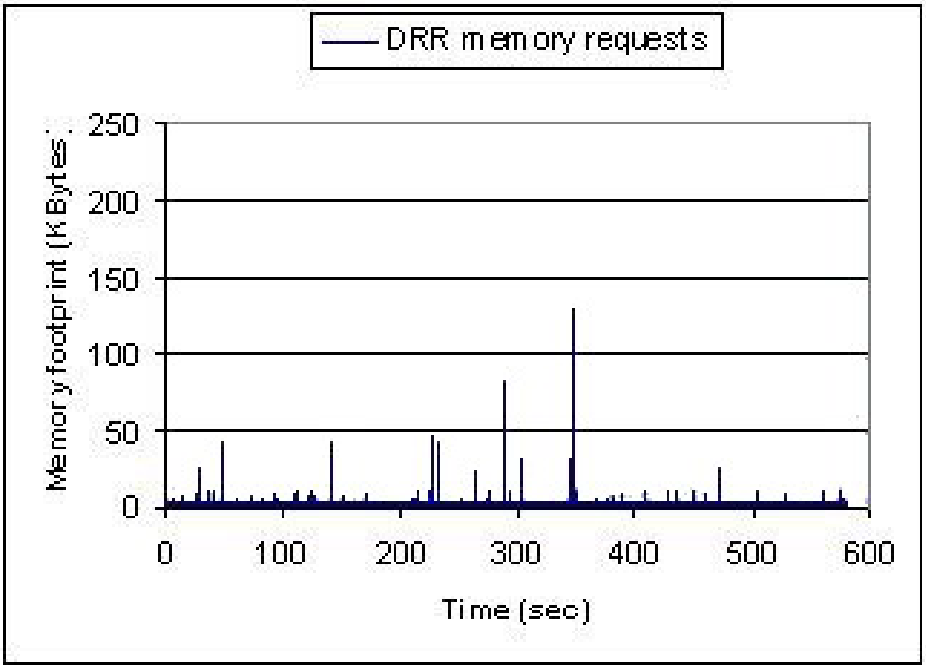
Επιπλέον, τα περισσότερα από αυτά τα διατιθέμενα μεγέθη είναι σχετικά μικρά (δηλαδή, μεταξύ 32 ή 16 Bytes) και μόνο πολύ λίγα block είναι πολύ μεγαλύτερα (πχ. 163KB). Επιπλέον, βλέπουμε ότι οι περισσότεροι από τους τύπους στοιχείων αλληλεπιδρούν ο ένας με τον άλλον και είναι ενεργοί σχεδόν σε όλο το χρόνο εκτέλεσης της εφαρμογής. Μέσα σε αυτό το πλαίσιο, εφαρμόζουμε τη μεθοδολογία μας και χρησιμοποιούμε τη διάταξη που παρέχεται στο Τμήμα 8.3 και προσπαθούμε να ελαχιστοποιήσουμε την απώλεια του αποτυπώματος DM (πχ. τεμαχισμός, από πάνω στις επιγραφές, κλπ) αυτής της εφαρμογής. Σαν αποτέλεσμα, λαμβάνουμε μια τελική λύση που αποτελείται από έναν εξατομικευμένο διαχειριστή DM με 4 διαχωρισμένες pool ή τις περιοχές για τα σχετικά μεγέθη στην εφαρμογή. Η πρώτη pool χρησιμοποιείται για το μικρότερο μέγεθος κατανομής που ζητείται στην εφαρμογή, δηλαδή 32 bytes. Η δεύτερη pool επιτρέπει τις κατανομές των μεγεθών μεταξύ 756 bytes και 1.024 bytes. Κατόπιν, η τρίτη pool χρησιμοποιείται για τα αιτήματα κατανομής 16.384 bytes. Τέλος, η τέταρτη pool χρησιμοποιείται για τα μεγάλα αιτήματα κατανομής (πχ. 163 ή 265KB). Η pool για το μικρότερο μέγεθος έχει έναν ενιαίο συνδεδεμένο κατάλογο, επειδή δεν πρέπει να συγχωνευτεί ή να χωρίσει, δεδομένου ότι μόνο ένα μέγεθος block μπορεί να ζητηθεί από αυτήν. Το υπόλοιπο των pool περιλαμβάνει τους διπλά συνδεόμενους καταλόγους ελεύθερων block, με τις επιγραφές που περιέχουν το μέγεθος κάθε αντίστοιχου block και της πληροφορίας για την επικρατούσα κατάστασή τους (δηλαδή, σε λειτουργία ή ελεύθερα). Αυτοί οι μηχανισμοί υποστηρίζουν αποτελεσματικά την άμεση συγχώνευση και το διαχωρισμό μέσα σε αυτές τις pool, και σχεδιάζεται με τη μεθοδολογία μας ο εσωτερικός και εξωτερικός τεμαχισμός στο εξατομικευμένο διαχειριστή DM.

Σε αυτήν την περιπτωσιακή μελέτη, έχουμε συγκρίνει τη λύση μας με τους ανά περιοχή σημασιολογικούς διαχειριστές [82], [56]. Επίσης, έχουμε εξετάσει το διαχειριστή μας με τις ίδιες βελτιστοποιημένες εκδόσεις Kingsley και Lea που χρησιμοποιούνται στο προηγούμενο παράδειγμα (δηλαδή, DRR) δεδομένου ότι αυτοί είναι οι τύποι διαχειριστών DM που βρίσκονται στα ενσωματωμένα συστήματα. Τα επιτευχθέντα αποτελέσματα αποτυπώματος μνήμης απεικονίζονται στην Εικόνα 8.10. Αυτά τα αποτελέσματα δείχνουν ότι οι τιμές που λήφθηκαν με την διαχείριση DM σχεδίασαν τη μεθοδολογία λαμβάνοντας υπόψη τις σημαντικές βελτιώσεις στο αποτύπωμα μνήμης έναντι της χειρωνακτικά σχεδιασμένης εφαρμογής ενός διαχειριστή περιοχών (28.47%), του Lea (29.46%) και της βελτιστοποιημένης έκδοσης Kingsley (33.01%). Αυτά είναι τα αποτελέσματα, επειδή ο εξατομικευμένος διαχειριστής DM, είναι σε θέση τεμαχισμού

του συστήματος με δύο τρόπους. Κατ' αρχάς, επειδή ο σχεδιασμός και η συμπεριφορά του ποικίλλουν σύμφωνα με τα ζητούμενα διαφορετικά μεγέθη block. Δεύτερον, στις pool όπου μια σειρά των αιτημάτων μεγεθών block επιτρέπεται, χρησιμοποιεί τις άμεσες συγχωνευμένες και διαιρεμένες υπηρεσίες για να μειώσει και τον εσωτερικό και εξωτερικό τεμαχισμό. Στους διαχειριστές περιοχών, τα μεγέθη block κάθε μιας διαφορετικής περιοχής καθορίζονται σε ένα μέγεθος block και όταν χρησιμοποιούνται τα block διάφορων μεγεθών, αυτό δημιουργεί τον εσωτερικό τεμαχισμό. Στο Lea, η σειρά των μεγεθών block που επιτρέπονται, δεν εγκαθιστά ακριβώς αυτά που χρησιμοποιούνται στις εφαρμογές και τα μικτά μεγέθη αποεκχώρησης λίγων μεγεθών block και έτσι παράγουν απόβλητα των καταλόγων μεγεθών που δεν χρησιμοποιούνται στο σύστημα. Επιπλέον, η συχνή χρήση των συνεχών/διαιρεμένων μηχανισμών στο Lea δημιουργεί πρόσθετα γενικά κόστη στο χρόνο εκτέλεσης έναντι του διαχειριστή περιοχών, ο οποίος έχει ρυθμίσει καλύτερα τα μεγέθη κατανομής σε εκείνους που χρησιμοποιούν την εφαρμογή. Τέλος, στο Kingsley, οι συγχωνευμένοι/διαιρεμένοι μηχανισμοί εφαρμόζονται, αλλά μια αρχική μνήμη ορίου είναι διατηρημένη και διανεμημένη μεταξύ των διαφορετικών καταλόγων για τα μεγέθη. Σε αυτήν την περίπτωση, κάποια από τα "απόβλητα" (ή pool των DM block στο Kingsley) είναι υποαπασχολούμενα. Επιπλέον, η τελική ενσωματωμένη εφαρμογή συστημάτων που χρησιμοποιεί τον εξατομικευμένο διαχειριστή DM, επιτυγχάνει τα καλύτερα ενεργειακά αποτελέσματα από τις εφαρμογές, χρησιμοποιώντας τους διαχειριστές DM γενικού σκοπού. Σε αυτήν την περίπτωση μελέτης, ο διαχειριστής DM υιοθετεί τις λιγότερες προσβάσεις διαχείρισης στους DM και το αποτύπωμα μνήμης από οποιοδήποτε άλλον διαχειριστή. Κατά συνέπεια, ο διαχειριστής DM μας επιτρέπει τη γενική αποταμίευση κατανάλωσης ενέργειας σχετικά με τις περιοχές, 11% πέρα από Kingsley και 14% πέρα από το Lea (Εικόνα 8.11).

8.5.3 Μεθοδολογία που εφαρμόζεται σε ένα τρισδιάστατο δεδομένο σύστημα

Η τρίτη περιπτωσιολογική μελέτη είναι η τρισδιάστατη δεδομένη ενότητα [94] μιας ολόκληρης τρισδιάστατης τηλεοπτικής εφαρμογής συστημάτων. Αυτή η ενότητα ανήκει στην κατηγορία τρισδιάστατων αλγορίθμων με τα εξελικτικά πλέγματα [92] που προσαρμόζουν την ποιότητα κάθε αντικειμένου που επιδεικνύεται στην οθόνη που αποσπά έγκαιρα την προσοχή των χρηστών (πχ. ποιότητα των συστημάτων [94] υπηρεσιών). Επομένως, τα αντικείμενα αντιπροσωπεύονται από κορυφές (ή τρίγωνα)και αντιμετωπίζουν την ανάγκη να αποθηκευτούν δυναμικά λόγω της αβεβαιότητας να συντάσσουν το χρόνο των χαρακτηριστικών γνωρισμάτων των αντικειμένων που δίνουν (δηλαδή, αριθμός



Σχήμα 8.10: Μέγιστα αποτελέσματα αποτυπώματος μνήμης στην τρισδιάστατη εφαρμογή ανοικοδόμησης.

και ανάλυση). Κατ' αρχήν, εκείνες οι κορυφές που βρίσκονται στις πρώτες τρεις φάσεις ολόκληρης της διαδικασίας απεικόνισης, δηλαδή, του μετασχηματισμού model-view, φωτεινής διαδικασίας και κανονικού μετασχηματισμού [94] άποψης. Τέλος, το σύστημα επεξεργάζεται τα χαρακτηριστικά των αντικειμένων στις επόμενες τρεις φάσεις (δηλαδή, ψαλίδισμα, άποψη-και επαναπροσαρμογή [94]) της διαδικασίας απεικόνισης, για να παρουσιάσει το τελικό αντικείμενο. Σύμφωνα με τα πειράματά μας, αυτή η εφαρμογή μοιάζει πολύ με τη συμπεριφορά DM της οπτικής εφαρμογής αποκωδικοποιητών σύστασης MPEG4 (VTC) στα πρότυπα [40].

Σε αυτήν την περίπτωση, έχουμε συγκρίνει τον εξατομικευμένο διαχειριστή μας με το Lea v2.7.2, το βελτιστοποιημένο της έκδοσης Kingsley, και λόγω των ιδιαίτερων φάσεων εμπιστοσύνης της, όπου οι ενδιάμεσες τιμές χτίζονται με έναν σειριακό τρόπο και καταστρέφονται τελικά στο τέλος κάθε φάσης (για τις φάσεις που χειρίζονται τις κορυφές), έχουμε χρησιμοποιήσει επίσης Obstacks [56]. Το Obstacks είναι ένας γνωστός εξατομικευμένος διαχειριστής DM που βελτιστοποιείται με παρόμοιας με σωρό συμπεριφορά για τις εφαρμογές. Το Obstacks χρησιμοποιείται εσωτερικά από το GCC του GNU. Όπως η Εικόνα 8.12 παρουσιάζει, το Lea και ο διαχειριστής περιοχών επιτυγχάνουν τα καλύτερα αποτελέσματα στο αποτύπωμα μνήμης από το διαχειριστή DM Kingsley. Επί-

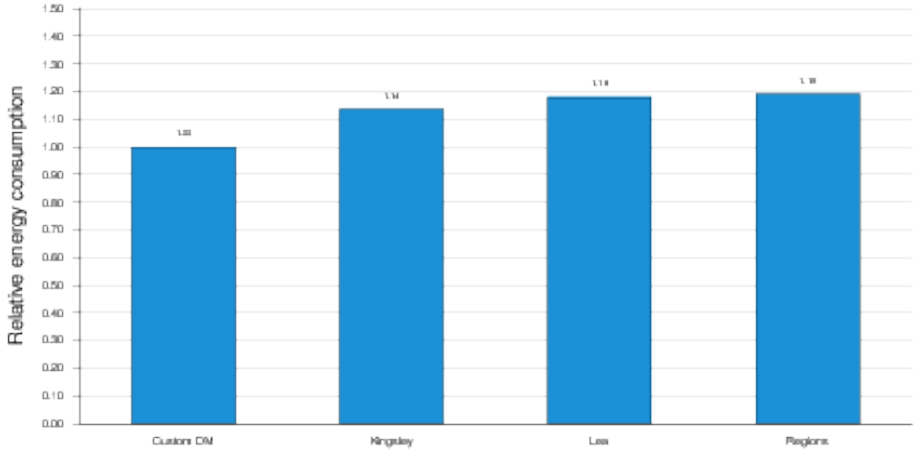
σης, λόγω της παρόμοιας με σωρό συμπεριφοράς της εφαρμογής στις φάσεις που χειρίζονται τα τρίγωνα, το Obstacks επιτυγχάνει ακόμα τα καλύτερα αποτελέσματα από τους διαχειριστές Lea, στο αποτύπωμα μνήμης. Εντούτοις, ο διαχειριστής που σχεδιάζεται με τη μεθοδολογία μας, βελτιώνει περαιτέρω τις τιμές αποτυπώματος μνήμης που λαμβάνονται από το Obstacks. Το γεγονός είναι ότι η βελτιστοποιημένη συμπεριφορά Obstacks δεν μπορεί να χρησιμοποιηθεί στις τελικές φάσεις της δεδομένης διαδικασίας, επειδή τα πρόσωπα όλων χρησιμοποιούνται ανεξάρτητα σε ένα διαταραγμένο σχεδιασμό και ελευθερώνονται χωριστά. Κατά συνέπεια, η Obstacks πάσχει από μια υψηλή ποινική ρήτρα στο αποτύπωμα μνήμης λόγω του κατακερματισμού, επειδή δεν έχει πάρει μια κατάλληλη δομή συντήρησης των block DM για τέτοια συμπεριφορά αποεκχώρησης.

Από μια ενεργειακή άποψη, ο εξατομικευμένος DM διαχειριστής βελτιώνει επίσης τα αποτελέσματα που επιτυγχάνονται με τους μελετημένους γενικής χρήσης διαχειριστές στο βασισμένο προσομοιωτή ARM. Οι ενεργειακοί αριθμοί του τελικού ενσωματωμένου συστήματος μεταβάλλονται από 18% σε 20%, αντίστοιχα. Στην περίπτωση Obstacks και Kingsley, παράγουν τις λιγότερες προσβάσεις μνήμης από τον εξατομικευμένο DM διαχειριστή και σε αυτό οφείλεται η βελτιστοποιημένη διαχείρισή των block DM για την απόδοση, αλλά η μεγαλύτερη κατανάλωση του αποτυπώματος μνήμης, επιτρέπει τελικά τη γενική αποταμίευση 13% και 14% στην κατανάλωση ενέργειας για το διαχειριστή DM μας, αντίστοιχα (Εικόνα 8.13).

Τέλος, για να αξιολογήσουμε τη διαδικασία σχεδιασμού με την μεθοδολογία μας, μας πήρε δύο εβδομάδες για τον προτεινόμενο σχεδιασμό και τη ροή εφαρμογής των τελικών εξατομικευμένων διαχειριστών DM για κάθε περιπτωσιολογική μελέτη. Επίσης, όπως ο πίνακας 8.2 παρουσιάζει, αυτοί οι διαχειριστές DM επιτυγχάνουν τις λιγότερες τιμές αποτυπώματος μνήμης με μόνο ένα 10% υπερυψωμένο (κατά μέσον όρο) κατά τη διάρκεια του χρόνου εκτέλεσης του γρηγορότερου γενικής χρήσης διαχειριστή DM που παρατηρείται σε αυτές τις περιπτωσιολογικές μελέτες, δηλαδή, Kingsley. Επιπλέον, η μείωση στην απόδοση δεν είναι σχετική, δεδομένου ότι οι εξατομικευμένοι διαχειριστές DM την συντηρούν σε πραγματικό χρόνο συμπεριφοράς που απαιτείται από τις εφαρμογές, και κατά συνέπεια, ο χρήστης δεν θα παρατηρήσει οποιαδήποτε διαφορά. Επιπλέον, οι προτεινόμενοι εξατομικευμένοι διαχειριστές DM , περιλαμβάνουν τις βελτιστοποιημένες εσωτερικές οργανώσεις διαχείρισης DM των block μνήμης για κάθε μελετημένη εφαρμογή, οι οποίοι παράγουν συνήθως μια μείωση στις προσβάσεις μνήμης έναντι των γενικής χρήσης διαχειριστών κατάστασης προόδου, όπως το Lea ή οι περιοχές, οι οποίοι σχεδιάζονται για ένα ευρύ φάσμα των σχεδίων συμπεριφοράς των αιτημάτων μνήμης και DM. Μόνο οι Kingsley και Obstacks παράγουν τις λιγότερες προσβάσεις μνήμης από το διαχειριστή DM συνήθειας λόγω της προσανατολισμένης απόδοσης των σχεδιασμών τους

ξεπερνώντας το, αλλά σπαταλώντας ένα μεγάλο μέρος του αποτυπώματος μνήμης λόγω του τεμαχισμού. Κατά συνέπεια, αυτοί οι διαχειριστές απαιτούν τις μεγαλύτερες μνήμες εντός ολοκληρωμένου κυκλώματος για να αποθηκεύσουν τα δυναμικά στοιχεία αλλά απαιτείται περισσότερη ενέργεια ανά πρόσβαση [97], η οποία αντιδρά στις πιθανές βελτιώσεις στην κατανάλωση ενέργειας λόγω των λιγότερων προσβάσεων μνήμης. Εν περιλήψει, οι εξατομικευμένοι διαχειριστές DM μας μειώνουν επίσης κατά 15% κατά μέσον όρο τη συνολική κατανάλωση ενέργειας του τελικού ενσωματωμένου συστήματος έναντι των γενικής χρήσης διαχειριστών DM.

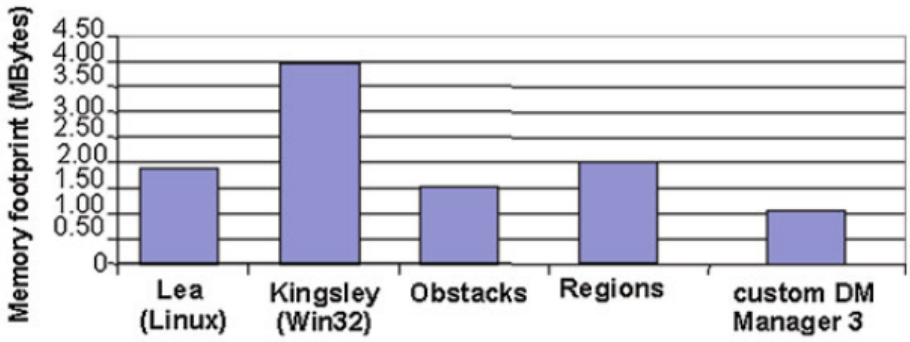
Αν και η μεθοδολογία μας για το σχεδιασμό των εξατομικευμένων διαχειριστών DM έχει οδηγηθεί από την ελαχιστοποίηση του αποτυπώματος μνήμης, μπορεί να αποκτήσει νέο στόχο προς την τέλεια επίτευξη των διαφορετικών ανταλλαγών μεταξύ οποιονδήποτε σχετικών παραγόντων σχεδιασμού, όπως η βελτίωση της απόδοσης ή η κατανάλωση λίγο περισσότερου αποτυπώματος μνήμης, για να επιτύχει περισσότερη ενέργεια ως αποταμίευση.



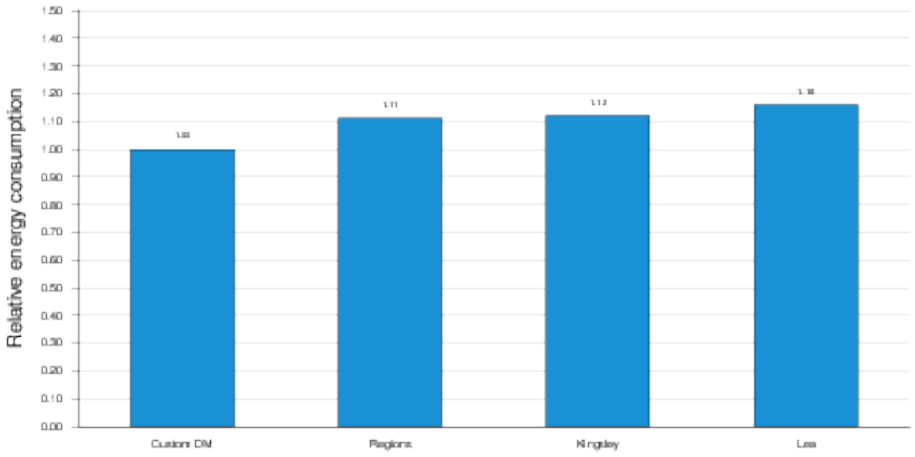
Σχήμα 8.11: Σχετική ενεργειακή βελτίωση του εξατομικευμένου διαχειριστή DM στην τρισδιάστατη εφαρμογή ανοικοδόμησης.

8.6 Συμπεράσματα

Οι ικανότητες των φορητών ενσωματωμένων συσκευών έχουν βελτιωθεί τα τελευταία χρόνια και κάνουν εφικτή την διαχείριση των πολύ σύνθετων και δυναμικών εφαρμογών πολυμέσων. Τέτοιες εφαρμογές έχουν αυξήσει πρόσφατα την πολυπλοκότητα τους. Έτσι, στα καταναλωτικά ενσωματωμένα συστήματα, νέες μεθοδολογίες σχεδιασμού πρέπει να είναι διαθέσιμες για να χρησιμοποι-



Σχήμα 8.12: Μέγιστα αποτελέσματα αποτυπώματος μνήμης στην τρισδιάστατη απόδοση της εφαρμογής.



Σχήμα 8.13: Σχετική βελτίωση του εξατομικευμένου διαχειριστή DM στην απόδοση της εφαρμογής.

ήσουν αποτελεσματικά τη παρούσα μνήμη, σε αυτά τα πολύ περιορισμένα ενσωματωμένα συστήματα. Σε αυτό το κεφάλαιο έχουμε παρουσιάσει μια συστηματική μεθοδολογία, που καθορίζει και ερευνά το δυναμικό χώρο σχεδιασμού διαχείρισης μνήμης των σχετικών αποφάσεων, προκειμένου να σχεδιαστεί η δυναμική εξατομικευμένη μνήμη διαχειριστών με μια μείωση του αποτυπώματος μνήμης, για τις ασύρματες εφαρμογές δικτύων και δυναμικών πολυμέσων. Τα αποτελέσματά μας στις πραγματικές εφαρμογές παρουσιάζουν σημαντικές βελτιώσεις στο αποτύπωμα μνήμης, πέρα από την κατάσταση προόδου γενικής χρήσης και στους χειρωνακτικά βελτιστοποιημένους εξατομικευμένους διαχειριστές DM, που υφίστανται μόνο μικρά γενικά έξοδα στο χρόνο εκτέλεσης.