ΑΠΟΔΟΤΙΚΗ ΔΙΑΧΕΙΡΙΣΗ ΠΛΕΟΝΑΣΜΟΥ ΓΙΑ ΑΞΙΟΠΙΣΤΗ ΑΠΟΘΗΚΕΥΣΗ ΡΟΩΝ

Η
ΜΕΤΑΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ ΕΞΕΙΔΙΚΕΥΣΗΣ

Υποβάλλεται στην

ορισθείσα από την Γενική Συνέλευση Ειδικής Σύνθεσης
του Τμήματος Πληροφορικής
Εξεταστική Επιτροπή

από τον

Μενέλαο Μαρκουλάκη

ως μέρος των Υποχρεώσεων

για τη λήψη

του

ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΔΙΠΛΩΜΑΤΟΣ ΣΤΗΝ ΠΛΗΡΟΦΟΡΙΚΗ

ΜΕ ΕΞΕΙΔΙΚΕΥΣΗ ΣΤΑ ΥΠΟΛΟΓΙΣΤΙΚΑ ΣΥΣΤΗΜΑΤΑ

Ιούνιος 2009

ΑΦΙΕΡΩΣΗ

Η σελίδα αυτή είναι προαιρετική και περιέχει αφιέρωση σε κάποιο σημαντικό πρόσωπο.

Μέγιστο: 1 σελίδα μαζί με την επικεφαλίδα.

Προτεινόμενο μέγεθος: 1-2 γραμμές.

ΕΥΧΑΡΙΣΤΙΕΣ

Ευχαριστώ τον κ. Αναστασιάδη για την πολύτιμη βοήθεια του στην ολοκλήρωση της μεταπτυχιακής μου εργασίας.

Η παρούσα εργασία χρηματοδοτήθηκε από το πρόγραμμα INTERSAFE (No 303090/YD7631) της Κοινοτικής πρωτοβουλίας INTERREG IIIA (Ελλάδα-Αλβανία).

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΥΡΕΤΗΡΙΟ ΠΙΝΑΚΩΝ

ΕΥΡΕΤΗΡΙΟ ΣΧΗΜΑΤΩΝ

ΠΕΡΙΛΗΨΗ

Μενέλαος Μαρκουλάκης του Αλεξάνδρου και της Χριστίνας. MSc, Τμήμα Πληροφορικής, Πανεπιστήμιο Ιωαννίνων, Ιούνιος, 2009. Αποδοτική Διαχείριση Πλεονασμού για Αξιόπιστη Αποθήκευση Ροών.

Επιβλέποντας: Στέργιος Β. Αναστασιάσης.

Το πρόβλημα της αποθήκευσης πολλαπλών ροών δεδομένων σε πραγματικό χρόνο εμφανίζεται σε σημαντικές εφαρμογές, όπως είναι η διαχείριση πόρων σε μεγάλα υπολογιστικά συστήματα, η προστασία του περιβάλλοντος και ο έλεγχος της κυκλοφορίας οχημάτων. Στην παρούσα εργασία σχεδιάσαμε και υλοποιήσαμε ένα πρωτότυπο σύστημα προκειμένου να μελετήσουμε βασικά σχεδιαστικά ζητήματα κατανεμημένης αποθήκευσης ροών. Το προτεινόμενο σύστημα παράγει πλεονάζοντα δεδομένα με βάση τον αλγόριθμο Reed-Solomon, τα οποία στην συνέχεια διανέμει μαζί με τα αρχικά δεδομένα σε πολλαπλούς αποθηκευτικούς κόμβους. Το σύστημα διασφαλίζει την αξιόπιστη διαχείριση των μεταδεδομένων με συνεχή καταγραφή των τροποποιήσεων τους και την περιοδική αποθήκευση τους σε πολλαπλούς ανεξάρτητους κόμβους. Τέλος, μειώνει το απαιτούμενο εύρος ζώνης του δικτύου με την συγχώνευση στην ίδια απομακρυσμένη κλήση αποθήκευσης των δεδομένων από πολλαπλές ροές. Εξετάζουμε πειραματικά τις απαιτήσεις σε εύρος ζώνης του συστήματος κατά τα διάφορα στάδια αποθήκευσης καθώς και κατά την ανάκτηση των αρχικών δεδομένων σε περίπτωση σφάλματος.

EXTENDED ABSTRACT IN ENGLISH

Markoulakis, Menelaos. MSc, Computer Science Department, University of Ioannina, Greece. June 2009.
Surname, Name, Initials. Title Abbreviation (MSc ή PhD), Computer Science Department, University of Ioannina, Greece. Graduation Month, Graduation Year. Tilte of Dissertation in English. Efficient Management of Redundancy for Reliable Stream Storage

Thesis Supervisor: Stergios V. Anastasiadis.

In the present thesis, we focus on the unique requirements of stream storage that includes write-mostly access requests, appending of new data at the end of existing files and accumulation of large amounts of incoming data that arrive in real-time at rates that potentially change over time. Existing stream playback servers are not appropriate for this purpose because they are mainly optimized for the real-time concurrent retrieval of large numbers of streams that have been previously stored over mirrored data files. On the other hand general-purpose file servers can be costly for our purposes because they have been designed to support mostly moderate-sized random-access requests to mirrored or erasure-correcting-code redundant data.

In order to address the space and bandwidth challenges that arise in reliable stream storage, we choose to take advantage of a two-tier cluster-based system configuration. The front-end nodes are responsible for receiving the incoming data, while the back-end nodes do the actual storage to multiple disks. We strive to tolerate failures inexpensively

by generating redundancy based on erasure-correcting coding instead of mirroring, and avoid excessive network overhead by batching the data from multiple streams into a single write request per back-end storage node. We keep the metadata partitioned across different front-end nodes for improved scalability. We tolerate loss of metadata by logging their recent modifications into multiple databases and taking multiple periodical checkpoints. We have built a prototype cluster-based system based on above design choices and evaluate the resource requirements across different parts of it using large numbers of incoming streams.

ΚΕΦΑΛΑΙΟ 1. ΕΙΣΑΓΩΓΗ

1.1. Αντικείμενο μεταπτυχιακής εργασίας

Στόχος της παρούσας μεταπτυχιακής εργασίας είναι ο σχεδιασμός και η υλοποίηση ενός συστήματος το οποίο θα αποθηκεύει αποδοτικά και αξιόπιστα ροές δεδομένων με διάφορους ρυθμούς μετάδοσης σε πραγματικό χρόνο. Ο όρος αποδοτική αποθήκευση αναφέρεται στην ταχύτητα εγγραφής και το πλήθος ροών που υποστηρίζει ταυτόχρονα το σύστημα. Από την άλλη πλευρά, με τον όρο αξιόπιστη αποθήκευση αναφερόμαστε στη εξασφάλιση ότι ο χρήστης μπορεί να ανακτήσει τα δεδομένα μετά την επιβεβαίωση της εγγραφής ακόμα και εάν συμβούν αποτυχίες δίσκου ή κόμβου. .

Μέχρι σήμερα έχουν υλοποιηθεί διάφορα συστήματα αποθήκευσης δεδομένων τα οποία λύνουν τα παραπάνω προβλήματα με βάση τις ανάγκες και τους στόχους που έχει το καθένα από αυτά. Αυτά τα συστήματα μπορούν να χωριστούν σε δυο κατηγορίες, οι διακομιστές ανάκτησης πολυμέσων και στα συστήματα γενικού σκοπού.

Οι διακομιστές ανάκτησης αποθηκεύουν ροές δεδομένων με στόχο να έχουν καλή απόδοση στα αιτήματα ανάκτησης. Από την άλλη πλευρά, το προτεινόμενο σύστημα

αποθηκεύει ροές δεδομένων με κύριο στόχο του να έχει καλή απόδοση σε αιτήματα αποθήκευσης πολλαπλών διαφορετικών ροών δεδομένων(write request).

Τα γενικού σκοπού συστήματα είναι συστήματα τα οποία αποτελούνται από μια συλλογή κόμβων αποθήκευσης ώστε να παρέχουν απόδοση και αξιοπιστία με μικρό κόστος και μεγαλύτερη κλιμακωσιμότητα.
Και αυτά τα συστήματα έχουν βασικές διαφορές σε σχέση με το προτεινόμενο σύστημα. Το σύστημα μας αποθηκεύει μεταβλητού ρυθμού ροές δεδομένων, αλλά το εύρος ροών είναι μεγαλύτερο στο προτεινόμενο σύστημα σε σχέση με αυτό που καλύπτουν τα γενικού σκοπού συστήματα. Επίσης, το σύστημα μας αποθηκεύει μόνο ροές δεδομένων, άρα εξυπηρετεί κυρίως ακολουθιακές εγγραφές, σε αντίθεση με τα γενικού σκοπού συστήματα στα οποία, συχνά, η αποθήκευση νέων δεδομένων έχει ως αποτέλεσμα την επανεγγραφή δεδομένων.

1.2. Δομή της Διατριβής

Στο Κεφάλαιο 1 γίνεται μια σύντομή περιγραφή στο πρόβλημα και παρουσιάζεται ο βασικός στόχος της παρούσας μεταπτυχιακής εργασίας. Στο Κεφάλαιο 2 γίνεται μια αναδρομή σε υπάρχουσα συστήματα αποθήκευσης και ο τρόπος που επιλύουν τα διάφορα ζητήματα που προκύπτουν. Στο Κεφάλαιο 3 παρουσιάζεται το θεωρητικό υπόβαθρο στο οποίο στηριχθήκαμε για την υλοποίηση της εργασίας. Στο Κεφάλαιο 4 παρουσιάζεται η αρχιτεκτονική του συστήματος. Στο Κεφάλαιο 5 παρουσιάζεται η υλοποίηση του συστήματος. Στο Κεφάλαιο 6 παρουσιάζονται τα πειραματικά αποτελέσματα. Στο Κεφάλαιο 7 παρουσιάζονται τα συμπεράσματα.

ΚΕΦΑΛΑΙΟ 2. ΣΧΕΤΙΚΕΣ ΕΡΓΑΣΙΕΣ

2.1. Εισαγωγή

Σε αυτό το κεφάλαιο συνοψίζουμε την προηγούμενη σχετική έρευνα σε συστήματα αποθήκευσης δεδομένων γενικού σκοπού και πολυμέσων. Η παρούσα διατριβή αναφέρεται στο σχεδιασμό ενός συστήματος, το οποίο αποθηκεύει ταυτόχρονα πολλαπλές ετερογενείς ροές δεδομένων.

2.2. Κατανεμημένα Συστήματα Αποθήκευσης

2.2.1. Βασισμένη σε αντικείμενα

Ένα *αντικείμενο αποθήκευσης* είναι μια λογική συλλογή δεδομένων σε μια συσκευή αποθήκευσης, που υποστηρίζει: 1) μεθόδους για την πρόσβαση των δεδομένων, 2) χαρακτηριστικά που περιγράφουν τα δεδομένα και 3) πολιτικές ασφάλειας που αποτρέπουν μη-πιστοποιημένη πρόσβαση [7].

Τα αντικείμενα μπορούν να θεωρηθούν ως συνδυασμός των *αρχείων* και των *μπλοκ* και συνδυάζουν τα πλεονεκτήματα και των δυο. Όπως και τα μπλοκ, έτσι και τα αντικείμενα είναι μια μονάδα δεδομένων που μπορεί να αποθηκευτεί άμεσα σε μια συσκευή, χωρίς να περάσει από έναν κεντρικό υπολογιστή. Αυτή η άμεση πρόσβαση προσφέρει πλεονεκτήματα απόδοσης παρόμοια με αυτά των μπλοκ. Όπως τα αρχεία, τα αντικείμενα προσπελάζονται χρησιμοποιώντας μια διεπαφή που ανεξαρτητοποιεί τις εφαρμογές αποθήκευσης από τα μεταδεδομένα τα οποία είναι απαραίτητα για την αποθήκευση ενός αντικειμένου, καθιστώντας το αντικείμενο προσβάσιμο από διαφορετικές πλατφόρμες. Η παροχή της άμεσης πρόσβαση στις συσκευές αποθήκευσης είναι επομένως η βασική συμβολή της μεθόδου που βασίζεται στα αντικείμενα.

Τα αντικείμενα δεν έχουν σταθερό μέγεθος και έτσι μπορούν να χρησιμοποιηθούν για την αποθήκευση οποιουδήποτε τύπου δεδομένων, όπως αρχεία, εγγραφές βάσεων δεδομένων, ιατρικές εικόνες, ή πολυμέσα. Ένα αντικείμενο θα μπορούσε να χρησιμοποιηθεί ακόμα και για την αποθήκευση ενός συστήματος αρχείων ή μια βάση δεδομένων. Τα αντικείμενα αποτελούνται από τα δεδομένα, τα χαρακτηριστικά που περιγράφουν, ποιοι μπορούν να προσπελάσουν τα δεδομένα και τα αντίστοιχα μεταδεδομένα. Τα δεδομένα που είναι αποθηκευμένα σε ένα αντικείμενο δεν είναι ορατά στην συσκευή αποθήκευσης των αντικειμένων και αποθηκεύονται στο τμήμα του αντικειμένου που αναφέρεται στα δεδομένα.

Τα συστήματα τα οποία χρησιμοποιούν αντικείμενα για την αποθήκευση των δεδομένων τους έχουν κάποια πλεονεκτήματα σε σχέση με αυτά τα οποία χρησιμοποιούν μπλοκ.

- Η συσκευή αποθήκευσης ξέρει πια δεδομένα σχετίζονται και έτσι μπορεί να τα διατάξει στο δίσκο με τον καλύτερο δυνατό τρόπο

- Η συσκευή αποθήκευσης μπορεί να διαχειριστεί τα αντικείμενα ανεξάρτητα, δηλαδή μπορεί να χρησιμοποιεί διαφορετικούς τρόπους ασφάλειας ανά αντικείμενο, παρόμοια με τον τρόπο με τον οποίο τα αρχεία προστατεύονται από ένα διακομιστή.

- Τα αντικείμενα επιτρέπουν στις εφαρμογές αποθήκευσης να θέτουν εύκαμπτες πολιτικές ασφαλείας που θα οδηγήσουν στην έγκριση για πρόσβαση σε μια ολόκληρη συσκευή, σε μια συλλογή των αντικειμένων, ή ακόμα και σε ένα μέρος των δεδομένων που περιέχει ένα αντικείμενο

- Τα αντικείμενα διατηρούν τα δικά τους μεταδεδομένα για τα δεδομένα τα οποία περιέχουν. Για αυτό το λόγο όταν ένας πελάτης θέλει να χρησιμοποιήσει ένα αντικείμενο δε χρειάζεται να επικοινωνεί κάθε φορά με τον διακομιστή μεταδεδομένων για να αποκτά πρόσβαση σε αυτό. Ο πελάτης επικοινωνεί μόνο την πρώτη φορά με τον διακομιστή μεταδεδομένων για να αποκτήσει πρόσβαση σε ένα αντικείμενο και όταν το αντικείμενο χρησιμοποιηθεί για πρώτη φορά αποθηκεύει το αναγνωριστικό του πελάτη. Έτσι, το αντικείμενο θα γνωρίζει ότι ο συγκεκριμένος πελάτης έχει πρόσβαση σε αυτό και θα του δίνει τη δυνατότητα να το χρησιμοποιεί άμεσα στο μέλλον

2.2.2. Συστοιχίες αποθήκευσης

Τα εμπορικά συστήματα αποθήκευσης είναι συνήθως μονολιθικά και πολύ ακριβά. Τα συστήματα αυτά αποτελούνται από κάποια συστατικά τα οποία διατηρούν εσωτερικά κοινή πληροφορία για να προσφέρουν μεγάλη διαθεσιμότητα στους πελάτες τους [6]. Από την άλλη πλευρά, υπάρχουν τα συστήματα αποθήκευσης που βασίζονται σε συστοιχίες, τα οποία αποτελούνται από πολλούς μικρούς κόμβους αποθήκευσης ώστε να παρέχουν απόδοση και αξιοπιστία ανάλογη των εμπορικών συστημάτων. Τα συστήματα αυτά έχουν χαμηλότερο κόστος και προσφέρουν μεγαλύτερη κλιμακωσιμότητα σε σχέση με τα εμπορικά. Τα συστήματα αυτά χρησιμοποιούν δυο εναλλακτικούς τρόπους κωδικοποίησης της πληροφορία τους για να πετύχουν μεγάλη διαθεσιμότητα. Ένας τρόπος είναι να χρησιμοποιούν τη μέθοδο του αντικατοπτρισμού (mirroring), στην οποία διατηρούν πολλαπλά αντίγραφα ενός μπλοκ δεδομένων σε περισσότερους από ένα κόμβους. Εναλλακτικά μπορούν να χρησιμοποιήσουν την τεχνική του πλεονασμού (parity). Για παράδειγμα, ένας αλγόριθμος παραγωγής πλεονάζουσας πληροφορίας, δέχεται ως είσοδο *m* μπλοκ δεδομένων και παράγει *(n-m)* μπλοκ πλεονάζουσας

πληροφορίας. Έτσι, με οποιαδήποτε *m* από τα παραπάνω *n* μπλοκ μπορεί να ανακτήσει την αρχική πληροφορία.

Τα συστήματα που χρησιμοποιούν τη μέθοδο του αντικατοπτρισμού πετυχαίνουν μεγαλύτερη ρυθμαπόδοση δίσκου κατά την ανάγνωση δεδομένων και την αποκατάσταση σφαλμάτων. Από την άλλη πλευρά, τα συστήματα που χρησιμοποιούν τη τεχνική του πλεονασμού καταφέρνουν να αντιμετωπίζουν αποτυχίες που συμβαίνουν στο σύστημα καταναλώνοντας το λιγότερο αποθηκευτικό χώρο. Όταν, στα συστήματα υπάρχουν σειριακές προσπελάσεις, τα πλεονεκτήματα των δυο μεθόδων είναι τα παρόμοια.

Το Ursa Minor είναι ένα σύστημα αποθήκευσης το οποίο χρησιμοποιεί αντικείμενα για την αποθήκευση των δεδομένων του [6]. Ένα αντικείμενο προσδιορίζεται από βασικά χαρακτηριστικά, όπως είναι το μέγεθος του και τα δεδομένα που περιέχει. Τα δεδομένα που περιέχει ένα αντικείμενο δεν αντιμετωπίζονται σαν μια ενιαία ομάδα, δηλαδή μπορεί κάποιος να προσπελάσει οποιαδήποτε ψηφιολέξη των δεδομένων που περιέχει το αντικείμενο. Κάθε αντικείμενο έχει ένα μοναδικό αναγνωριστικό σε έναν ονοματοχώρο. Το βασικό πλεονέκτημα των συστημάτων που χρησιμοποιούν τα αντικείμενα σε σχέση με αυτά που χρησιμοποιούν μπλοκ είναι ότι το αντικείμενο διαθέτει περισσότερη πληροφορία για τα χαρακτηριστικά των δεδομένων που περιέχει και επίσης διατηρεί όλα τα μεταδεδομένα στη συσκευή αποθήκευσης. Αντίθετα, τα συστήματα που χρησιμοποιούν μπλοκ διατηρούν τα μεταδεδομένα στην εφαρμογή αποθήκευσης. Τελικά, οι πελάτες μπορούν να έχουν άμεση πρόσβαση στους κόμβους που περιέχουν τα αντικείμενα. Λειτουργίες που αφορούν τα μεταδεδομένα, όπως η δημιουργία και η διαγραφή ενός αντικειμένου, γίνονται μέσω του διαχειριστή των αντικειμένων. Ένα πελάτης που θέλει να ανακτήσει τα δεδομένα που βρίσκονται σε κάποιο αντικείμενο θα πρέπει πρώτα να επικοινωνήσει με τον διαχειριστή των αντικειμένων για να του ζητήσει τα μεταδεδομένα για τα δεδομένα που θέλει να προσπελάσει και να γίνει και η απαραίτητη πιστοποίηση. Στη συνέχεια μπορεί να επικοινωνήσει με τον κόμβο που διατηρεί αυτά τα δεδομένα και να τα προσπελάσει άμεσα. Το Ursa Minor ανήκει στα συστήματα που χρησιμοποιούν τόσο την τεχνική του αντικατοπτρισμού, όσο και την τεχνική του πλεονασμού για να κωδικοποιήσουν την πληροφορία που περιέχουν.

Το σύστημα αρχείων Panasas, επίσης, χρησιμοποιεί αντικείμενα για την αποθήκευση των δεδομένων του [17]. Κάθε αρχείο χωρίζεται σε τμήματα καθένα από τα οποία αποθηκεύεται σε ένα αντικείμενο, παρέχοντας έτσι πλεονασμό και υψηλή ρυθμαπόδοση κατά την προσπέλαση του εκάστοτε δίσκου. Τα μεταδεδομένα διατηρούνται σε κόμβους του συστήματος, οι οποίοι μεσολαβούν για να προσπελάσουν οι πελάτες τα αντικείμενα. Οι πελάτες για να προσπελάσουν τα αντικείμενα χρησιμοποιούν τα πρωτόκολλα iSCSI [15] και Object Storage Device (OSD). Το πρωτόκολλο OSD χρησιμοποιείται για τη δημιουργία και τη διαγραφή αντικειμένων, καθώς επίσης για το χειρισμό των χαρακτηριστικών των αντικειμένων και την προσπέλαση των δεδομένων τους. Όλες οι λειτουργίες Εισόδου/Εξόδου που αφορούν τα αντικείμενα πραγματοποιούνται άμεσα από το κόμβο που διατηρεί τα αντικείμενα χωρίς να χρειάζεται να επικοινωνεί με τους κόμβους που διατηρούν τα μεταδεδομένα του συστήματος. Οι πελάτες επικοινωνούν με τους κόμβους που διατηρούν τα μεταδεδομένα του συστήματος χρησιμοποιώντας τη μέθοδο της απομακρυσμένης κλήσης διαδικασίας (RPC). Έτσι, παίρνουν την απαραίτητη πιστοποίηση για να προσπελάσουν τα αντικείμενα και μαθαίνουν την τοποθεσία των αντικειμένων αυτών. Το Panasas έχει και ένα κόμβο, ο οποίος διατηρεί πληροφορίες για την σύνθεση του συστήματος. Αυτός ο κόμβος είναι υπεύθυνος να ελέγχει τις υπόλοιπες υπηρεσίες και κόμβους του συστήματος. Τα μεταδεδομένα του συστήματος διατηρούνται στα αντικείμενα και όχι σε κάποια ξεχωριστή βάση δεδομένων.

Το FAB υλοποιείται με μια συλλογή από *bricks*. Τα bricks είναι μικρά υπολογιστικά συστήματα και αποτελούνται από δίσκους, επεξεργαστή και NVRAM [14]. Με τα bricks κατανέμονται τα δεδομένα και οι λειτουργίες στους κόμβους του συστήματός. Συνολικά, παρέχουν ένα σύνολο λογικών τόμων στους πελάτες, οι οποίοι για να προσπελάσουν αυτά τα δεδομένα χρησιμοποιούν κάποιο πρωτόκολλο, όπως είναι το iSCSI. Το FAB παρέχει υπηρεσίες που δεν επηρεάζονται από τυχόν αποτυχίες που συμβαίνουν στο σύστημα. Προκειμένου να προσφέρει αξιοπιστία στους πελάτες χρησιμοποιεί δυο μεθόδους, τη μέθοδο του πλεονασμού και τη μέθοδο του αντικατοπτρισμού. Και οι δυο βασίζονται στην ιδέα της ψηφοφορίας, όπου κάθε αίτημα εκτελείται αφού λάβει απάντηση από την πλειοψηφία των *bricks* που αποθηκεύουν δεδομένα. Σε υψηλό επίπεδο, ένα αίτημα ανάγνωσης ή εγγραφής υποβάλλεται σε επεξεργασία ακολουθώντας

τα παρακάτω βήματα. Ο συντονιστής βρίσκει ένα σύνολο από *bricks* που έχουν αποθηκεύσει το ζητούμενο μπλοκ και εκτελούν είτε το πρωτόκολλο για τον αντικατοπτρισμό είτε το πρωτόκολλο για την τεχνική πλεονασμού στους κόμβους που αποθηκεύουν δεδομένα. Κάθε κόμβος που έχει αποθηκευμένο το μπλοκ που ζητείται μετατρέπει το αίτημα σε φυσική διεύθυνση στο δίσκο για να προσπελάσει τα δεδομένα.

Στην περίπτωση που χρησιμοποιείται το πρωτόκολλο για τον αντικατοπτρισμό, όταν εγγράφονται δεδομένα, ο συντονιστής παράγει μια μοναδική χρονοσφραγίδα και αποθηκεύει το νέο μπλοκ και την χρονοσφραγίδα στην πλειοψηφία των *bricks* που αποθηκεύουν δεδομένα. Όταν έχουμε ένα αίτημα ανάγνωσης, διαβάζει από την πλειοψηφία των κόμβων και επιστρέφει το αποτέλεσμα με τη πιο πρόσφατη χρονοσφραγίδα. Το FAB επίσης υποστηρίζει μια μέθοδο πλεονασμού η οποία χρησιμοποιεί τον αλγόριθμο (m,n) Reed-Solomon. Αυτός ο αλγόριθμος δέχεται ως είσοδο *m* μπλοκ δεδομένων και παράγει *(n-m)* μπλοκ πλεονάζουσας πληροφορίας από *m* μπλοκ δεδομένων και μπορεί να αναπαράγει την αρχική πληροφορία με οποιαδήποτε m από τα παραπάνω *n* μπλοκ. Όπως και όταν χρησιμοποιείται το πρωτόκολλο του αντικατοπτρισμού, έτσι και εδώ, κάθε αίτημα επικοινωνεί με ένα υποσύνολο των *bricks* που περιέχουν το μπλοκ που ζητείται. Εντούτοις, όταν χρησιμοποιείται το πρωτόκολλο του πλεονασμού ο συντονιστής πρέπει να λάβει απάντηση από $m+\lceil(n-m)/2\rceil$ bricks, το οποίο είναι η τομή οποιονδήποτε δυο πλειοψηφιών που περιέχει τουλάχιστον m bricks.

2.3. Αποδιπλοτυποποίηση κατά την αποθήκευση

Η αποδιπλοτυποποιημένη αποθήκευση μπορεί να θεωρηθεί ως μια καινούργια γενιά αποθηκευτικών συστημάτων για την προστασία δεδομένων που αντικαθιστά τις βιβλιοθήκες ταινιών (tape libraries) [18]. Η αποδιπλοτυποποίηση καταργεί τα τμήματα με πλεονάζουσα πληροφορία και συμπιέζει τα δεδομένα για να τα αποθηκεύσει στο δίσκο, εξοικονομώντας όσο το δυνατόν περισσότερο αποθηκευτικό χώρο στο δίσκο.

Το σύστημα Venti είναι ένα δικτυακό σύστημα αποθήκευσης που χρησιμοποιεί ως μονάδα αποθήκευσης το μπλοκ και έχει ως σκοπό την αρχειοθέτηση των δεδομένων [13]. Η διεπαφή που προσφέρει το σύστημα είναι ένα απλό πρωτόκολλο, που επιτρέπει στις εφαρμογές των πελατών να διαβάζουν και να εγγράφουν μπλοκ δεδομένων μεταβλητού μεγέθους. Το Venti δεν παρέχει τις υπηρεσίες ενός συστήματος αρχείων και εφεδρείας, αλλά προσφέρει το υπόβαθρο για αρχειοθετούμενη αποθήκευση στις εφαρμογές που το χρησιμοποιούν. Το Venti προσδιορίζει τα μπλοκ δεδομένων χρησιμοποιώντας μια συνάρτηση κατακερματισμού στα δεδομένα τους. Χρησιμοποιώντας μια συνάρτηση κατακερματισμού με ανθεκτικότητα στις συγκρούσεις, η οποία προσφέρει μια επαρκώς μεγάλη τιμή εξόδου, εξασφαλίσει τη μοναδικότητα στα αποτελέσματα που παράγει για κάθε μπλοκ. Το αποτέλεσμα της συνάρτησης κατακερματισμού για κάθε μπλοκ ονομάζεται μοναδικό αναγνωριστικό και μπορεί να χρησιμοποιηθεί για το προσδιορισμό του μπλοκ, ώστε να μπορούν να εκτελεστούν σε αυτό αιτήματα ανάγνωσης και εγγραφής. Αυτή η προσέγγιση εξασφαλίζει μια σειρά από ιδιότητες για το αποθηκευτικό σύστημα. Καθώς ένα μπλοκ μπορεί να προσδιοριστεί από ένα μοναδικό αποτύπωμα με βάση τα περιεχόμενα του, το μπλοκ δεν μπορεί να τροποποιηθεί χωρίς να τροποποιηθεί η διεύθυνση του. Επιπλέον, πολλαπλές τροποποιήσεις των ίδιων δεδομένων μπορούν να ομαδοποιηθούν και να μην χρειαστεί επιπλέον αποθηκευτικός χώρος. Η συνάρτηση κατακερματισμού παράγει ένα ονοματοχώρο για τα μπλοκ. Χωρίς συνεργασία και συντονισμό, πολλοί πελάτες μπορούν να μοιράζονται σε αυτόν τον ονοματοχώρο τον ίδιο διακομιστή Venti.

Το Data Domain File System (DDFS) [18] χωρίζει κάθε αρχείο σε ένα αριθμό από τμήματα μεταβλητού μεγέθους. Το μέγεθος τους εξαρτάται από το περιεχόμενο του αρχείου και κάθε τμήμα έχει ένα μοναδικό αναγνωριστικό. Το DDFS χρησιμοποιεί το αναγνωριστικό για να προσδιορίσει τμήματα με το ίδιο περιεχόμενο και ως ένα μέρος του περιγραφέα του τμήματος ο οποίος χρησιμοποιείται για να αναφερθούν στο τμήμα. Κάθε αρχείο προσδιορίζεται ως μια σειρά περιγραφέων των τμημάτων του. Κατά τη διάρκεια μιας εγγραφής, το DDFS προσδιορίζει τα τμήματα με το ίδιο περιεχόμενο και αποθηκεύει μόνο το ένα αντίγραφο για κάθε τμήμα. Πριν την αποθήκευση του νέου τμήματος, το DDFS χρησιμοποιεί την απόκλιση του Ziv-Lempel αλγορίθμου για την

συμπίεση του τμήματος. Το DDFS χρησιμοποιεί διάφορες τεχνικές για να μειώσει τη συμφόρηση στο δίσκο. Το συνοπτικό διάνυσμα (summary vector) είναι μια συνοπτική δομή δεδομένων, που βρίσκεται στη μνήμη, η οποία χρησιμοποιείται για τον προσδιορισμό νέων τμημάτων. Η δομή των τμημάτων για ροές δεδομένων είναι μια μέθοδος οργάνωσης της πληροφορίας για να βελτιωθεί ο τρόπος με τον οποίο αποθηκεύεται η πληροφορία στο δίσκο για σειριακές προσβάσεις τμημάτων. Η παραπάνω οργάνωση της πληροφορίας στο δίσκο επιτρέπει στην κρυφή μνήμη να διατηρεί τη τοποθεσία στον δίσκο, των αναγνωριστικών των τμημάτων με το ίδιο περιεχόμενο επιτυγχάνοντας μεγάλη αναλογία επιτυχιών.

2.4. Διακομιστές Ανάκτησης Πολυμέσων

Μια πρώιμη σχεδίαση ενός διακομιστή ανάκτησης πολυμέσων [5] αναλαμβάνει σταθερού ρυθμού δεδομένα μια ροής και υποστηρίζει πολλαπλές ροές με τον ίδιο σταθερό ρυθμό. Ο βασικός σχεδιασμός του RAID για ροές δεδομένων χρησιμοποιεί αλγορίθμους για την προσπέλαση δίσκου οι οποίοι μεγιστοποιούν τον αριθμό των ταυτόχρονων ροών δεδομένων που μπορούν να υποστηριχθούν από ένα διακομιστή aανάκτησης πολυμέσων. Έχει προταθεί ένας κυκλικός τρόπος λειτουργίας, στον οποίο εκτελείται μόνο μια εντολή Εισόδου/Εξόδου για κάθε ροή. Έτσι, ο αριθμός των ταυτόχρονων ροών δεδομένων που μπορούν να υποστηριχθούν αποφασίζεται από τον αριθμό των λειτουργιών Εισόδου/Εξόδου που μπορούν να εκτελεστούν σε ένα κύκλο. Όταν, ένα σύστημα, που έχει ένα δίσκο, αποθηκεύει ροές δεδομένων, η ανάγνωση των δεδομένων τους από το δίσκο είναι σειριακή, οπότε σε κάθε κύκλο μπορούμε να ξέρουμε τα δεδομένα που θα ζητηθούν στον επόμενο κύκλο. Έτσι, σε κάθε κύκλο μπορούμε να ανακτούμε από το δίσκο στη μνήμη τα δεδομένα που θα ζητηθούν στον επόμενο κύκλο. Επομένως, οι λειτουργίες Εισόδου/Εξόδου που εκτελούνται σε ένα κύκλο ταξινομούνται με βάση τη θέση τους στο δίσκο. Αυτή η ιδέα μπορεί να επεκταθεί και σε ένα σύστημα με πολλούς δίσκους, όπου τα δεδομένα κάθε ροής έχουν χωριστεί σε τμήματα και κάθε τμήμα έχει αποθηκευτεί σε ένα δίσκο. Ο μέγιστος αριθμός ταυτόχρονων ροών δεδομένων που μπορούν να υποστηριχθούν εξαρτάται από τον αριθμό των λειτουργιών

Εισόδου/Εξόδου που εκτελούνται σε ένα κύκλο. Καθώς αυξάνουμε την ποσότητα των δεδομένων που φέρνουμε στη μνήμη σε ένα κύκλο τόσο αυξάνονται και οι απαιτήσεις σε μνήμη του συστήματος.

Για να παρέχουν ανοχή σε σφάλματα τα συστήματα αυτά χωρίζουν τους δίσκους σε ομάδες, τις οποίες ονομάζουν συστοιχίες [2]. Στη συνέχεια κάθε δίσκος σε μια συστοιχία αποθηκεύει ένα μπλοκ που περιέχει πλεονάζουσα πληροφορία. Το μπλοκ αυτό υπολογίζεται εκτελώντας την πράξη XOR (EXCLUSIVE OR) στα εναπομείναντα μπλοκ των δίσκων που ανήκουν σε αυτή τη συστοιχία. Κατά τη διάρκεια μιας συνηθισμένης πράξης ανάγνωσης το σύστημα διαβάζει ολόκληρη την ομάδα των μπλοκ που ανήκουν σε μια συστοιχία. Στην περίπτωση που ένας δίσκος καταρρεύσει, το σύστημα χρησιμοποιεί το μπλοκ που περιέχει την πλεονάζουσα πληροφορία για να ανακτήσει τα δεδομένα του μπλοκ του δίσκου που κατέρρευσε. Για να μπορέσουν να επιτύχουν εξισορρόπηση φορτίου, κατανέμουν το φορτίο που περιείχε ένας δίσκος που κατέρρευσε σε όλους τους υπόλοιπους [11]. Εναλλακτικά, μπορεί να χρησιμοποιηθεί η μέθοδος του αντικατοπτρισμού, στην οποία το αντίγραφο του μπλοκ ενός δίσκου χωρίζεται σε τμήματα τα οποία κατανέμονται σε περισσότερους από έναν άλλους δίσκους. Με αυτή τη τεχνική, το φορτίο του δίσκου που καταρρέει δεν τον επωμίζεται ένας κόμβος αλλά περισσότεροι [5]. Ένας άλλος τρόπος να κατανεμηθούν τα αντίγραφα ενός μπλοκ στους δίσκους είναι αυτός με τν χρήση της μεθόδου εκ περιτροπής [10]. Η παραπάνω προσέγγιση έχει επεκταθεί και σε συστήματα τα οποία υποστηρίζουν ροών δεδομένων μεταβλητού ρυθμού [1].

ΚΕΦΑΛΑΙΟ 3. ΘΕΩΡΗΤΙΚΟ ΥΠΟΒΑΘΡΟ

3.1. Διαδικασίες απομακρυσμένης κλήσης

Το υπολογιστικό μοντέλο πελάτη-διακομιστή είναι ένα δημοφιλές μοντέλο για κατανεμημένη επεξεργασία. Μπορεί να χρησιμοποιηθεί σε οποιοδήποτε επεξεργαστικό περιβάλλον, όπου ένα σύνολο οντοτήτων ζητάει να υλοποιηθεί μια εργασία και ένα άλλο σύνολο εκτελεί αυτή την εργασία [4].

Η ιδέα της απομακρυσμένης κλήσης διαδικασίας (Remote Procedure Call-RPC) βασίζεται στην παρατήρηση ότι η κλήση διαδικασιών είναι ένας γνωστός και κατανοητός μηχανισμός για την μεταφορά του ελέγχου και των δεδομένων σε ένα πρόγραμμα το οποίο τρέχει σε ένα υπολογιστή [3]. Επομένως, ο ίδιος μηχανισμός επεκτείνεται για την μεταφορά του ελέγχου και των δεδομένων μεταξύ δυο κόμβων μέσω ενός δικτύου επικοινωνίας. Όταν μια απομακρυσμένη διαδικασία καλείται, το περιβάλλον που καλεί την διαδικασία μπλοκάρει, οι παράμετροι της διαδικασίας μεταβιβάζονται μέσω δικτύου στο περιβάλλον όπου θα εκτελεστεί η διαδικασία. Όταν, η διαδικασία τερματίσει και έχει παράγει το αποτέλεσμα της, το αποτέλεσμα μεταβιβάζεται στο περιβάλλον που αναμένει το αποτέλεσμα της κλήση και στη συνέχεια το περιβάλλον αυτό συνεχίζει την εκτέλεση του, με τον ίδιο τρόπο όπως σε μια τοπική κλήση μιας διαδικασίας. Όσο, το περιβάλλον που περιμένει την απάντηση από την απομακρυσμένη διαδικασία είναι μπλοκαρισμένο, όλες οι άλλες διαδικασίες που βρίσκονται στο ίδιο μηχάνημα εκτελούνται κανονικά.

Όταν ένας χρήστης επιθυμεί να εκτελέσει μια απομακρυσμένη κλήση εκτελεί μια τοπική κλήση η οποία καλεί την αντίστοιχη διαδικασία του *user stub* [3]. Το user stub είναι υπεύθυνο να καθορίσει την διαδικασία που πρέπει να κληθεί και να στείλει τις παραμέτρους αυτής της διαδικασίας σε ένα ή περισσότερα μηνύματα. Τα μηνύματα αυτά τα αναθέτει στο υποσύστημα του RPC που είναι υπεύθυνο για την επικοινωνία μεταξύ διαφορετικών κόμβων, για να τα στείλει στο κόμβο όπου θα κληθεί η διαδικασία. Τα μηνύματα αυτά λαμβάνονται από το *server stub* το οποίο βρίσκεται στο μηχάνημα όπου θα εκτελεστεί πραγματικά η διαδικασία. Το server stub παίρνει την πληροφορία που βρίσκεται στα μηνύματα που έχει λάβει και εκτελεί μια τοπική κλήση, η οποία καλεί την αντίστοιχη διαδικασία του διακομιστή. Όσο, γίνεται η διαδικασία αυτή η διεργασία που βρίσκεται στο μηχάνημα που κάλεσε την απομακρυσμένη διαδικασία είναι μπλοκαρισμένη. Όταν, στον διακομιστή ολοκληρωθεί η κλήση της διαδικασίας, ο διακομιστής επιστρέφει το αποτέλεσμα της στο server stub το οποίο θα αναλάβει να το στείλει στο user stub. Το user stub λαμβάνει το αποτέλεσμα και το επιστρέφει στην διαδικασία που το αναμένει.

3.2. Κωδικοποίηση διόρθωσης σφαλμάτων

Έστω n συσκευές αποθήκευσης $D_1, D_2, \ldots, D_n$, καθεμία από τις οποίες διατηρεί k bytes. Αυτές ονομάζονται *συσκευές δεδομένων*. Έστω m συσκευές αποθήκευσης $C_1, C_2, \ldots, C_m$, καθεμία από τις οποίες διατηρεί k bytes. Αυτές ονομάζονται *συσκευές πλεονάζουσας πληροφορία.* Το περιεχόμενο κάθε συσκευής πλεονάζουσας πληροφορία υπολογίζεται από το περιεχόμενο των συσκευών δεδομένων. Στόχος είναι να οριστεί ο υπολογισμός του περιεχομένου για κάθε C_i έτσι ώστε αν οποιεσδήποτε m συσκευές από τις $D_1, D_2, \ldots, D_n, C_1, C_2, \ldots, C_m$, καταρρεύσει να μπορεί να ανακτηθεί το περιεχόμενο των συσκευών που κατέρρευσαν από τις υπόλοιπες [12].

3.2.1. Μπλοκ Κώδικες

Όλοι οι κώδικες διόρθωσης σφαλμάτων βασίζονται στην αρχή της πρόσθεσης επιπλέον πληροφορίας με στόχο την διόρθωση τυχόν σφαλμάτων που μπορεί να συμβούν σε μια διαδικασία μεταφοράς ή αποθήκευσης δεδομένων [9,8]. Τα επιπλέον σύμβολα που παράγονται προστίθενται στο τέλος των πραγματικών συμβόλων δεδομένων για να παρέχουν μια κωδικοποιημένη ακολουθία ή *κωδικολέξη*. Οι *μπλοκ κώδικες* επεξεργάζονται τα δεδομένα ανά μπλοκ και η πληροφορία κάθε μπλοκ είναι ανεξάρτητη από την πληροφορία που περιέχουν τα άλλα μπλοκ. *Συστηματική κωδικοποίηση* σημαίνει ότι τα σύμβολα δεδομένων εμφανίζονται πάντα στις πρώτες *k* θέσεις μιας κωδικολέξης (από αριστερά). Στις επόμενες *n-k* θέσεις μιας κωδικολέξης αποθηκεύονται τα σύμβολα της πλεονάζουσας πληροφορίας η οποία θα χρησιμοποιηθεί για την ανίχνευση και τη διόρθωση σφαλμάτων. Το σύνολο των κωδικολέξεων ονομάζεται *κώδικας διόρθωσης σφαλμάτων* και θα τον δηλώνουμε με το σύμβολο *C*.

Πιο επίσημα, θεωρούμε μια πηγή η οποία παράγει σύμβολα από ένα αλφάβητο F_q, το οποίο αποτελείται από q σύμβολα. Το F_q συμβολίζει ένα πεδίο που απεικονίζεται από τα q σύμβολα. Αναφερόμαστε σε μια πλειάδα $(c_0, c_1, \ldots, c_{n-1}) \in F_q^n$ με n στοιχεία ως ένα διάνυσμα n διαστάσεων. Στη συνέχεια, ένας (n,k) μπλοκ κώδικας C σε ένα αλφάβητο με q σύμβολα είναι ένα σύνολο από q^k διανύσματα n-διαστάσεων ονομάζονται κωδικολέξεις. Ο κωδικοποιητή απεικονίζει ένα μήνυμα $m \in F_q^k$ στην αντίστοιχη κωδικολέξη.

Θεωρούμε κώδικα διόρθωσης σφαλμάτων C με q στοιχεία από το αλφάβητο F_q. Για να μπορέσουμε να πετύχουμε την διόρθωση σφαλμάτων δεν χρειάζεται να έχουμε όλα τα q^n δυνατά διανύσματα μεγέθους n. Αντί για αυτό, το C ορίζεται ως ένα υποσύνολο των διανυσμάτων n διαστάσεων του χώρου F_q^n, με τα στοιχεία να είναι όσο το δυνατόν διαφορετικά μεταξύ τους.

Θεωρούμε δυο διανύσματα $\overline{c}_1 = \left(c_{1,0}, c_{1,1}, \ldots, c_{1,n-1}\right)$ και το $\overline{c}_2 = \left(c_{2,0}, c_{2,1}, \ldots, c_{2,n-1}\right)$ τα οποία ανήκουν στο F_q^n. Η απόσταση *Hamming* ανάμεσα στο $\overline{c}_1$ και το $\overline{c}_2$, συμβολίζεται $d_H\left(\overline{c}_1, \overline{c}_2\right)$, και ορίζεται ως ο αριθμός των στοιχείων στα οποία τα διανύσματα διαφέρουν,

$$d_H\left(\overline{c}_1, \overline{c}_2\right) = \left|\left\{\imath : c_{1,i} \neq c_{2,i}, 0 \leq i < n\right\}\right| \quad (1)$$

όπου $|A|$ συμβολίζει τον αριθμό των στοιχείων στο σύνολο *A*. Δοθέντος ενός κώδικά *C*, η μικρότερη απόσταση *Hamming* $d_{\min}$ ορίζεται ως η ελάχιστη απόσταση *Hamming* μεταξύ όλων των πιθανών διακριτών ζευγών κωδικολέξεων στον C:

$$d_{\min} = \min_{\overline{c}_1, \overline{c}_2 \in C} \left\{ d_H\left(\overline{c}_1, \overline{c}_2 \middle| \overline{c}_1 \neq \overline{c}_2\right)\right\}$$

Ένας κώδικας στον οποίο το $d_{\min} = n - k + 1$ ονομάζεται κώδικας *Maximum Distance Separable* (MDS). Θεωρούμε ότι το μέγεθος ενός κώδικα *C* είναι $|C| = q^k$. Έτσι, ο υπολογισμός της μικρότερης απόστασης $d_{\min}$ ενός μπλοκ κώδικα C απαιτεί $q^{k-1}\left(q^k - 1\right)$ αποστάσεις ανάμεσα στα διακριτά ζεύγη των κωδικολέξεων του. Η πλειάδα $\left(n, k, d_{\min}\right)$ χρησιμοποιείται για να προσδιορίσει τις παραμέτρους ενός μπλοκ κώδικα μεγέθους *n*, ο οποίος κωδικοποιεί μηνύματα μεγέθους *m* και έχει ελάχιστη απόσταση *Hamming* $d_{\min}$. Η πλεονάζουσα πληροφορία που παράγει ο κώδικας είναι ο αριθμός των συμβόλων πλεονασμού σε μια κωδικολέξη. Πιο συγκεκριμένα έχουμε:

$$r = n - \log_q |C|$$

Με $\overline{c}$ συμβολίζεται μια κωδικολέξη ενός κώδικα διόρθωσης σφαλμάτων *C*. *Σφαίρα Hamming* $S_t\left(\overline{c}\right)$, με ακτίνα *t* και κέντρο το $\overline{c}$, είναι ένα σύνολο διανυσμάτων στον F_q^n με απόσταση μικρότερη ή ίση του *t* από το κέντρο $\overline{c}$

$$S_t(\bar{c}) = \left\{ \bar{x} \in V_2 \middle| d_H(\bar{x}, \bar{c}) \le t \right\}$$

Η δυνατότητα διόρθωσης σφαλμάτων, *t*, ενός κώδικα *C* είναι η μέγιστη ακτίνα της *Σφαίρας Hamming* $S_t(\bar{\upsilon})$ μεταξύ όλων των κωδικολέξεων $\bar{\upsilon} \in C$. Έτσι για όλα τα διαφορετικά ζεύγη, $\bar{c}_i, \bar{c}_j \in C$ η σχετική *Σφαίρα Hamming* είναι κενή, δηλαδή,

$$t = \max \left\{ l \middle| S_l(\bar{c}_i) \cap S_l(\bar{c}_j) = \varnothing, \bar{c}_i \le \bar{c}_j \right\}_{\bar{c}_1, \bar{c}_2 \in C}$$

Ισοδύναμα, η δυνατότητα διόρθωσης σφαλμάτων μπορεί να εκφραστεί σε συνάρτηση της μικρότερης απόστασης του *C*, $d_{\min}$ με τον ακόλουθο τύπο:

$$t = \lfloor (d_{\min} - 1)/2 \rfloor$$

όπου $\lfloor x \rfloor$ συμβολίζει τον μέγιστο ακέραιο που είναι μικρότερος ή ίσος του *x*.

3.2.2. Γραμμικοί Μπλοκ Κώδικες

Ένας μπλοκ κώδικας *C* σε ένα χώρο F_q, με *q* σύμβολα μεγέθους *n* και q^k κωδικολέξεις είναι ένα *q*-αδικός γραμμικός κώδικας (n,k), αν και μόνο αν οι q^k κωδικολέξεις σχηματίζουν ένα διανυσματικό υποχώρο *k* διαστάσεων στον διανυσματικό χώρο των πλειάδων μεγέθους *n* του F_q^n. Ο αριθμός *n* προσδιορίζει το μέγεθος του κώδικα και ο αριθμός *k* τη διάσταση του κώδικα. Ο ρυθμός του κώδικα *k* είναι $R = k/n$. Το *βάρος Hamming* $wt(c)$ μιας κωδικολέξης *c* είναι ο αριθμός των μη-μηδενικών στοιχείων της κωδικολέξης. Το ελάχιστο βάρος $w_{\min}$ ενός κώδικα *C* είναι το μικρότερο βάρος Hamming οποιασδήποτε μη-μηδενικής κωδικολέξης:

$w_{\min} = \min_{c \in C, c \neq o} wt(c)$. Για ένα γραμμικό αλγόριθμο *C*, η ελάχιστη απόσταση $d_{\min}$ ικανοποιεί την ισότητα $d_{\min} = w_{\min}$. Αυτό σημαίνει ότι η μικρότερη απόσταση ενός γραμμικού μπλοκ κώδικα είναι ίσος με το μικρότερο βάρος από τις $2^k - 1$ μη-μηδενικές κωδικολέξεις.

Επειδή, ένας γραμμικός μπλοκ κώδικας *C* είναι ένας διανυσματικός χώρος *k* διαστάσεων, υπάρχουν *k* γραμμικά ανεξάρτητα διανύσματα τα οποία σχεδιάζονται ως $\bar{g}_0, \bar{g}_1, \ldots \bar{g}_{k-1}$, έτσι ώστε κάθε κωδικολέξη $\bar{c}$ του *C* να μπορεί να αναπαρασταθεί ως γραμμικός συνδυασμός αυτών των διανυσμάτων,

$$\bar{c} = m_0 \bar{g}_0 + m_1 \bar{g}_1 + \ldots + m_{k-1} \bar{g}_{k-1}$$

όπου $m_i \in F_q$. Κάθε $\bar{g}_i$ μπορεί να θεωρηθεί ως μια γραμμή ενός πίνακα και βάζοντας τα όλα σε ένα πίνακα δίνουν ένα $k \times n$ πίνακα *G*,

$$G = \begin{bmatrix} \bar{g}_0 \\ \bar{g}_1 \\ \vdots \\ \bar{g}_{k-1} \end{bmatrix}$$

Έστω

$$\bar{m} = \begin{bmatrix} m_0 & m_1 & \cdots & m_{k-1} \end{bmatrix}$$

Έτσι προκύπτει ότι

$$\bar{c} = \bar{m} G$$

και κάθε κωδικολέξη $c \in C$ μπορεί να αναπαρασταθεί σε συνδυασμό κάποιου διανύσματος $\bar{m}$. Επειδή οι γραμμές του πίνακα G παράγουν τον $(n, k, d_{\min})$ γραμμικό

κώδικα *C*, το *G* ονομάζεται *γεννήτορας πίνακας* για τον *C*. Η παραπάνω ισότητα μπορεί να θεωρηθεί ως η πράξη κωδικοποίησης για τον κώδικα *C*. Λόγω του ότι ο *C* είναι ένας διανυσματικός χώρος *k* διαστάσεων που ανήκει στο F_q^n, υπάρχει ένας *δυικός χώρος* C^T $(n-k)$ διαστάσεων, που παράγεται από ένα πίνακα Η ο οποίος ονομάζεται *πίνακας-ελέγχου ισοτιμίας* για τον κώδικα *C*, τέτοιος ώστε $CH^T = 0$, όπου H^T ο ανάστροφος του πίνακα *H*. Πιο συγκεκριμένα, κάθε διάνυσμα $\overline{u} \in F_q^n$ είναι κωδικολέξη $u \in C$, αν και μόνο αν

$$uH^T = \overline{0}$$

Έστω $\overline{r}$ ένα διάνυσμα *n* διαστάσεων στον χώρο F_q και έστω *H* ένας πίνακας-ελέγχου ισοτιμίας του κώδικα *C*. Το διάνυσμα

$$\overline{u} = \overline{r}H^T$$

ονομάζεται *σύνδρομο* του $\overline{r}$. Το *σύνδρομο* μπορεί να χρησιμοποιηθεί για την ανίχνευση σφαλμάτων. Έστω μια κωδικολέξη $\overline{c}$ του γραμμικού κώδικα *C* στον χώρο F_q μεταφέρεται μέσο ενός καναλιού και λαμβάνεται το διάνυσμα *n* διαστάσεων $\overline{r}$. Ισχύει ότι:

$$\overline{r} = \overline{c} + \overline{e}$$

Όπου η παραπάνω πράξη πραγματοποιείται στο χώρο F_q, και το $\overline{e}$ είναι ένα *διάνυσμα σφάλματος*, το οποίο έχει *0* στις θέσεις ποτ το κανάλι δεν εισήγαγε σφάλματα. Το διάνυσμα $\overline{r}$ που έχει ληφθεί μπορεί να είναι οποιοδήποτε διάνυσμα του F_q^n, καθώς οποιοδήποτε σφάλμα είναι πιθανό. Έστω *H* είναι ένας πίνακας-ελέγχου ισοτιμίας του *C*. Τότε το σύνδρομο

$$\overline{s} = \overline{r}H^T = \left(\overline{c} + \overline{e}\right)H^T = \overline{e}H^T$$

Επομένως, $\overline{s} = 0$, αν το διάνυσμα $\overline{r}$ είναι μια κωδικολέξη. Όμως, αν $\overline{r} \neq \overline{0}$ τότε πρέπει να έχει συμβεί κάποιο σφάλμα. Δεν ξέρουμε τι σφάλμα έχει συμβεί, αλλά ξέρουμε ότι υπάρχει σφάλμα.

3.2.3. Αποκωδικοποίηση

Ορίζουμε ως *πρότυπο πίνακα* για ένα $\left(n, k, d_{\min}\right)$ γραμμικό κώδικα *C* ένα πίνακα ο οποίος περιέχει όλα τα πιθανά διανύσματα $\overline{r}$ που μπορούν να ληφθούν, οργανώνοντας τα με τέτοιο τρόπο ώστε να βρεθεί η κωδικολέξη $\overline{u}$ που είναι πιο κοντά στο $\overline{r}$. Ο πρότυπος πίνακας περιέχει q^{n-k} σειρές και $2^k + 1$ στήλες. Τα διανύσματα που περιέχονται στις q^k δεξιότερες στήλες του πίνακα περιέχουν όλα τα διανύσματα του F_q^n. Ένας πρότυπος πίνακας *C* δημιουργείται ακολουθώντας την παρακάτω διαδικασία:

1. Έστω *j=0*. Στην πρώτη θέση της πρώτης γραμμής του πίνακα τοποθετούμε το μηδενικό σύνδρομο. Στις q^k δεξιότερες θέσεις της πρώτης γραμμής τοποθετούμε όλες τις κωδικολέξεις του C.
2. Έστω *j=j+1*. Βρίσκουμε το $\overline{e} \in F_q^n$ με τη μικρότερο βάρος Hamming, όπου το $\overline{e}$ να μην ανήκει στον *C* και δεν περιέχεται στις προηγούμενες γραμμές. Τοποθετούμε στη πρώτη θέση της γραμμής, το αντίστοιχο σύνδρομο $\overline{s} = \overline{e}H^T$. Στις επόμενες q^k θέσεις της γραμμής, το αποτέλεσμα της πρόσθεσης του $\overline{e}$ στις αντίστοιχες θέσεις της πρώτης γραμμής.
3. Επαναλαμβάνουμε την παραπάνω διαδικασία μέχρι όλα τα q^k διανύσματα του F_q^n να έχουν συμπεριληφθεί στον πίνακα, με $j < 2^{n-k}$.

Για παράδειγμα, ο δυαδικός $(4, 2, 2)$ γραμμικός κώδικας με πίνακα γεννήτορα στη συμμετρική μορφή που φαίνεται παρακάτω:

$$C = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix},$$

ενώ ο πίνακας ελέγχου ισοτιμίας είναι ο παρακάτω

$$H = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

Μετέπειτα έχουμε τις παρακάτω αντιστοιχίες μεταξύ των μηνυμάτων δυο bit και των κωδικολέξεων των τεσσάρων bit:

$$[00] \rightarrow [0000]$$
$$[01] \rightarrow [0110]$$
$$[10] \rightarrow [1011]$$
$$[11] \rightarrow [1101]$$

Ο πρότυπος πίνακας του κώδικα είναι ο ακόλουθος:

$\overline{s}$	00	01	10	11
00	0000	1011	1011	1101
11	1000	1110	0011	0101
10	0100	0010	1111	1001
01	0001	0111	1010	1100

Βρίσκουμε την λαμβανομένη λέξη $\overline{r} = \overline{c} + \overline{e}$ και παράγουμε ως έξοδο για αποκωδικοποιημένο μήνυμα το περιεχόμενο της πρώτης γραμμής του πίνακα που βρίσκεται στην ίδια στήλη με την $\overline{r}$. Αυτή διαδικασία απαιτεί να είναι αποθηκευμένος όλος ο πίνακας και αντιστοιχίζεται η λαμβανόμενη λέξη με κάποια θέση του πίνακα.

Ουσιαστικά, σε μια απλή διαδικασία αποκωδικοποίησης ακολουθεί τα παρακάτω βήματα:

1. Υπολογισμός του συνδρόμου $\bar{s} = (\bar{c} + \bar{e}) H^T = \bar{e} H^T$.
2. Στον πρότυπο πίνακα ψάχνουμε το μοτίβο σφάλματος $\bar{e}$ που αντιστοιχίζεται στο $\bar{s}$
3. Τότε $\bar{c} = \bar{r} + \bar{e}$

Για το αυτό λόγο χρειαζόμαστε μόνο τις δυο πρώτες στήλες (από αριστερά) του πρότυπου πίνακα. Για παράδειγμα, στέλνουμε τη κωδικολέξη $\bar{c} = [0110]$ και λαμβάνουμε $\bar{r} = [0010]$. Το σύνδρομο είναι

$$\bar{s} = \bar{r} H^T = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

Το αντίστοιχο διάνυσμα σφάλματος είναι $\bar{e}' = [0 \quad 1 \quad 0 \quad 0]$ και η υπολογιζόμενη κωδικολέξη είναι

$$\bar{c}' = \bar{r} + \bar{e}' = [0 \quad 0 \quad 1 \quad 0] + [0 \quad 1 \quad 0 \quad 0] = [0 \quad 1 \quad 1 \quad 0]$$

Η *διαγραφή* είναι ένα σφάλμα του οποίου η θέση είναι γνωστή, αλλά όχι η αρχική τιμή του. Η διόρθωση ενός σφάλματος απαιτεί γνώση και της θέσης του σφάλματος και της τιμή του σφάλματος, ενώ η ανάθεση τιμής στη διαγραφή απαιτεί μόνο τον προσδιορισμό της τιμής του σφάλματος. Γενικά, αν υπάρχουν *f* διαγραφές και *e* σφάλματα, μπορούν να διορθωθούν εάν ισχύει

$$2e + f < d_{\min}$$

Θεωρούμε τη συνήθη περίπτωση στην οποία έχουμε την δυαδική πληροφορία με γνωστό το κώδικα αποκωδικοποίησης. Ο κώδικας αποκωδικοποίησης με γνωστές τις θέσεις σφαλμάτων μπορεί να δημιουργηθεί με τον ακόλουθο τρόπο. Έστω $\bar{r}$ το λαμβανόμενο διάνυσμα.

1. Τοποθετούμε μηδέν σε όλες τις γνώστες θέσεις σφαλμάτων και ονομάζουμε $\bar{c}_0$ το αντίστοιχο αποτέλεσμα του αλγόριθμου αποκωδικοποίησης
2. Τοποθετούμε μονάδα σε όλες τις γνώστες θέσεις σφαλμάτων και ονομάζουμε $\bar{c}_1$ το αντίστοιχο αποτέλεσμα του αλγόριθμου αποκωδικοποίησης
3. Η πληροφορία που θα προκύψει είναι ένα από τα $\bar{c}_0$, $\bar{c}_1$ το οποίο περιέχει το μικρότερο αριθμός διόρθωσης σφαλμάτων σε σχέση με το $\bar{r}$

3.2.4. Κυκλική κώδικες και Πολυώνυμα

Έστω *C* ένας $(n, k, d_{\min})$ γραμμικός μπλοκ κώδικας. Επίσης, έστω $\bar{m}$ ένα μήνυμα και $\bar{c}$ η αντίστοιχη κωδικολέξη στον *C*. Μπορούμε να αντιστοιχίσουμε ένα *πολυώνυμο* $\bar{c}(x)$ με κάθε κωδικολέξη με τον ακόλουθο τρόπο:

$$\bar{c} = [c_0, c_1, \cdots, c_{n-1}] \rightarrow \bar{c}(x) = c_0 + c_1 x + \cdots + c_{n-1} x^{n-1}$$

όπου το x δηλώνει τις θέσεις ενός στοιχείου c_i του $\bar{c}$ με την μορφή $c_i x^i$ του $\bar{c}$, $0 \le i < n$. Ένας γραμμικός αλγόριθμος για μπλοκ C ονομάζεται *κυκλικός*, αν και μόνο αν κάθε κυκλική μετατόπιση μιας κωδικολέξης είναι μια άλλη κωδικολέξη:

$$\bar{c} = [c_0, c_1, \cdots, c_{n-1}] \in C \Leftrightarrow \bar{c}^{(1)} = [c_{n-1}, c_0, \cdots, c_{n-2}] \in C$$

Στην ορολογία των πολυωνύμων, μια κυκλική μετατόπιση κατά μια θέση, συμβολίζεται ως $\bar{c}^{(1)}(x)$ και επιτυγχάνεται με ένα πολλαπλασιασμό με το $x \bmod (x^n - 1)$:

$$\overline{c}(x) \in C \Leftrightarrow \overline{c}^{(1)} = x\overline{c}(x) \bmod \left(x^n - 1\right) \in C$$

Μια σημαντική ιδιότητα των κυκλικών κωδίκων είναι ότι όλα τα πολυώνυμα $\overline{c}$ είναι πολλαπλάσια ενός μοναδικού πολυωνύμου $\overline{g}(x)$, το οποίο ονομάζεται *πολυώνυμο γεννήτορας* του κώδικα. Αυτό το πολυώνυμο καθορίζεται από τις ρίζες του, οι οποίες ονομάζονται *μηδενικά του κώδικα*. Μπορεί να αποδειχτεί ότι το πολυώνυμο γεννήτορας $\overline{g}(x)$ διαιρεί το $\left(x^n - 1\right)$. Επομένως, για να βρεθεί το πολυώνυμο γεννήτορας, θα πρέπει το πολυώνυμο $\left(x^n - 1\right)$ να αναχθεί στα ελάχιστα πολυώνυμα $\varphi_j(x)$, $j = 1, 2, \cdots, l$,

$$\left(x^n - 1\right) = \varphi_1(x)\varphi_2(x)\cdots\varphi_l(x)$$

Επομένως, το πολυώνυμο $\overline{g}(x)$ προκύπτει από τη σχέση

$$\overline{g}(x) = \prod_{j \in J \subset \{1,2,\cdots,l\}} \varphi_j(x)$$

Έστω $\overline{m}(x)$ το μήνυμα που πρέπει να κωδικοποιηθεί. Ένας τρόπος με τον οποίο μπορεί να κωδικοποιηθεί το μήνυμα είναι με βάση την παρακάτω ισότητα:

$$\overline{c}(x) = \overline{m}(x)\overline{g}(x)$$

Ένα άλλο πολυώνυμο, $\overline{h}(x)$, ονομάζεται πολυώνυμο *ελέγχου πλεονάζουσας πληροφορίας*, το οποίο μπορεί να συσχετιστεί με τον πίνακα ελέγχου πλεονάζουσας πληροφορίας. Ο πίνακας γεννήτορας και το πολυώνυμο ελέγχου πλεονάζουσας πληροφορίας συσχετίζονται ως εξής

$$\overline{g}(x)\overline{h}(x) = x^n + 1$$

Σημειώνουμε ότι στα πεδία δυαδικών αριθμών, *α-β* και *α+β* (υπόλοιπο 2) δίνει το ίδιο αποτέλεσμα. Το πολυώνυμο ελέγχου πλεονάζουσας πληροφορίας μπορεί να υπολογιστεί ως εξής

$$\overline{h}(x) = \left(x^n + 1\right) / \overline{g}(x) = h_0 + h_1 x + \cdots h_k x^k$$

Έστω $\overline{r}(x) = \overline{c}(x) + \overline{e}(x)$, όπου $\overline{e}(x)$ είναι ένα πολυώνυμο σφάλματος που σχετίζεται με το διάνυσμα σφάλματος. Το *πολυώνυμο σύνδρομο* ορίζεται ως εξής

$$\overline{s}(x) = \overline{r}(x) \bmod \overline{g}(x) = \overline{e}(x) \bmod \overline{g}(x)$$

Ο αλγόριθμος αποκωδικοποίησης θα πρέπει να βρει το πολυώνυμο σφάλματος $\overline{e}(x)$ από το γνωστό πολυώνυμο σύνδρομο $\overline{s}(x)$. Υποθέτουμε ότι ένα σφάλμα εμφανίζεται σε μια θέση που σχετίζεται με το $\left(x^n - 1\right)$ (το πρώτο bit που ελήφθηκε):

$$\overline{e}(x) = x^{n-1}$$

Το αντίστοιχο πολυώνυμο σύνδρομο είναι

$$\overline{s}(x) = x^{n-1} \bmod \overline{g}(x)$$

Το σύνδρομο αποκωδικοποίησης ελέγχει το σύνδρομο για κάθε λαμβανόμενη θέση, και αν το μοτίβο $x^{n-1} \bmod \overline{g}(x)$ ανακαλυφθεί, αυτή η θέση διορθώνεται.

Ο BCH κώδικας ο οποίος πήρε το όνομα του από τους Bose, Ray-Chaudhuri και Hocquenghem για εργασίες που δημοσιεύτηκαν το 1959 και 1960. Ο BCH κώδικας για

το $d_{\min} \geq 2t_d + 1$ είναι ένας κυκλικός κώδικας του οποίου το πολυώνυμο γεννήτορας έχει $2t_d$ συνεχόμενες ρίζες $a^b, a^{b+1}, \cdots, a^{b+2t_d-1}$. Ο αλγόριθμος Reed-Solomon αποτελεί μια ειδική περίπτωση του BCH κώδικα, στον οποίο το πολυώνυμο γεννήτορας έχει γραμμικούς παράγοντες:

$$\overline{g}(x) = \prod_{j=b}^{b+2t_d-1} \left(x + a^j\right)$$

όπου *b* ένας ακέραιος, με *b=0* ή *b=1*. Ο αλγόριθμος Reed-Solomon ανήκει στους MDS αλγόριθμους, η οποία τους δίνει σημαντικές ιδιότητες. Συγκεκριμένα, ο αλγόριθμος μπορεί να διορθώσει μέχρι t_d τυχαία σφάλματα, πολλές ριπές σφαλμάτων, όπου κάθε ομάδα έχει μέχρι $m(t_d - 1) + 1$ λανθασμένα bit. Αυτός είναι ο λόγος που ο αλγόριθμος Reed-Solomon είναι πολύ δημοφιλής σε πολλές εφαρμογές.

ΚΕΦΑΛΑΙΟ 4. ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΣΥΣΤΗΜΑΤΟΣ

4.1. Πρόβλημα

Στόχος μας είναι να αναπτύξουμε ένα σύστημα το οποίο να μπορεί να δέχεται ροές δεδομένων και να τις αποθηκεύει αξιόπιστα και αποδοτικά, με βασική προϋπόθεση ότι ο κύριος όγκος των αιτημάτων που εξυπηρετεί θα είναι εγγραφές.

Μέχρι σήμερα έχουν αναπτυχθεί διάφορα συστήματα αποθήκευσης δεδομένων, τα οποία μπορούν να κατηγοριοποιηθούν σε διακομιστές ανάκτησης πολυμέσων και σε συστήματα γενικού σκοπού.

Το σύστημα που αναπτύξαμε στην παρούσα εργασία διαφέρει από τους διακομιστές ανάκτησης επειδή έχει σχεδιαστεί να εξυπηρετεί αιτήματα εγγραφής και όχι αιτήματα ανάγνωσης (playback).

Από την άλλη πλευρά το προτεινόμενο σύστημα διαφέρει από τα συστήματα γενικού σκοπού, επειδή αποθηκεύει ροές με μεταβλητό ρυθμό μετάδοσης, των οποίων το εύρος των ρυθμών είναι μεγαλύτερο σε σχέση με αυτό που καλύπτουν τα συστήματα γενικού σκοπού. Επίσης, το προτεινόμενο σύστημα αποθηκεύει κυρίως ροές δεδομένων. Τα νέα δεδομένα προστίθενται στο τέλος κάποιου αρχείου σε αντίθεση με τα συστήματα γενικού

σκοπού στα οποία η εγγραφή δεδομένων μπορεί να σημαίνει και επανεγγραφή παλαιότερων δεδομένων.

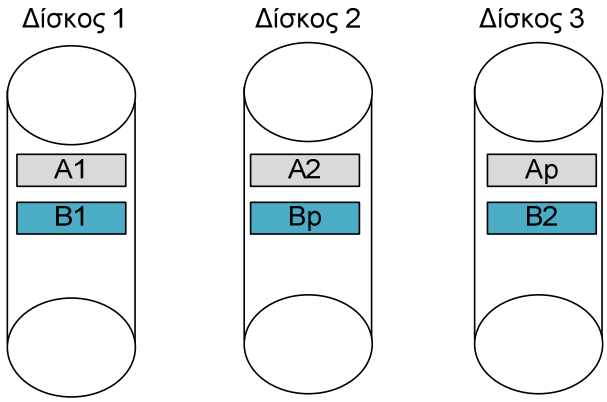
Άρα κανένα από τα υπάρχοντα συστήματα αποθήκευσης δεν έχει σχεδιαστεί για να μπορεί να αποθηκεύει ροές δεδομένων αξιόπιστα και αποδοτικά.

4.2. Σχεδιαστικές Αποφάσεις

Σκοπός της παρούσας εργασίας είναι να υλοποιηθεί ένα σύστημα αποθήκευσης το οποίο θα εκμεταλλεύεται τα χαρακτηριστικά των ροών δεδομένων για την επίλυση των διαφόρων σχεδιαστικών ζητημάτων που προκύπτουν. Το σύστημα μας θα πρέπει να δίνει τη δυνατότητα στους πελάτες του να ανακτούν και να αποθηκεύουν ροές ακόμα και αν συμβαίνουν αποτυχίες δίσκων ή κόμβων. Επίσης, το σύστημα μας έχει δυο επίπεδα κόμβων. Στο πρώτο επίπεδο βρίσκονται οι κόμβοι που μιλάνε με τους πελάτες, και στο δεύτερο αυτοί που αποθηκεύουν δεδομένα. Για κάθε ροή δεδομένων υπεύθυνος είναι ένας κόμβος του πρώτου επιπέδου.

Βασικός στόχος του συστήματος είναι να εξυπηρετεί αιτήματα αποθήκευσης με καλή απόδοση. Αυτό εξαρτάται από το πόσο γρήγορα μπορούν να βρεθούν τα μπλοκ τα οποία θα χρησιμοποιηθούν για την αποθήκευση μιας ροής και πόσοι κόμβοι χρησιμοποιούνται ταυτόχρονα για κάθε ροή. Στο σύστημα μας, η επιλογή των κόμβων, που θα χρησιμοποιηθούν για την αποθήκευση μιας ροής δεδομένων, γίνεται από τον κόμβο ο οποίος είναι υπεύθυνος για αυτή τη ροή. Αυτός επιλέγει τους κόμβους και δεσμεύει τα μπλοκ που θα χρησιμοποιηθούν για την αποθήκευση της ροής. Ο κόμβος θα λαμβάνει περιοδικά την συνολική εικόνα του συστήματος από ένα κόμβο ελέγχου και με βάση αυτή θα επιλέγει τους κόμβους που θα χρησιμοποιηθούν για την αποθήκευση. Για το δεύτερο παράγοντα, ο αριθμός των κόμβων που λαμβάνουν μέρος στην αποθήκευση μίας ροής θα είναι *N*, *N*-*K* κόμβοι που έχουν μπλοκ δεδομένων και ο *K*-οστός που περιέχει το μπλοκ πλεονασμού που προκύπτει από τα *N*-*K* μπλοκ δεδομένων (Σχήμα 4.1). Τέλος, επειδή ένας κόμβος είναι υπεύθυνος για την αποθήκευση μιας ροής, αυτός ο κόμβος θα

μπορεί να διατηρεί το μπλοκ πλεονασμού στη μνήμη του, γεγονός που συμβάλει στην βελτιωμένη απόδοση.



Σχήμα 4.1: Για κάθε ροή Α που χωρίζεται σε δυο μέρη, έστω Α1 και Α2 παράγεται η επιπλέον πληροφορία Ap

Το σύστημα μας θα πρέπει να μπορεί να μεγιστοποιεί το πλήθος των ταυτόχρονων ροών δεδομένων, που μπορεί να υποστηρίζει. Μια νέα ροή θα γίνεται δεχτή, μόνο αν δεν επηρεάζει την λειτουργία των άλλων ροών. Αυτή η απόφαση λαμβάνεται από το κόμβο ελέγχου και βασίζεται στο διαθέσιμο εύρος ζώνης. Ο κόμβος ελέγχου διατηρεί πληροφορία για κάθε διακομιστή διανομής, για το πόσο φορτωμένες είναι οι ουρές των αιτήσεων και με βάση αυτή τη πληροφορία αποφασίζει αν θα δεχτεί ένα νέο αίτημα. Τέλος, επειδή το προτεινόμενο σύστημα υποστηρίζει ροές διαφόρων ρυθμών μεταδόσεων, η εξισορρόπηση φόρτου είναι δύσκολο να επιτευχθεί. Ειδικότερα, το πρόβλημα προκύπτει σε χαμηλού ρυθμού μετάδοσης ροές, επειδή τότε προκύπτουν μικρού μεγέθους εγγραφές στο δίσκο. Κατά την επικοινωνία δύο κόμβων, μια ροή χαμηλού ρυθμού στέλνει λίγα δεδομένα και δημιουργεί υψηλή επιβάρυνση στο δίκτυο. Το προτεινόμενο σύστημα αντιμετωπίζει αυτό το πρόβλημα συναθροίζοντας δεδομένα διαφορετικών ροών στο ίδιο μήνυμα.

Σχεδιάσαμε το σύστημα αποθήκευσης έχοντας τους εξής στόχους :

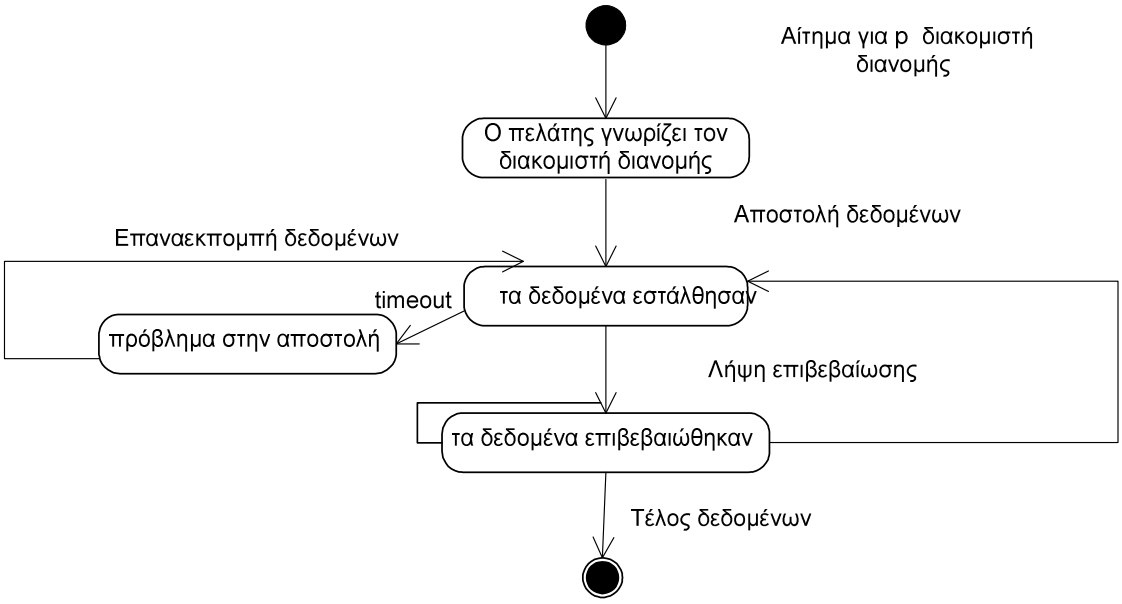
- Το σύστημα αποθηκεύει ροές δεδομένων διαφόρων ρυθμών μετάδοσης που μπορεί να αλλάζουν με το χρόνο είτε λόγω των απαιτήσεων της εφαρμογής ή εξαιτίας μεταβολών στην κατάσταση του δικτύου μεταφοράς.
- Το σύστημα εξυπηρετεί κυρίως ακολουθιακές εγγραφές, οι οποίες προσθέτουν δεδομένα στο τέλος κάποιου αρχείου. Το σύστημα μας μπορεί να εκτελεί αποδοτικά τις συγκεκριμένες λειτουργίες και ταυτόχρονα να εκτελεί και τυχαίες αναγνώσεις και εγγραφές δεδομένων με χαμηλότερη απόδοση
- Υποστηρίζει αιτήματα ανάγνωσης δεδομένων, αλλά αυτά τα αιτήματα είναι λιγότερο συχνά σε σχέση με τις εγγραφές. Στο σύστημα μας υπάρχει μόνο ένα αντίγραφο για δεδομένα, οπότε δεν μπορούμε να υποστηρίξουμε αποδοτικά πολλαπλά ταυτόχρονα αιτήματα ανάγνωσης για τα ίδια δεδομένα.
- Επειδή μπορεί να συμβούν αποτυχίες στους δίσκους ή σε ολόκληρους κόμβους, πρέπει να εξασφαλίσουμε στους πελάτες ότι δε θα χάνουν τα δεδομένα τους.
- Υποστηρίζει ένα αριθμό χρηστών, οι οποίοι θα μπορούν να υποβάλλουν ταυτόχρονα αιτήματα για αποθήκευση δεδομένων. Το σύστημα με βάση τους διαθέσιμους πόρους θα αποφασίζει αν θα δεχθεί ένα νέο αίτημα ή όχι.
- Ο διαθέσιμος αποθηκευτικός χώρος του συστήματος μας θα πρέπει να χρησιμοποιηθεί κατάλληλα ώστε να μην υπάρχουν κόμβοι οι οποίοι να είναι περισσότερο φορτωμένοι από τους υπόλοιπους, Ως εκ τούτου, θα πρέπει να χρησιμοποιηθεί μια κατάλληλη μέθοδος για την επίτευξη εξισορρόπησης φόρτου.
- Τα μεταδεδομένα του συστήματος μας θα πρέπει να είναι κατανεμημένα σε περισσότερους από ένα κόμβους. Έτσι, για κάθε ροή δεδομένων υπεύθυνος είναι ένας κόμβος για να εξυπηρετεί τα αιτήματα του (αυτός που περιέχει τα μεταδεδομένα της).

4.3. Γενική Επισκόπηση Συστήματος

Περιγράφουμε μια κατανεμημένη αρχιτεκτονική ενός συστήματος που αποθηκεύει ροές με μεταβλητό ρυθμό μεταδόσεων σε πολλαπλούς δίσκους. Οι πελάτες στέλνουν δεδομένα στο σύστημα και το σύστημα είναι υπεύθυνο να τα αποθηκεύσει αξιόπιστα.

Το σύστημα μας αποτελείται από τρία βασικά συστατικά: το διακομιστής ελέγχου (ΔΕ), το διακομιστής διανομής (ΔΔ) και τον διακομιστής αποθήκευσης (ΔΑ).

Κάθε πελάτης, που θέλει να αποθηκεύσει δεδομένα, αρχικά επικοινωνεί με τον ΔΕ δια μέσου μιας δικτυακής σύνδεσης, χρησιμοποιώντας τη διεπαφή απομακρυσμένης κλήσης διαδικασίας (RPC). Ο πελάτης λαμβάνει από τον ΔΕ τη διεύθυνση ενός από τους διαθέσιμους ΔΔ. Αφού, ο πελάτης πληροφορηθεί για το ΔΔ που θα χρησιμοποιήσει, αρχίζει να του στέλνει δεδομένα δια μέσου μιας σύνδεσης που βασίζεται και πάλι στη διεπαφή απομακρυσμένης κλήσης διαδικασίας. Στη συνέχεια, όταν ο ΔΔ αρχίζει να λαμβάνει δεδομένα, χρησιμοποιεί την πληροφορία που έχει για την κατάσταση του συστήματος προκειμένου να επιλέξει ποιους ΔΑ δεδομένων θα χρησιμοποιήσει. Έτσι, συνθέτει μια ομάδα, *n* ΔΑ δεδομένων, από τους οποίους *k* θα αποθηκεύουν τα δεδομένα της ροής και *n-k* την πλεονάζουσα πληροφορία. Έπειτα, δεσμεύει τα μπλοκ στους άνωθεν κόμβους και στη συνέχεια αρχίζει να τους στέλνει δεδομένα. Για κάθε αίτημα εγγραφής που τους στέλνει, ο ΔΔ περιμένει επιβεβαίωση ότι τα δεδομένα έχουν αποθηκευτεί. Ο ΔΔ παράλληλα κατασκευάζει και το μπλοκ πλεονασμού, το οποίο στέλνει στον κατάλληλο ΔΑ.



Σχήμα 4.2: Διάγραμμα ροής δεδομένων

4.3.1. Διακομιστής Ελέγχου (ΔΕ)

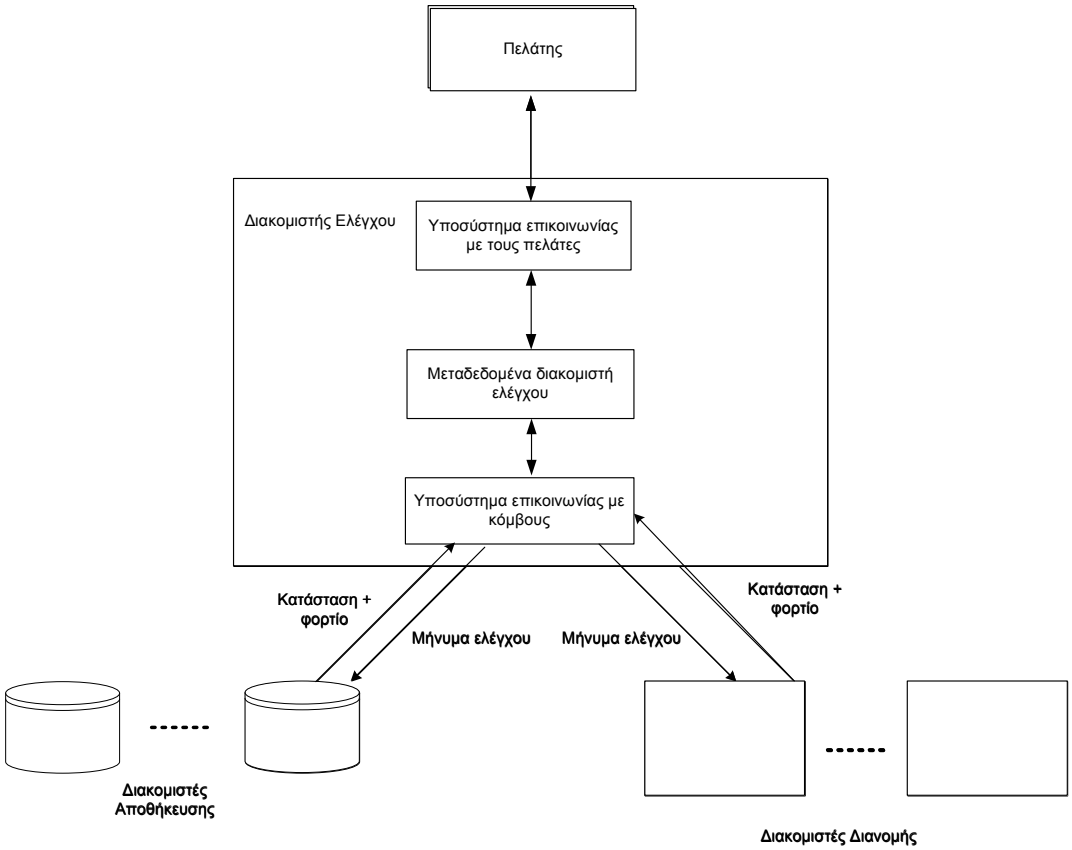
Ο ΔΕ διατηρεί μια γενική εικόνα για τη σύνθεση του συστήματος. Δηλαδή, διατηρεί πληροφορία για την κατάσταση και το φορτίο των ΔΔ και ΔΑ. Όλα τα δεδομένα των ροών δεδομένων που βρίσκονται σε ένα ΔΔ περνούν από μια ουρά πριν σταλούν στον ΔΑ. Έτσι, με βάση το πόσο φορτωμένες είναι οι ουρές αποστολής δεδομένων στους ΔΔ, ο ΔΕ δέχεται ένα νέο αίτημα. Ο ΔΕ ενημερώνεται περιοδικά για το μέγεθος αυτών των ουρών και ανάλογα με αυτό δέχεται ένα νέο αίτημα ή όχι. Ο ΔΕ αποτελείται από δυο υποσυστήματα τα οποία επικοινωνούν μεταξύ τους με κοινή μνήμη.

Το πρώτο υποσύστημα είναι υπεύθυνο να επικοινωνεί με τους ΔΔ και ΔΑ, να διαπιστώνει ποιοί από τους κόμβους του συστήματος είναι ενεργοί και να τους στέλνει διάφορες πληροφορίες. Έτσι, στέλνει περιοδικά μηνύματα ελέγχου και αναμένει απάντηση της κατάστασής τους. Αν κάποιος κόμβος δεν απαντήσει τότε σημαίνει ότι δεν είναι πλέον ενεργός. Στη περίπτωση που ο κόμβος, ο οποίος δεν απαντήσει στον ΔΕ,

είναι ΔΔ, τότε ο ΔΕ θα πρέπει να αναθέσει την ανάκτηση της πληροφορίας του σε κάποιον άλλο ΔΔ. Όπως, έχουμε προαναφέρει τα μεταδεδομένα του κάθε ΔΔ είναι αποθηκευμένα στην κύρια μνήμη του, στα αρχεία μεταδεδομένων και στα αρχεία καταγραφής συναλλαγών τα οποία βρίσκονται σε δυο ΔΑ. Έτσι, ο ΔΕ γνωρίζει τους διακομιστές αποθήκευσης που διατηρούν τα μεταδεδομένα και τα αρχεία καταγραφής συναλλαγών του ΔΔ. Αν πέσει κάποιος ΔΑ, τότε υπεύθυνοι να ανακτήσουν την πληροφορία είναι οι εκάστοτε ΔΔ. Δηλαδή, ο κάθε ΔΔ θα ανακτήσει τα δεδομένα τα οποία ήταν αποθηκευμένα στο ΔΑ για τις ροές που είναι υπεύθυνος.

Το δεύτερο υποσύστημα του ΔΕ είναι υπεύθυνο για την επικοινωνία με τους πελάτες. Ο ΔΕ αποτελεί το σημείο πρόσβασης των πελατών στο σύστημα μας. Ένας πελάτης που θέλει να αποθηκεύσει δεδομένα στο σύστημα μας πρέπει πρώτα να επικοινωνήσει με τον ΔΕ. Ο ΔΕ θα πρέπει να αναθέσει την αποθήκευση των δεδομένων του συγκεκριμένου αιτήματος σε κάποιον από τους ΔΔ καταγράφοντας ποιος από τους ΔΔ είναι υπεύθυνος για κάθε ροή. Η επιλογή των ΔΔ γίνεται εκ περιτροπής, δηλαδή οι ΔΔ προσπελάζονται ακολουθιακά αναλαμβάνοντας ο καθένας από μια ροή δεδομένων. Βασικά, θα ήταν καλύτερο αν η επιλογή του ΔΔ γινόταν με βάση το πόσο φορτωμένος είναι, δηλαδή με βάση το πόσο φορτωμένες είναι οι ουρές αποστολής του. Στην παρούσα εργασία υπολογίζεται το μήκος αυτών των ουρών αλλά δεν έχει χρησιμοποιηθεί για να επιλέξουμε τον ΔΔ που θα αναλάβει ένα αίτημα.

Τέλος, για λόγους ασφάλειας υπάρχει ακριβές αντίγραφο της πληροφορίας που περιέχει ο ΔΕ και σε ένα δεύτερο κόμβο. Έτσι, αν ο ΔΕ καταρρεύσει, το σύστημα μπορεί να συνεχίσει τη λειτουργία του χρησιμοποιώντας το κόμβο που περιέχει το αντίγραφο της πληροφορίας του.



Σχήμα 4.3: Υποσυστήματα διακομιστή ελέγχου

4.3.2. Διακομιστής διανομής (ΔΔ)

Ο ΔΔ αποτελείται από τρία βασικά υποσυστήματα τα οποία επικοινωνούν μεταξύ τους με κοινή μνήμη.

Το πρώτο υποσύστημα είναι υπεύθυνο για την επικοινωνία με τον ΔΕ.

Το δεύτερο υποσύστημα αναλαμβάνει να επικοινωνεί με τους πελάτες και να δέχεται και να αποθηκεύει τοπικά τα δεδομένα που του στέλνουν.

Το τρίτο υποσύστημα έχει ως βασική του λειτουργία την επικοινωνία με τους ΔΑ όπου στέλνει τα δεδομένα ροών που λαμβάνει από το πελάτη. Όπως έχουμε προαναφέρει, για κάθε ροή δεδομένων που αναλαμβάνει ένας ΔΔ να αποθηκεύσει, γίνεται αυτόματα

υπεύθυνος και για όλα τα επακόλουθα αιτήματα που θα αφορούν τη συγκεκριμένη ροή. Ως εκ τούτου μόνο ο ΔΔ, που έχει αποθηκεύσει μια ροή δεδομένων, έχει την απαραίτητη πληροφορία (μεταδεδομένα), ώστε να μπορεί να εξυπηρετεί τις αντίστοιχες αιτήσεις. Άρα, τα μεταδεδομένα του συστήματος μας είναι κατανεμημένα στους ΔΔ, με τον περιορισμό ότι ο κάθε ΔΔ περιέχει μόνο τα μεταδεδομένα των ροών δεδομένων τις οποίες έχει αυτός αποθηκεύσει. Τα μεταδεδομένα του συστήματος μας θα μπορούσε να ήταν όλα αποθηκευμένα στον ΔΕ. Όταν ένας ΔΔ θα ήθελε να αποθηκεύσει μια ροή δεδομένων, θα έπρεπε κάθε φορά να επικοινωνεί με τον ΔΕ για να παραλάμβανε όλη την πληροφορία που χρειάζονταν για την αποθήκευση (πχ ποιους διακομιστές να χρησιμοποιήσει). Αντίθετα, έχοντας ο κάθε ΔΔ όλη τη πληροφορία που χρειάζεται για να εκτελέσει μια λειτουργία τοπικά, το σύστημά μας αποκτά μεγαλύτερη κλιμακωσιμότητα σε σχέση με το αν είχαμε όλη την πληροφορία του συστήματος μας σε ένα κεντρικό κόμβο. Έτσι μπορούμε να προσθέτουμε ΔΔ πιο εύκολα, χωρίς να επηρεάζεται η λειτουργία κάποιου άλλου κόμβου. Στην περίπτωση που θα είχαμε τα μεταδεδομένα μας κεντρικοποιημένα (στον ΔΕ), οι περισσότεροι ΔΔ επιβαρύνουν την απόδοση του ΔΕ, καθώς θα έπρεπε ο τελευταίος να επικοινωνεί με περισσότερους κόμβους και να τους μοιράζει την πληροφορία που θα του ζητούσαν.

Ο ΔΔ διατηρεί αυτά τα μεταδεδομένα στη μνήμη του για λόγους απόδοσης. Τα μεταδεδομένα που διατηρεί ο κάθε ΔΔ τοπικά και γνωρίζει μόνο ο ίδιος, αφορούν τις ροές δεδομένων που έχει αποθηκεύσει:1) οι ΔΑ όπου βρίσκονται τα δεδομένα τους, 2) το μέγεθος των δεδομένων που περιέχει καθένας από αυτούς, 3) το τμήμα της ροής που έχει ο κάθε ΔΑ και 4) τη θέση των δεδομένων σε κάθε ΔΑ. Εφόσον διατηρεί τα μεταδεδομένα του στη μνήμη και είναι ο μόνος που διαθέτει τη συγκεκριμένη πληροφορία εμφανίζεται ο κίνδυνος σε περίπτωση κατάρρευσης να χαθούν και όλα τα μεταδεδομένα του. Για το λόγο αυτό, χρησιμοποιούνται αρχεία καταγραφής συναλλαγών (logs) και διαδικασία περιοδικής αποθήκευση (checkpoint). Τις παραπάνω διαδικασίες τις αναλαμβάνει το υποσύστημα του διακομιστή διανομής που επικοινωνεί με τους διακομιστές αποθήκευσης.

Ειδικότερα, για τη δημιουργία των αρχείων συναλλαγών χρησιμοποιήθηκε η βάση δεδομένων BerkeleyDB και ουσιαστικά πριν γίνει κάποια ενημέρωση των μεταδεδομένων του ΔΔ, προσθέτουμε μια εγγραφή στη βάση δεδομένων η οποία περιγράφει την ενημέρωση που θα γίνει και στην συνέχεια πραγματοποιείται η ενημέρωση.

Η BerkeleyDB είναι μια μη σχεσιακή βάση δεδομένων, η οποία προσφέρει υψηλής απόδοσης κλιμακώσιμες υπηρεσίες στις εφαρμογές που την χρησιμοποιούν. Η BerkeleyDB τρέχει στον ίδιο χώρο διευθύνσεων με την εφαρμογή που την χρησιμοποιεί και έτσι δεν απαιτεί κάποιου είδους διαδιεργασιακή επικοινωνία, για να εκτελεστεί κάποια λειτουργία στην βάση δεδομένων. Επίσης, υποστηρίζει και το μοντέλο πελάτη-διακομιστή με βάση το οποίο η βάση δεδομένων βρίσκεται σε διαφορετικό μηχάνημα σε σχέση με τον πελάτη που εκτελεί λειτουργίες σε αυτήν. Στην παρούσα εργασία χρησιμοποιήθηκε το μοντέλο πελάτη-διακομιστή.

Εναλλακτικά, για την υλοποίηση των αρχείων συναλλαγών εκτός της βάσης δεδομένων θα μπορούσαμε να χρησιμοποιήσουμε ένα απλό αρχείο καταγραφής. Όμως, προτιμήσαμε τη βάσης για λόγους, ασφάλειας, απόδοσης και για λόγους ευχρηστίας. Για οποιαδήποτε διαδικασία θέλουμε να εκτελέσουμε χρησιμοποιούμε της διαδικασίες διεπαφής της βάσης.
Η διαδικασία περιοδικής αποθήκευσης, εκτελείται περιοδικά και αποθηκεύει τα μεταδεδομένα που βρίσκονται στη μνήμη του ΔΔ στους δίσκους δυο ΔΑ (δύο αντίγραφα). Σε δύο αντίγραφα υπάρχει και το αρχείο καταγραφής συναλλαγών του, τα οποία και αυτά είναι αποθηκευμένα στους ΔΑ.
Μόλις, ο ΔΔ δημιουργήσει τα αρχεία καταγραφής των συναλλαγών του και τα αρχεία μεταδεδομένων του, το υποσύστημα του ΔΔ, που επικοινωνεί με τον ΔΕ, αναλαμβάνει να τον ενημερώσει για αυτές τις ενέργειες του ΔΔ. Δηλαδή, να τον ενημερώσει για το ποίοι ΔΑ χρησιμοποιήθηκαν.

Ο ΔΔ έχει τη δυνατότητα να εκτελεί τις παρακάτω λειτουργίες:

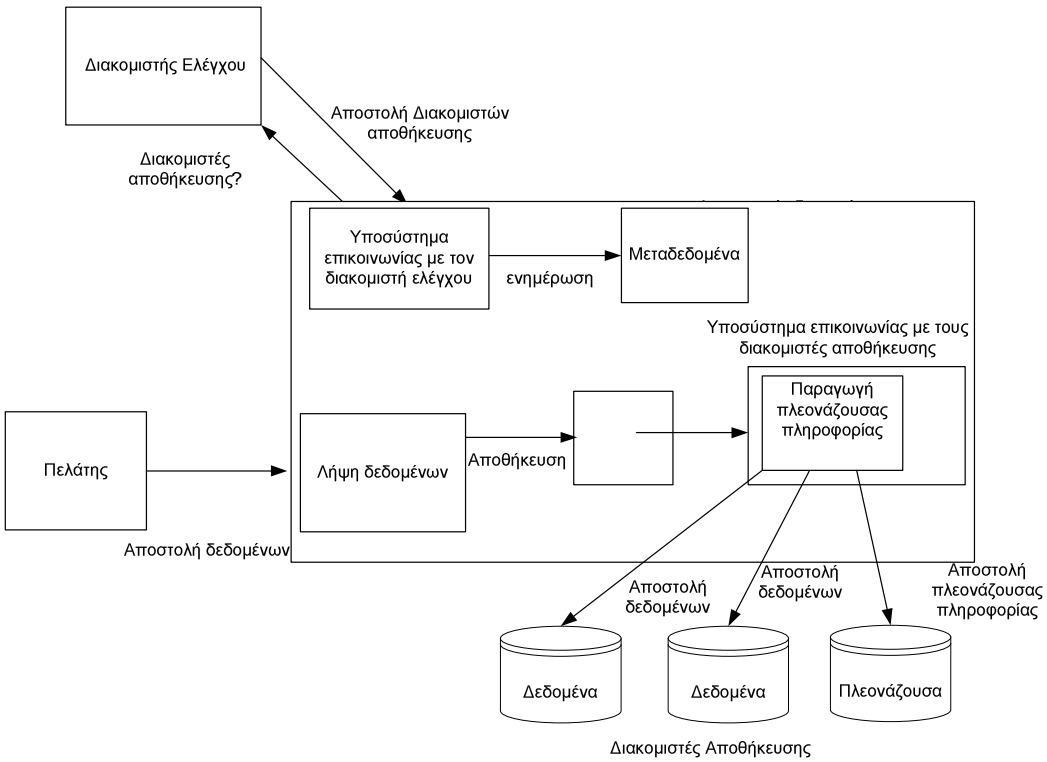
1. Αποθήκευση ροής δεδομένων
2. Ανάγνωση ροής δεδομένων
3. Ανάκτηση των μεταδεδομένων αποθήκευσης ροών ενός διακομιστή διανομής που έχει καταρρεύσει
4. Ανάκτηση της πληροφορία που σχετίζεται με τις ροές που είναι υπεύθυνος από ένα διακομιστής αποθήκευσης που έχει καταρρεύσει

4.3.2.1. Αποθήκευση ροής δεδομένων

Για κάθε ροή δεδομένων, που αναλαμβάνει ο ΔΔ, δημιουργεί μια ομάδα *N* ΔΑ, οι οποίοι θα χρησιμοποιηθούν για την αποθήκευση της. *N-K* από τους διακομιστές θα περιέχουν τα δεδομένα της ροής δεδομένων ενώ οι υπόλοιποι Κ θα περιέχουν τη πλεονάζουσα πληροφορία της ροής. Για τους ΔΑ που μπορεί να χρησιμοποιήσει τον ενημερώνει το υποσύστημα του που επικοινωνεί με τον ΔΕ.

Για κάθε αίτημα αποθήκευσης που δέχεται ο ΔΔ ενεργοποιεί το υποσύστημα που επικοινωνεί με τον πελάτη και το υποσύστημα που επικοινωνεί με τους ΔΑ. Όταν το υποσύστημα που επικοινωνεί με τον πελάτη, αρχίζει να δέχεται δεδομένα τα αποθηκεύει τοπικά στη μνήμη του ΔΑ. Έτσι, όταν το υποσύστημα που επικοινωνεί με τους ΔΑ διαπιστώσει ότι στη μνήμη του ΔΔ υπάρχουν δεδομένα προς αποθήκευση, αρχίζει να τα επεξεργάζεται για να τα στείλει στους ΔΑ. Αυτό που κάνει είναι να ομαδοποιεί τα δεδομένα αυτά σε ομάδες των *N-K* μπλοκ. Το μέγεθος μπλοκ (*B*) είναι μια σταθερά του συστήματος, η οποία περιγράφει την μονάδα μεταφοράς των δεδομένων προς τους ΔΑ. Στη συνέχεια για κάθε (*N-K*)***B* δεδομένα παράγει την πλεονάζουσα πληροφορία η οποία θα είναι *K*B*. Στη συνέχεια χωρίζει τα (*N-K*)***B* δεδομένα σε (*N-K*) τμήματα και τα στέλνει σε (*N-K*) ΔΑ δεδομένων και αντίστοιχα τη πλεονάζουσα πληροφορία σε *K* ΔΑ. Για την επικοινωνία του με τους ΔΑ χρησιμοποιείται η απομακρυσμένη κλήση διαδικασίας. Το υποσύστημα που επικοινωνεί με τους ΔΑ, όπως προαναφέραμε στέλνει τα δεδομένα των ροών δεδομένων σε μηνύματα μεγέθους *B*. Όμως, μπορεί μια ροή δεδομένων να μην έχει δεδομένα για να γεμίσει αυτό το μήνυμα. Έτσι, μπορεί να περάσει στο ίδιο μήνυμα δεδομένα μιας άλλης ροής που όμως προορίζονται στον ίδιο ΔΑ. Με

αυτό τον τρόπο επιτυγχάνουμε να συναθροίζουμε δεδομένα διαφορετικών ροών στο ίδιο μήνυμα από τον ΔΔ στους ΔΑ. Έτσι, καταφέρνουμε να εξοικονομούμε πόρους του συστήματος καθώς θα σταλούν λιγότερα μηνύματα στο δίκτυο σε σχέση με την κατάσταση στην οποία κάθε μήνυμα περιέχει δεδομένα μιας μόνο ροής .



Σχήμα 4.4: Αποθήκευση ροής δεδομένων

4.3.2.2. Ανάγνωση ροής δεδομένων

Όταν στον ΔΔ φτάσει ένα αίτημα ανάγνωσης, ενεργοποιεί τα υποσυστήματα που επικοινωνούν με τον πελάτη και με τους ΔΑ.

Το υποσύστημα που επικοινωνεί με τους ΔΑ αναλαμβάνει να επικοινωνήσει με τους δυο ΔΑ που περιέχουν τα δεδομένα της ροής που ζητήθηκε. Οι διακομιστές αρχίζουν να του

στέλνουν μηνύματα μεγέθους Β και το υποσύστημα αναλαμβάνει να ανασυνθέσει τοπικά τα δεδομένα και να τα αποθηκεύσει στη μνήμη.
Στη συνέχεια το υποσύστημα που επικοινωνεί με τον πελάτη παίρνει τα δεδομένα από τη μνήμη και τα στέλνει στον πελάτη.

Στην παρούσα εργασία για την ανάγνωση μιας ροής δεδομένων έχει υλοποιηθεί μόνο το πρώτο τμήμα. Δηλαδή, το υποσύστημα που επικοινωνεί με τους ΔΑ λαμβάνει τα δεδομένα από τους ΔΑ, ανασυνθέτει τοπικά τη πληροφορία και την αποθηκεύει σε ένα αρχείο.

4.3.2.3. Ανάκτηση της πληροφορίας ενός χρονοδρομολογητή

Όταν καταρρεύσει ένας ΔΔ, ο ΔΕ αναθέτει σε κάποιον άλλον να ανακτήσει την πληροφορία του. Το υποσύστημα του ΔΔ που αναλαμβάνει να ανακτήσει τα μεταδεδομένα του ΔΔ που έχει καταρρεύσει, επικοινωνεί με τον ΔΕ και του ζητάει τους ΔΑ που περιέχουν τα αρχεία εγγραφής συναλλαγών και τα αρχεία μεταδεδομένων του ΔΔ που έχει καταρρεύσει. Την πληροφορία αυτή την αποθηκεύει στην μνήμη. Στην συνέχεια το υποσύστημα που επικοινωνεί με τους ΔΑ χρησιμοποιεί αυτή τη πληροφορία για να επικοινωνήσει με έναν από τους δυο ΔΑ και του ζητάει το περιεχόμενο του αρχείου μεταδεδομένων του ΔΔ που έχει καταρρεύσει. Αφού ανακτήσει το αρχείο μεταδεδομένων, διατηρεί τα μεταδεδομένα στη μνήμη του, αρχίζει να ζητάει τις εγγραφές που υπάρχουν στο αρχείο εγγραφής συναλλαγών και να ενημερώνει ανάλογα τα μεταδεδομένα που έχει ανακτήσει. Μόλις, ανακτήσει όλες τις εγγραφές θα έχει όλα τα μεταδεδομένα που είχε ο ΔΔ που κατέρρευσε.

4.3.2.4. Ανάκτηση της πληροφορίας ενός διακομιστή αποθήκευσης

Όταν ένας ΔΔ θέλει να χρησιμοποιήσει ένα ΔΑ και αυτός έχει καταρρεύσει, θα πρέπει να ανακτήσει όλα τα δεδομένα των ροών δεδομένων που έχει αποθηκεύσει σε αυτόν. Έτσι,

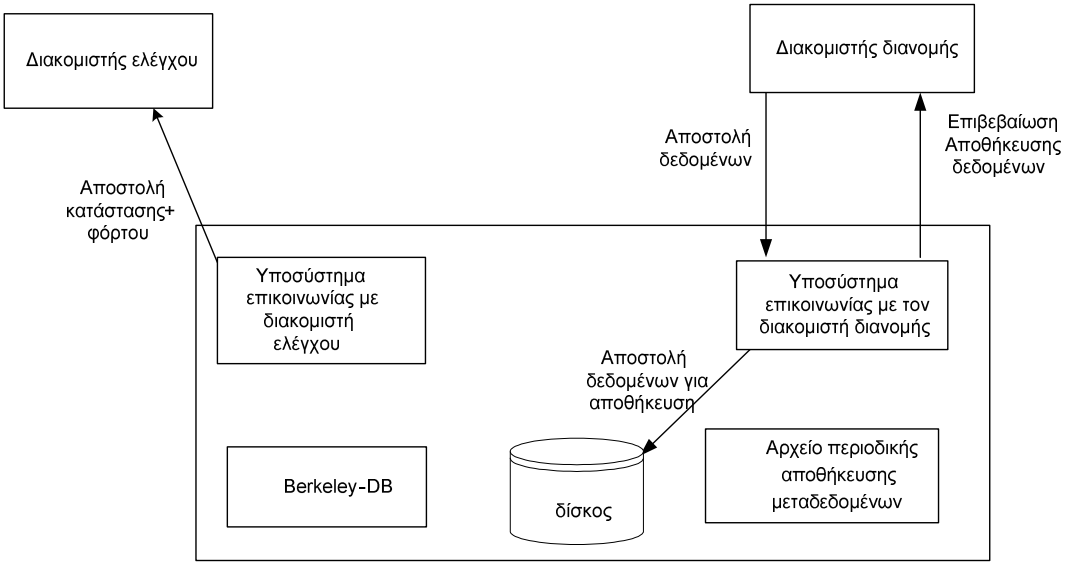
το υποσύστημα που επικοινωνεί με τους ΔΑ αναλαμβάνει να εκτελέσει την ακόλουθη διαδικασία.

Για κάθε ροή που έχει δεδομένα σε αυτόν τον κόμβο, επικοινωνεί με *K* ΔΑ που περιέχουν δεδομένα της ροής και τους ζήτα να του στείλουν τα δεδομένα της. Αυτοί θα του στείλουν τα δεδομένα σε μηνύματα. Στη συνέχεια, το υποσύστημα θα χρησιμοποιήσει τα δεδομένα που θα του στείλουν οι δυο ΔΑ για να ανακτήσει την πληροφορία που είχε ο κόμβος που κατέρρευσε. Ανάλογα την πληροφορία που είχε ο κόμβος που κατέρρευσε εκτελεί τα παρακάτω. Αν περιείχε δεδομένα, τότε χρησιμοποιείται μια μέθοδος αποκωδικοποίησης μεταξύ της πλεονάζουσας πληροφορίας και των δεδομένων, του κόμβου που είναι ενεργός, και παράγονται τα δεδομένα του κόμβου που κατέρρευσε. Τα δεδομένα παράγονται ανά *B* μέγεθος. Αν περιείχε πλεονάζουσα πληροφορία, τότε χρησιμοποιεί μια μέθοδος κωδικοποίησης μεταξύ των δεδομένων των κόμβων που είναι ενεργοί και παράγονται τα δεδομένα του κόμβου που κατέρρευσε. Τα δεδομένα παράγοντα και εδώ ανά Β μέγεθος .
Τέλος, μόλις ανακτήσει την πληροφορία επιλέγει ένα ΔΑ, ο οποίος δεν ανήκει στην ομάδα των ΔΑ της ροής. Σε αυτόν τον ΔΑ στέλνει τη πληροφορία που είχε πριν ο ΔΑ που κατέρρευσε.

4.3.3. Διακομιστής αποθήκευσης

Ο ΔΑ αποτελείται από δυο υποσυστήματα, τα οποία επικοινωνούν μεταξύ τους με κοινή μνήμη. Το πρώτο υποσύστημα αναλαμβάνει να επικοινωνεί περιοδικά με τον ΔΕ και να τον ειδοποιεί ότι ο ΔΑ στέλνοντας του παράλληλα και τον διαθέσιμο αποθηκευτικό χώρο του κόμβου.
Το δεύτερο υποσύστημα επικοινωνεί με τον ΔΔ, ο όποιος του στέλνει δεδομένα και το υποσύστημα αναλαμβάνει να τα αποθηκεύσει στον δίσκο και να ενημερώσει τον ΔΔ. Επιπλέον, εκτός των δεδομένων των ροών δεδομένων αποθηκεύει και τα μεταδεδομένα και τα αρχεία εγγραφής συναλλαγών των ΔΔ. Επίσης, για οτιδήποτε πληροφορία

αποθηκεύει διατηρεί και τα αντίστοιχα μεταδεδομένα, ώστε να μπορεί να βρίσκει την πληροφορία που χρειάζεται γρήγορα.



Σχήμα 4.5: Υποσυστήματα διακομιστή αποθήκευσης

ΚΕΦΑΛΑΙΟ 5. ΥΛΟΠΟΙΗΣΗ ΣΥΣΤΗΜΑΤΟΣ

5.1. Εισαγωγή

Το σύστημα αποτελείται από τρία μέρη τον διακομιστή ελέγχου (ΔΕ), διακομιστή διανομής (ΔΔ) και τον διακομιστή αποθήκευσης (ΔΑ).

Όπως έχουμε προαναφέρει για την διαδικτυακή επικοινωνία μεταξύ των κόμβων του συστήματος έχει χρησιμοποιηθεί η τεχνική της απομακρυσμένης κλήσης διαδικασία και πιο συγκεκριμένα όπως αυτή παράγεται από το εργαλείο rpcgen. Έτσι, κάθε κόμβος προσφέρει ένα σύνολο διαδικασιών στου υπολοίπους και με βάση αυτές τις διαδικασίες επιτυγχάνεται η δικτυακή επικοινωνία.

5.2. Διακομιστής Ελέγχου

Όπως έχουμε προαναφέρει, ο ΔΕ αποτελείται από τρία υποσύστημα: το υποσύστημα που επικοινωνεί με τον πελάτη, το υποσύστημα που επικοινωνεί με τους ΔΔ και το υποσύστημα που επικοινωνεί με τους ΔΑ. Καθένα από αυτά τα υποσυστήματα είναι νήματα σε επίπεδο χρήστη τα οποία ανήκουν στην ίδια διεργασία και επικοινωνούν

μεταξύ τους με κοινή μνήμη. Ο συγχρονισμός των νημάτων γίνεται με μεταβλητές συνθήκης.

Ο ΔΕ προσφέρει το παρακάτω σύνολο διαδικασιών στους υπόλοιπους κόμβους ώστε να επικοινωνήσουν μαζί του.

```
/*Εγγραφή διακομιστή αποθήκευσης*/
int register_storage_node(storage_node)=1;

/*Αποστολή διακομιστών αποθήκευσης*/
rec_nodes  send_nodes()=2;

/*κόμβοι που περιέχουν μεταδεδομένα*/
int send_metadata(metadata_group1)=3;

/*Αποστολή κόμβων που περιέχουν μεταδεδομένα */
struct metadata_group1 send_metadata_back(send_back_metadata_in)=4;
```

Σχήμα 5.1: Διαδικασίες του διακομιστή ελέγχου

Όλοι οι ΔΑ που θέλουν να χρησιμοποιηθούν από το σύστημα για να αποθηκεύσουν ροές δεδομένων, επικοινωνούν με το υποσύστημα του ΔΔ που είναι υπεύθυνο με την επικοινωνία με τους ΔΑ, καλώντας την διαδικασία *register_storage_node*, στέλνοντας τα στοιχεία τους. Έτσι, ο ΔΕ διατηρεί μια λίστα με όλους τους ΔΑ του συστήματος, αποθηκεύοντας για καθένα από αυτούς τη τρέχουσα κατάσταση του και το τρέχον φορτίο του.

Ο ΔΕ διατηρεί και μια λίστα με όλους τους ΔΔ του συστήματος. Για κάθε ΔΔ αποθηκεύει την τρέχουσα κατάσταση του, την κατάσταση των ουρών αιτήσεων και τους διακομιστές αποθήκευσης που διατηρούν τα μεταδεδομένα του και τα αρχεία καταγραφής συναλλαγών του.

Για να μπορέσει ένας ΔΔ να αποθηκεύσει ροές δεδομένων θα πρέπει να ενημερωθεί από το υποσύστημα του ΔΕ που είναι υπεύθυνο για την επικοινωνία με τους ΔΔ, για το ποιους ΔΑ μπορεί να χρησιμοποιήσει. Το υποσύστημα αυτό μόλις του ζητηθούν οι διαθέσιμοι ΔΑ θα τους αντιγράψει από την μνήμη που τους έχει αποθηκεύσει το πρώτο υποσύστημα σε ένα μπλοκ προσωρινής αποθήκευσης. Στη συνέχεια, θα στείλει αυτό το μπλοκ στον διακομιστή διανομής.

Ο ΔΔ ενημερώνεται για τους διαθέσιμους ΔΑ, καλώντας τη διαδικασία *send_nodes* του ΔΕ, με την οποία στέλνει την διεύθυνση και το τρέχον φορτίο του. Το αντίστοιχο υποσύστημα του ΔΕ αναλαμβάνει να αποθηκεύσει στη μνήμη του την πληροφορία που του έστειλε ο ΔΔ και να του επιστρέψει όλους τους διαθέσιμους ΔΑ.

Με την διαδικασία *send_metadata*, ο ΔΔ ενημερώνει τον ΔΕ για τους ΔΑ όπου βρίσκονται τα μεταδεδομένα του και τα αρχεία εγγραφής των συναλλαγών του. Τέλος, η κλήση *send_metadata_back*, χρησιμοποιείται από έναν ΔΔ ο οποίος θέλει να ανακτήσει την κατάσταση ενός ΔΔ που έχει καταρρεύσει. Έτσι, με αυτή τη κλήση μαθαίνει από το αντίστοιχο υποσύστημα του ΔΕ τους ΔΑ που περιέχουν τα αρχεία καταγραφής συναλλαγών και τα αρχεία μεταδεδομένων του ΔΔ που έχει καταρρεύσει.

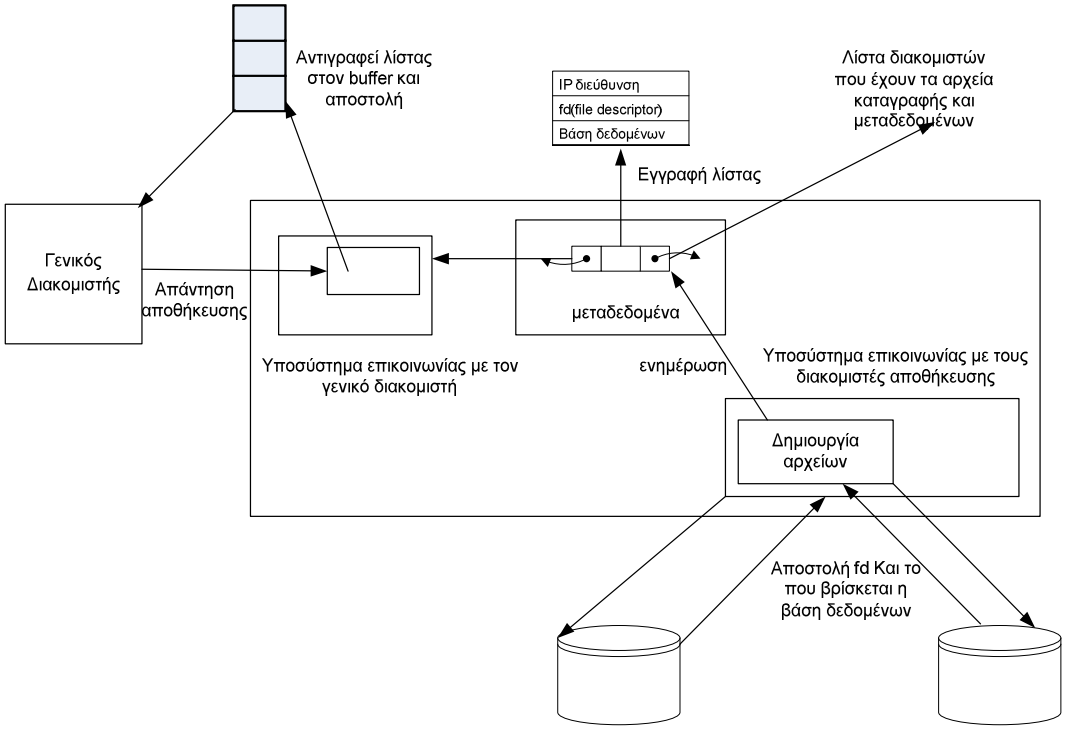
5.3. Διακομιστής Διανομής

Όπως, έχουμε προαναφέρει ο ΔΔ αποτελείται από τρία υποσυστήματα τα οποία είναι νήματα σε επίπεδο χρήστη και επικοινωνούν μεταξύ τους με κοινή μνήμη.

Κάθε ΔΔ κατά την εκκίνηση ενεργοποιεί το υποσύστημα που είναι υπεύθυνο για την επικοινωνία με τον ΔΕ. Το υποσύστημα αυτό αναλαμβάνει να ενημερώσει τον ΔΕ ότι ο ΔΔ είναι διαθέσιμος να δεχτεί κάποια ροή δεδομένων για αποθήκευση και ζητάει τη λίστα με τους ΔΑ, που μπορεί να χρησιμοποιήσει για να αποθηκεύσει τις ροές δεδομένων, καλώντας τη διαδικασία *send_nodes*. Στη συνέχεια διατηρεί την πληροφορία αυτή στην μνήμη, αφού πρώτα εξασφαλίσει ατομική πρόσβαση σε σχέση με τα υπόλοιπα νήματα. Ο ΔΔ διατηρεί μια λίστα με όλους τους ενεργούς ΔΑ, διατηρώντας για καθένα

από αυτούς το τρέχον φορτίο του και το χώρο που μπορεί να χρησιμοποιήσει για να αποθηκεύσει δεδομένα.

Μόλις, ενημερωθεί για τους διαθέσιμους ΔΑ, ενεργοποιεί το υποσύστημα που είναι υπεύθυνο για την επικοινωνία με τους ΔΑ. Στη συνέχεια δημιουργεί ένα υποσύστημα, το οποίο επιλέγει δυο από τους ΔΑ για να αποθηκεύσουν τα μεταδεδομένα του και τα αρχεία καταγραφής των συναλλαγών του, διατηρώντας την πληροφορία αυτή στη μνήμη. Μόλις τους επιλέξει, επικοινωνεί μαζί τους και δημιουργεί τα αρχεία μεταδεδομένων και τα αρχεία καταγραφής των συναλλαγών του. Στη συνέχεια χρησιμοποιεί την απομακρυσμένη κλήση *send_metadata* του ΔΕ για να τον ενημερώσει για τους ΔΑ που θα αποθηκεύουν τα αρχεία καταγραφής συναλλαγών και μεταδεδομένων του ΔΔ. Σε αυτή τη κλήση περνάει ως είσοδο τις διευθύνσεις των ΔΑ που χρησιμοποιήθηκαν, το δίσκο που βρίσκονται τα αρχεία καταγραφής συναλλαγών και τους περιγραφείς των αρχείων μεταδεδομένων στους δυο ΔΑ.



Σχήμα 5.2: Διαδικασία δημιουργίας των αρχείων καταγραφής και μεταδεδομένων του διακομιστή διανομής και κατάλληλη ενημέρωση του διακομιστή ελέγχου

Μετέπειτα, το υποσύστημα που επικοινωνεί με τους διακομιστές αποθήκευσης, ενεργοποιεί δυο νήματα σε επίπεδο χρήστη για να εκτελούν την ενημέρωση των αρχείων μεταδεδομένων του και των αρχείων καταγραφής των συναλλαγών του.

Το υποσύστημα περιοδικής αποθήκευσης αντιγράφων συνδέεται με τους δυο ΔΑ και αποθηκεύει τα μεταδεδομένα του. Περιοδικά τους στέλνει τα μεταδεδομένα των ροών για τις οποίες είναι υπεύθυνος ο ΔΔ.

Το δεύτερο υποσύστημα ενημερώνει τα αρχεία καταγραφής συναλλαγών του ΔΔ. Όπως, έχουμε προαναφέρει για την υλοποίηση των αρχείων καταγραφής των συναλλαγών χρησιμοποιήθηκε η βάση δεδομένων BerkeleyDB και ειδικότερα η διεπαφή πελάτη-διακομιστή που προσφέρει. Έτσι, το υποσύστημα αυτό αποτελεί τον πελάτη που στέλνει τις εγγραφές στον ΔΑ. Δέχεται εγγραφές σε μια ουρά, από άλλα υποσυστήματα που θέλουν να ενημερώσουν τα μεταδεδομένα που αφορούν τις ροές δεδομένων. Έτσι, προτού να ενημερώσουν τα μεταδεδομένα εισάγουν μια εγγραφή, που περιγράφει την ενημέρωση που θέλουν να κάνουν στην ουρά του υποσυστήματος και περιμένουν το νήμα να στείλει την εγγραφή στην βάση δεδομένων. Μόλις, στείλει την εγγραφή στις δυο βάσεις δεδομένων οι οποίες βρίσκονται σε δυο ΔΑ, ενημερώνει το αντίστοιχο υποσύστημα να κάνει την αλλαγή που επιθυμεί στα μεταδεδομένα των ροών δεδομένων.

Ο ΔΔ διατηρεί μια σύνδεση με κάθε ΔΑ που χρησιμοποιεί. Το κάθε υποσύστημα, που συνδέεται με τον ΔΑ, έχει μια ουρά στην οποία εισάγονται τα δεδομένα των ροών δεδομένων που προορίζονται προς τον συγκεκριμένο ΔΑ. Σε κάθε του εκτέλεση στέλνει όλη του την ούρα σε μια απομακρυσμένη κλήση και περιμένει μέχρι ο ΔΑ να του επιβεβαίωση ότι έχουν αποθηκευτεί όλα τα δεδομένα που του έστειλε. Αν κάποια από αυτά απέτυχε να τα αποθηκεύσει, τότε τα ξαναστέλνει στον ΔΑ. Επίσης, για κάθε ΔΑ δεσμεύει μνήμη προσωρινής αποθήκευσης μεγέθους *B*, την οποία ονομάζουμε *μπλοκ ροής*, στην οποία εισάγονται τα δεδομένα που προορίζονται προς τον ΔΑ. Μόλις, αυτή η μνήμη γεμίσει εισάγεται στην ουρά του υποσυστήματος που θα αναλάβει να τον στείλει στον ΔΑ. Σε αυτό το μπλοκ ροής γράφονται δεδομένα τα οποία μπορεί να ανήκουν σε

διαφορετικές ροές δεδομένων και έτσι επιτυγχάνουμε να συναθροίζουμε δεδομένα διαφορετικών ροών στο ίδιο μήνυμα.

Τέλος, υπάρχει ένα υποσύστημα που εκτελείται συνεχώς και ελέγχει όλα τα μπλοκ των ροών που περιέχουν δεδομένα για τους ΔΑ. Εάν, κάποιο περιέχει δεδομένα μεγέθους μικρότερο του Β και δεν έχει δεχτεί δεδομένα για κάποια περίοδο, τότε το υποσύστημα το εισάγει στην ουρά του αντίστοιχου υποσυστήματος για να την στείλει στον ΔΑ. Το μπλοκ ροής έχει μέγεθος μικρότερο του Β, καθώς σε αντίθετη περίπτωση θα είχε εισαχθεί αυτόματα στην αντίστοιχη ουρά. Μόλις εκτελέσει τις παραπάνω λειτουργίες, ο ΔΔ είναι έτοιμος να εκτελέσει όποια διαδικασία του ζητηθεί.

5.3.1. Αποθήκευση ροής δεδομένων

Κατά την αποθήκευση μιας ροής δεδομένων θεωρούμε ότι τα δεδομένα της ροής βρίσκονται ήδη στον ΔΔ και ότι αυτός θα πρέπει να τα αποθηκεύσει. Δεν έχει υλοποιηθεί δηλαδή το υποσύστημα του ΔΔ που επικοινωνεί με τον πελάτη.

Μόλις δεχτεί ένα αίτημα αποθήκευσης, ενεργοποιεί το υποσύστημα που επικοινωνεί με τους ΔΑ, το οποίο αναλαμβάνει την αποθήκευση και να διατηρεί τοπικά ένα πίνακα επιβεβαιώσεων, που θα ενημερώνει για τα μέρη της ροής τα οποία έχουν αποθηκευτεί στους δίσκους των ΔΑ.

Για κάθε νέα ροή που δέχεται προσθέτει μια εγγραφή στα μεταδεδομένα του για αυτή τη ροή. Τα μεταδεδομένα που αφορούν την ροή περιέχουν τα παρακάτω στοιχεία:

- Τις διευθύνσεις των διακομιστών που έχει αποθηκευτεί
- Το τμήμα της ροής έχει αποθηκευτεί σε κάθε διακομιστή
- Το περιγραφέα του αρχείου που έχει χρησιμοποιηθεί για την αποθήκευση της σε κάθε διακομιστή
- Το μέγεθος της πληροφορίας που έχει γραφτεί σε κάθε διακομιστή
- Ποίοι από αυτούς τους διακομιστές διατηρούν μπλοκ δεδομένων της ροής και ποιος το μπλοκ πλεονάζουσας πληροφορίας

- Έναν πίνακα επιβεβαιώσεων που περιέχει τα μέρη της ροής που έχουν σταλθεί στους διακομιστές αποθήκευσης και έχουν επιβεβαιωθεί ότι έχουν αποθηκευτεί
- Μια μεταβλητή αμοιβαίου αποκλεισμού ώστε η ενημέρωση των πεδίων της να γίνεται σειριακά.

Στη συνέχεια, επικοινωνεί με τους ΔΑ και δημιουργεί τα αρχεία που θα χρησιμοποιηθούν για την αποθήκευσης τη ροής, ενημερώνοντας κατάλληλα τα μεταδεδομένα της.

5.3.2. Λήψη δεδομένων

Το υποσύστημα ενεργοποιεί δυο υποσυστήματα τα οποία θα αναλάβουν να αποθηκεύσουν τη συγκεκριμένη ροή δεδομένων.

Το πρώτο υποσύστημα διαβάζει τα δεδομένα της ροής που βρίσκονται ήδη στον διακομιστή διανομής από ένα αρχείο. Όλα τα δεδομένα που διαβάζει τα αποθηκεύει στη μνήμη του σε μια ουρά. Κάθε στοιχείο της ουράς έχει μέγεθος ανάλογο του ρυθμού της εκάστοτε ροής δεδομένων. Δηλαδή, αν ο ρυθμός μια ροής είναι 1Mbyte/sec, τότε το μέγιστο μέγεθος του κάθε στοιχείου της ουράς είναι 1Mbyte. Ο ρυθμός μια ροής δεδομένων προσομοιώνεται με τον αριθμό των δεδομένων που θα διαβαστούν από το αντίστοιχο αρχείο ανά δευτερόλεπτο και θα εισαχθούν στην ουρά. Το υποσύστημα εισάγει ένα στοιχείο στην ουρά, αφού κλειδώσει την αντίστοιχη μεταβλητή αμοιβαίου αποκλεισμού και αν η ουρά ήταν άδεια ξυπνάει το υποσύστημα που πρέπει να διαβάσει αυτά τα δεδομένα.

Το δεύτερο υποσύστημα λαμβάνει τα δεδομένα από το πρώτο υποσύστημα, τα επεξεργάζεται και στη συνέχεια τα προωθεί στα αντίστοιχα υποσυστήματα που θα τα στείλουν στους ΔΑ. Μετά την επεξεργασία που θα κάνει παράγει δεδομένα τα οποία θα αποθηκευτούν σε τρεις ΔΑ. Τα δεδομένα προωθούνται για αποστολή στους ΔΑ σε ομάδες μεγέθους Β. Επίσης, διατηρεί τρία μπλοκ ενδιάμεσης αποθήκευσης μεγέθους Β,

ένα για κάθε ΔΑ, που θα χρησιμοποιηθεί για την αποθήκευση της ροής και σε κάθε του εκτέλεση εκτελεί τα διάφορα βήματα.

Αρχικά, διαβάζει από τη μνήμη τα δεδομένα της ροής και γεμίζει σειριακά τα μπλοκ μνήμη ενδιάμεσης αποθήκευσης, που αναφέρονται στους ΔΑ που θα περιέχουν τα δεδομένα της ροής. Το μέγιστο που μπορεί να αποθηκεύσει σε κάθε μπλοκ μνήμης για ένα διακομιστή αποθήκευσης είναι B bytes.

Στη συνέχεια για τα δεδομένα που έχει αποθηκεύσει στη μνήμη ενδιάμεσης αποθήκευσης θα πρέπει να παράγει την πλεονάζουσα πληροφορία. Για τον σκοπό αυτό κάνει χρήση της βιβλιοθήκης Jerasure και στη συγκεκριμένη περίπτωση της διαδικασίας κωδικοποίησης της βιβλιοθήκης. Η διαδικασία δέχεται ως είσοδο έναν πίνακα 3 θέσεων, όπου κάθε γραμμή του έχει 64 bytes. Σε αυτό το πίνακα περνάμε στις 2 πρώτες του θέσεις τα δεδομένα μας, τα οποία είναι στο σύνολο 128 ψηφιολέξεις (κάθε φορά) και ο αλγόριθμός μας επιστρέφει στην τρίτη θέση του την πλεονάζουσα πληροφορία μεγέθους 64 ψηφιολέξεις. Ο αλγόριθμος δέχεται, σε κάθε επανάληψη του υποσυστήματος ως είσοδο μεγέθους 2B δεδομένα και επιστρέφει μέγεθος B πλεονάζουσα πληροφορία. Αν έχουμε λιγότερα από μεγέθους 2B δεδομένα, τότε πριν εκτελεστεί η διαδικασία χρησιμοποιείται η τεχνική συμπλήρωσης. Δηλαδή, προσθέτουμε μηδενικά στα δεδομένα έως ότου να έχουμε μεγέθους 2B δεδομένα. Βέβαια, τα δεδομένα που στέλνονται στους ΔΑ δεν περιέχουν τα μηδενικά, στέλνεται μόνο η ωφέλιμη πληροφορία. Στην συνέχεια αντιγράφουμε την πλεονάζουσα πληροφορία στο μπλοκ ενδιάμεσης αποθήκευσης που έχει δεσμευτεί για τον ΔΑ που θα αποθηκεύσει την πλεονάζουσα πληροφορία.

Όπως έχουμε προαναφέρει ο ΔΔ διατηρεί ένα μπλοκ ροής για κάθε ΔΑ. Έτσι, κάθε μπλοκ ενδιάμεσης αποθήκευσης, πρέπει να αντιγράφει στα αντίστοιχα μπλοκ ροής (υπάρχει ένα για κάθε διακομιστή αποθήκευσης). Επειδή, σε αυτά τα μπλοκ μπορεί να υπάρχουν δεδομένα από διαφορετικές ροές δεδομένων, γράφουν περισσότερα από ένα υποσυστήματα, για να αντιγράψει ένα υποσύστημα τα μπλοκ ενδιάμεσης αποθήκευσης του στα αντίστοιχα μπλοκ ροής θα πρέπει να κλειδώσει και την αντίστοιχη μεταβλητή αμοιβαίου αποκλεισμού. Στη συνέχεια, μπορεί να αντιγράψει τα δεδομένα του και να προσθέσει μια εγγραφή στα μπλοκ ροής. Κάθε εγγραφή του μπλοκ ροής περιέχει κάποια πεδία. Περιέχει το περιγραφέα του αρχείου στο οποίο πρέπει να γραφτούν τα δεδομένα (στην πλευρά του διακομιστή αποθήκευσης), τη θέση του αρχείου από την οποία πρέπει

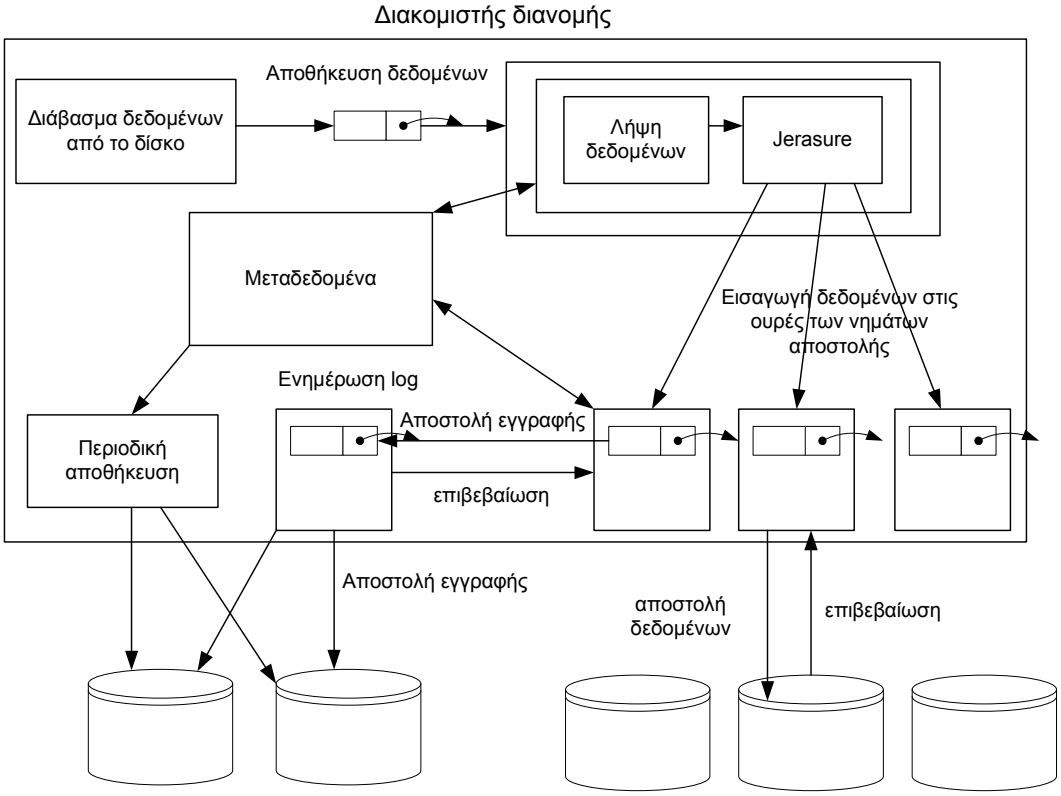
να ξεκινήσουμε για να αποθηκεύσουμε τα δεδομένα και τα δεδομένα της εγγραφής, που ουσιαστικά γράφονται σε ένα μπλοκ ροής που περιέχει και άλλα δεδομένα άλλων ροών δεδομένων. Επίσης διατηρεί το τρέχον μέγεθος της (μέγιστο μέγεθος που μπορεί να έχει είναι B), αν τα δεδομένα της συγκεκριμένης εγγραφής είναι δεδομένα ή πλεονάζουσα πληροφορία, τα όρια των δεδομένων της συγκεκριμένης εγγραφής στο μπλοκ ροής (send_data) και αν πρέπει να επανεγγραφούν δεδομένα όταν θα αποθηκεύσουμε τα δεδομένα στο αρχείο (εξηγείται παρακάτω). Επιπλέον συμπεριλαμβάνουμε τη χρονική στιγμή στην οποία έγινε η τελευταία αλλαγή σε αυτή την εργασία, που προορίζεται σε κάποιο υποσύστημα για να την στείλει σε κάποιον ΔΑ. Το πεδίο αυτό χρησιμοποιείται όταν το μπλοκ ροής έχει μέγεθος μικρότερο του B. Τότε, το υποσύστημα που ελέγχει τα μπλοκ ροής που έχουν δεδομένα και δεν έχουν προωθηθεί για αποστολή, τα προωθεί με βάση αυτόν τον χρόνο. Αν μετά την αντιγραφή των μπλοκ ενδιάμεσης αποθήκευσης κάποιο μπλοκ ροής γεμίσει τότε το υποσύστημα αναλαμβάνει να το εισάγει στην ουρά του αντίστοιχου υποσυστήματος που θα το στείλει στον ΔΑ.

Για κάθε μέρος της πληροφορίας (δεδομένων και όχι πλεονάζουσας) που αποθηκεύει στα μπλοκ ροής δημιουργεί μια εγγραφή στον πίνακα επιβεβαίωσης της ροής. Μόλις τα υποσυστήματα αποστολής λάβουν την επιβεβαίωση από το ΔΑ ότι η αντίστοιχη πληροφορία έχει γραφτεί στο δίσκο ενημερώνουν κατάλληλα το πίνακα της ροής.

Τέλος, το υποσύστημα απελευθερώνει τα μπλοκ ενδιάμεσης αποθήκευσης και δημιουργεί νέα μόνο αν αυτά έχουν γεμίσει. Σε αντίθετη περίπτωση τα δεδομένα διατηρούνται στη μνήμη, μέχρι να ληφθούν άλλα δεδομένα από τη συγκεκριμένη ροή. Αν φτάσουν, αντιγράφονται στα μπλοκ ενδιάμεσης αποθήκευσης έως ότου αυτά γεμίσουν. Στη συνέχεια εκτελείται ο αλγόριθμος πλεονασμού μόνο για το αντίστοιχο μέρος της πληροφορίας που πριν είχε μηδενικά. Τότε, θα πρέπει να επανεγγραφούν δεδομένα της πλεονάζουσας πληροφορίας που βρίσκονται στον αντίστοιχο ΔΑ, χρησιμοποιώντας το αντίστοιχο πεδίο που αναφέρθηκε προηγουμένως του μπλοκ ροής. Στα μπλοκ ροής δεν αντιγράφεται, μετέπειτα, όλη η πληροφορία που περιέχουν τα μπλοκ ενδιάμεσης αποθήκευσης αλλά μόνο η νέα πληροφορία σε σχέση με την προηγούμενη εκτέλεση του υποσυστήματος.

Τα υποσυστήματα που στέλνουν τα δεδομένα στους ΔΑ εκτελούν τα παρακάτω:

- Όσο δεν έχουν δεδομένα προς αποστολή μένουν ανενεργά
- Όταν φτάσουν κάποια δεδομένα, τα τοποθετούν σε ένα μήνυμα, ενημερώνοντας και τα αντίστοιχα πεδία της κάθε εγγραφής πριν τα στέλνουν στον αντίστοιχο ΔΑ. Περιμένουν έως ότου του απαντήσει ο ΔΑ, ο οποίος τους ενημερώνει αν αποθηκεύτηκαν όλα τα δεδομένα επιτυχώς.
- Για όλα τα δεδομένα που αποθηκεύτηκαν επιτυχώς ενημερώνουν αντίστοιχα τα μεταδεδομένα της αντίστοιχης ροής, δηλαδή το μέγεθος που έχει αποθηκεύσει ο αντίστοιχος ΔΑ για αυτή τη ροή και την ουρά επιβεβαίωσης για το μέρος της ροής που αποθηκεύτηκε
- Τα δεδομένα που δεν αποθηκεύτηκαν επιτυχώς τα ξαναστέλνουν στον ΔΑ



Σχήμα 5.3**:**Υποσυστήματα του διακομιστή διανομής και πως επικοινωνούν μεταξύ τους και με τους διακομιστές αποθήκευσης

5.3.3. Ανάγνωση ροής δεδομένων

Για την ανάγνωση μιας ροής δεδομένων, ο ΔΔ δημιουργεί δυο υποσυστήματα. Τα δύο αυτά υποσυστήματα δημιουργούν σύνδεση απομακρυσμένης κλήσης, με τους δυο ΔΑ που περιέχουν τα δεδομένα της ροής. Στην συνέχεια ζητούν από κάθε ΔΑ να τους στείλει τα δεδομένα της ροής. Οι ΔΑ στέλνουν τα δεδομένα της ροής σε μηνύματα μεγέθους Β, στέλνοντας επίσης και την αντίστοιχη μετατόπιση (offset) των δεδομένων. Μόλις, τα λάβει το αντίστοιχο υποσύστημα τα αποθηκεύει σε ένα αρχείο, στην αντίστοιχη μετατόπιση. Αφού ολοκληρωθεί αυτή η διαδικασία ο ΔΔ θα έχει στο δίσκο την ροή δεδομένων που του είχε στείλει ο αντίστοιχος πελάτης. Δεν έχει υλοποιηθεί το τμήμα που ο ΔΔ στέλνει τη ροή πίσω στον πελάτη.

5.3.4. Ανάκτηση πληροφορίας ενός διακομιστή διανομής

Όταν ένας ΔΔ θέλει να ανακτήσει την κατάσταση ενός ΔΔ που έχει καταρρεύσει, ενεργοποιεί το υποσύστημα που επικοινωνεί με τους ΔΑ. Το υποσύστημα αυτό δημιουργεί ένα υποσύστημα στο οποίο αναθέτει να ανακτήσει την πληροφορία. Το υποσύστημα επικοινωνεί με τον ΔΕ και χρησιμοποιεί την κλήση *send_metadata_back*, στην οποία περνά την διεύθυνση του ΔΔ που έχει καταρρεύσει και ο ΔΕ του επιστρέφει τους ΔΑ που περιέχουν τα μεταδεδομένα και τα αρχεία καταγραφής του συγκεκριμένου ΔΔ.

Στη συνέχεια επικοινωνεί με έναν από τους δυο ΔΑ και χρησιμοποιεί την κλήση *mymetadata_file* (εξηγείται παρακάτω) του ΔΑ στην οποία περνάει ως όρισμα τη διεύθυνση του ΔΔ που έχει καταρρεύσει. Ο ΔΑ του επιστρέφει σε ένα μπλοκ ενδιάμεσης αποθήκευσης το περιεχόμενο του αρχείου μεταδεδομένων. Αφού ανακτήσει την παραπάνω πληροφορία ανακτά από τον αντίστοιχο διακομιστή και τις εγγραφές του αρχείου καταγραφής συναλλαγών ενημερώνοντας κατάλληλα τα μεταδεδομένα που έχει ανακτήσει από το αρχείο μεταδεδομένων. Μόλις, ολοκληρώσει αυτή τη διαδικασία θα έχει τα μεταδεδομένα του ΔΔ που κατέρρευσε.

5.3.5. Ανάκτηση της πληροφορίας ενός διακομιστή αποθήκευσης

Όταν ο ΔΔ διαπιστώσει ότι κάποιος ΔΑ έχει καταρρεύσει, θα πρέπει να ανακτήσει τα δεδομένα, για κάθε ροή δεδομένων που έχει αποθηκεύσει στον συγκεκριμένο ΔΑ.

Έτσι, για να ανακτήσει τα δεδομένα μιας ροής, δημιουργεί μια σύνδεση απομακρυσμένης κλήσης, με καθένα από τους δυο ΔΑ που είναι ενεργοί και περιέχουν δεδομένα της ροής. Στην συνέχεια ζητά από καθένα να τους στείλει τα δεδομένα της ροής. Οι ΔΑ στέλνουν τα δεδομένα της ροής σε μηνύματα μεγέθους Β, στέλνοντας επίσης και την αντίστοιχη μετατόπιση των δεδομένων, τον ακριβή αριθμό των δεδομένων που έστειλαν και αν υπάρχουν άλλα δεδομένα. Στη συνέχεια, ο ΔΔ επιλέγει ένα ΔΑ για να αποθηκεύσει την πληροφορία που είχε ο κόμβος που κατέρρευσε. Δημιουργεί σε αυτόν ένα αρχείο το οποίο θα περιέχει τα ανακτώμενα δεδομένα της ροής. Μετέπειτα, διαβάζει τα δεδομένα που του έχουν στείλει οι δυο ενεργοί ΔΑ και παράγει την πληροφορία του κόμβου που έχει καταρρεύσει εκτελώντας την αντίστοιχη διαδικασία αποκωδικοποίησης και αποθηκεύει τα δεδομένα στον ΔΑ.

5.4. Διακομιστής Αποθήκευσης

Ο ΔΑ προσφέρει ένα σύνολο διαδικασιών απομακρυσμένης κλήσης στους ΔΔ ώστε να επικοινωνήσουν μαζί του.

```
/*Αποθήκευση δεδομένων*/
mytype_out write_data(mytype)=1;

/*Δημιουργία αρχείων για δεδομένα*/
files_out create_files(files)=2;

 /*Ανάγνωση δεδομένων*/
read_data_out myread(read_data)=3;

/*Δημιουργία αρχείων   για μεταδεδομένα*/
metadata_files_out create_metadata_files(metadata_files)=4;

/*διαδικασία περιοδικής αποθήκευσης*/
int mycheckpoint(checkpoint_arguments)=5;

/*επιστρέφεται το περιεχόμενο ενός αρχείου μεταδεδομένων */
recv_metadata_file mymetadata_file(send_metadata_file)=6;
```

Σχήμα 5.4: Διαδικασίες του διακομιστή αποθήκευσης

Στον διακομιστή αποθήκευσης εκτελούνται δυο δαίμονες. Ο πρώτος, ακούει στο δίκτυο και περιμένει από τους ΔΔ να του στείλουν δεδομένα για αποθήκευση. Μόλις, λάβει τα δεδομένα ο διακομιστής τα επεξεργάζεται και αφού γραφτούν στο δίσκο του, απαντάει στον αντίστοιχο ΔΔ ότι τα δεδομένα του αποθηκεύτηκαν. Αν κάποια δεδομένα δεν κατάφερε να τα αποθηκεύσει ενημερώνει αντίστοιχα τον ΔΔ.

Ο δεύτερος περιμένει εγγραφές δεδομένων τις οποίες τις αποθηκεύει σε μια βάση δεδομένων. Όπως έχουμε προαναφέρει για τη δημιουργία των αρχείων καταγραφής συναλλαγών, χρησιμοποιήθηκε η διεπαφή πελάτη-διακομιστή που προσφέρει η βάση δεδομένων Berkeley-DB. Έτσι αυτό το υποσύστημα αποτελεί τον διακομιστή αυτής της εφαρμογής και αναλαμβάνει να ενημερώνει το αρχείο συναλλαγών του αντίστοιχου ΔΔ.

Επίσης, κατά την εκκίνηση του, ο ΔΑ δημιουργεί πολλαπλά νήματα (π.χ. 10) τα οποία θα αναλάβουν να αποθηκεύσουν τα δεδομένα στους δίσκους.
Ο ΔΔ μπορεί να χρησιμοποιήσει τις παραπάνω διαδικασίες για να αλληλεπιδράσει με τον ΔΑ.

Με τη κλήση write_data μπορεί να αποθηκεύσει δεδομένα στον ΔΑ. Όταν, φτάσουν στον ΔΑ δεδομένα προς αποθήκευση, το κεντρικό υποσύστημα αναλαμβάνει να επεξεργαστεί το μήνυμα. Όπως, έχουμε προαναφέρει κάθε υποσύστημα του ΔΔ στέλνει όλα τα δεδομένα που βρίσκονται στην ουρά του κάθε φορά. Έτσι, το κεντρικό υποσύστημα χωρίζει το μήνυμα με βάση τα δεδομένα που αναφέρονται σε διαφορετική ροή (διαφορετικό περιγραφέα αρχείου), και αναθέτει στα υποσυστήματα που είναι υπεύθυνα να αποθηκεύσουν τα δεδομένα στο δίσκο τα δεδομένα του μηνύματος. Τα υποσυστήματα αυτά έχουν μια ουρά το καθένα, στην οποία εισάγονται τα δεδομένα που πρέπει να αποθηκεύσουν. Κάθε εγγραφή της ουρά περιέχει, τον περιγραφέα του αρχείου που πρέπει να εγγραφούν τα δεδομένα, αν θα πρέπει να επανεγγραφούν δεδομένα ή όχι (μονό στην περίπτωση της πλεονάζουσας πληροφορίας), τα δεδομένα, το μέγεθος των δεδομένων, τη θέση που πρέπει να αποθηκευτούν τα δεδομένα μέσα στο αρχείο, και τέλος ένα barrier για να ειδοποιήσει το κεντρικό υποσύστημα ότι τα δεδομένα αποθηκεύτηκαν. Κάθε υποσύστημα αναλαμβάνει να αποθηκεύσει τα δεδομένα μια συγκεκριμένης ροής δεδομένων, δηλαδή αποθηκεύει δεδομένα σε ένα αρχείο. Το κεντρικό υποσύστημα μόλις εισάγει τα δεδομένα στις ουρές των υποσυστημάτων, περιμένει σε ένα barrier μέχρι να αποθηκευτούν όλα τα δεδομένα του πακέτου που έχει λάβει. Τέλος, ειδοποιεί τον ΔΔ για την αποθήκευση των δεδομένων που του έστειλε.

Με την κλήση create_files δημιουργεί αρχεία στα οποία στη συνέχεια θα αποθηκεύσει τα δεδομένα κάποιας ροής δεδομένων και λαμβάνει τον περιγραφέα του αρχείου που δημιούργησε.

Με την κλήση create_metadata_files δημιουργεί αρχεία στα οποία στη συνέχεια θα αποθηκεύσει τα μεταδεδομένα του και λαμβάνει τον περιγραφέα του αρχείου που δημιούργησε

Με την κλήση mycheckpoint στέλνει τα μεταδεδομένα από τη μνήμη του στο αρχείο μεταδεδομένων που έχει δημιουργήσει προηγουμένως με την κλήση create_metadata_files.

Τέλος, με την κλήση mymetadata_file επιστρέφεται το περιεχόμενο ενός αρχείου μεταδεδομένων κάποιου ΔΔ. Η κλήση αυτή χρησιμοποιείται όταν κάποιος ΔΔ έχει καταρρεύσει και κάποιος άλλος προσπαθεί να ανακτήσει τα δεδομένα του. Τα δεδομένα επιστρέφονται σε ένα μπλοκ ενδιάμεσης αποθήκευσης. Καθώς, ένας ΔΔ εκτελεί

διάφορες διαδικασίες σε ένα ΔΑ, ο ΔΑ διατηρεί μια λίστα με όλους τους ΔΔ για τους οποίους διατηρεί αρχεία μεταδεδομένων και αρχεία συναλλαγών. Επίσης διατηρεί μια λίστα με τις ροές που αποθηκεύει.

ΚΕΦΑΛΑΙΟ 6. ΠΕΙΡΑΜΑΤΙΚΑ ΑΠΟΤΕΛΕΣΜΑΤΑ

6.1. Περιγραφή Πειραμάτων

Για την εκτέλεση των πειραμάτων μας χρησιμοποιήθηκαν δυο ειδών κόμβοι. Η πρώτη ομάδα αυτών των κόμβων έχει τη διανομή Debian 4 του Linux με έκδοση πυρήνα 2.6.18. Κάθε κόμβος περιλαμβάνει έναν τετραπύρηνο επεξεργαστή Intel Xeon E5345 2.33GHz, 2Gb μνήμη RAM και δυο 250Gb 7200 RPM SATA δίσκους (ονομαστικά χαρακτηριστικά: μέσος χρόνος αναζήτησης 8.7ms, ρυθμός μεταφοράς 70Mb/s). Σε αυτή την ομάδα κόμβων εκτελούνται ο ΔΕ του συστήματος μας και ο ΔΔ.

Η δεύτερη ομάδα κόμβων έχει τη διανομή Debian 4 του Linux με έκδοση πυρήνα 2.6.18. Κάθε κόμβος περιλαμβάνει έναν τετραπύρηνο επεξεργαστή Intel Xeon E5345 2.66GHz, 3Gb μνήμη RAM και δυο 300Gb 15000 RPM SATA δίσκους (ονομαστικά χαρακτηριστικά: μέσος χρόνος αναζήτησης 3.6ms, ρυθμός μεταφοράς 150Mb/s). Σε αυτή την ομάδα κόμβων εκτελείται οι ΔΑ

.

Τέλος για την μεταφορά των δεδομένων μεταξύ των κόμβων χρησιμοποιήσαμε ένα δίκτυο το οποίο είχε ταχύτητα μετάδοσης των δεδομένων 1Gbit/sec.

Στα πειράματα μας χρησιμοποιήθηκε ένας ΔΔ και τέσσερις ΔΑ. Ο ΔΔ είχε αποθηκευμένα στο δίσκο του ροές δεδομένων τις οποίες στέλνει στους ΔΔ. Για την αποθήκευση μιας ροής δεδομένων χρησιμοποιήθηκαν τρεις ΔΑ, στους δυο αποθηκεύονταν τα δεδομένα της ροής και στον τρίτο η πλεονάζουσα πληροφορία. Στην παρούσα εργασία υλοποιήθηκαν δυο ειδών πειράματα.

Στην πρώτη ομάδα πειραμάτων ο ΔΔ αναλαμβάνει να αποθηκεύσει ένα αριθμό ροών δεδομένων, οι οποίες έχουν μέγεθος 10 Mbyte και ρυθμό μετάδοσης 1Mbyte/sec. Σε αυτό το είδος πειραμάτων μετρήθηκε η ρυθμαπόδοση διαφόρων τμημάτων του συστήματος σε συνάρτηση με την σταθερά μπλοκ του συστήματος και σε συνάρτηση με τον αριθμό των ταυτόχρονων ροών δεδομένων που έπρεπε να αποθηκεύσει ο ΔΔ. Επίσης, μετρήθηκε και το ποσοστό στο οποίο χρησιμοποιούνται οι πόροι του κόμβου στον οποίο τρέχει ο ΔΔ (επεξεργαστής) .

Στη δεύτερη ομάδα πειραμάτων μετρήσαμε το χρόνος που χρειάζεται για να ανακτηθούν τα μεταδεδομένα ενός ΔΔ που έχει καταρρεύσει. Σε αυτό το πείραμα χρησιμοποιήθηκαν ροές δεδομένων με ρυθμό μετάδοσης 1Mbyte/sec και υπολογίστηκε ο χρόνος που χρειάζεται για να ανακτηθούν τα μεταδεδομένα σε συνάρτηση με την περίοδο που χρησιμοποιούσε ο ΔΔ για να εκτελέσει την διαδικασία περιοδικής αποθήκευσης μεταδεδομένων.

Πίνακας 6.1 Παράμετροι του συστήματος

Μέγεθος μπλοκ μνήμης	B
Αριθμός ταυτόχρονων ροών δεδομένων	N_R
Ρυθμός μετάδοσης ροών δεδομένων	R_R
Μέγεθος ροής δεδομένων	S_R
Περίοδος περιοδικής αποθήκευσης μεταδεδομένων	T_c
Ρυθμαπόδοση δικτύου	T_N
Ρυθμαπόδοση υπολογισμού πλεονάζουσας πληροφορίας	T_R
Ρυθμαπόδοση αλγόριθμού συνάθροισης δεδομένων	T_A
Ρυθμαπόδοση δίσκου	T_D

Για την λήψη των αποτελεσμάτων επαναλάβαμε κάθε πείραμα 5 φορές και στην συνέχεια υπολογίσαμε το μέσο όρο των αποτελεσμάτων των 5 επαναλήψεων.

6.2. Ρυθμαπόδοση

Σε αυτή την ομάδα πειραμάτων υπολογίσαμε τη ρυθμαπόδοση διαφόρων τμημάτων του συστήματος μας.

Η ρυθμαπόδοση ορίζεται ως ο συνολικός αριθμός bytes που μπορεί να επεξεργαστεί κάθε τμήμα προς τον συνολικό χρόνο που χρειάστηκε για αυτήν την επεξεργασία.

Υπολογίστηκε η ρυθμαπόδοση των παρακάτω τμημάτων:

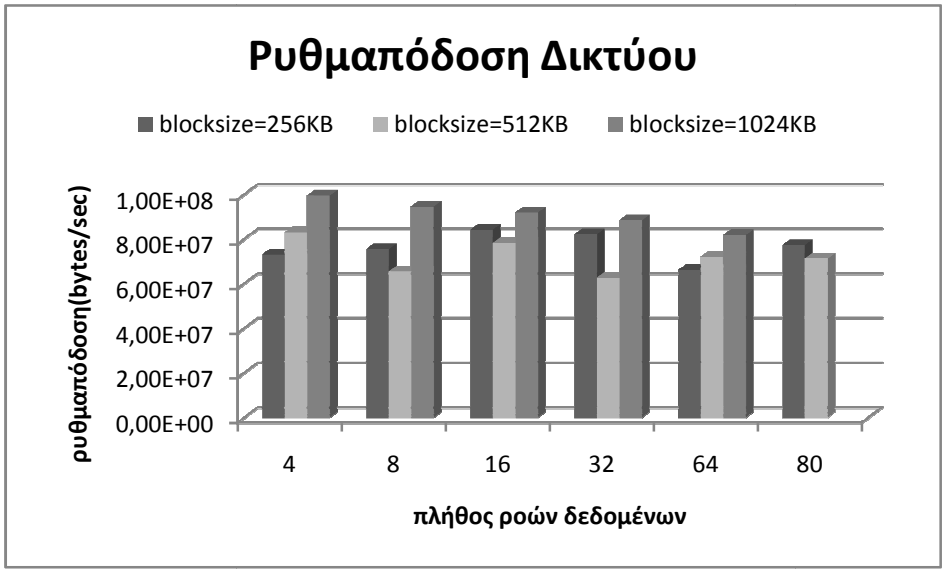
- Ρυθμαπόδοση δικτύου (T_N): Η ρυθμαπόδοση του δικτύου ορίζεται ως ο συνολικός αριθμός των bytes που έστειλε ο διακομιστής διανομής στους

διακομιστές αποθήκευσης προς το χρόνο που χρειάστηκε για να φτάσει η πληροφορία στους διακομιστές αποθήκευσης συν το χρόνο για να επιστρέψουν αυτοί την απάντηση. Δεν συμπεριλαμβάνεται ο χρόνος που χρειάστηκε για να επεξεργαστεί την πληροφορία ο εκάστοτε διακομιστής αποθήκευσης

- Τη ρυθμαπόδοση του αλγορίθμου παραγωγής της πλεονάζουσας πληροφορίας(T_R)
- Τη ρυθμαπόδοση του αλγορίθμου, ο οποίος χρησιμοποιείται για να συναθροίσει όλα τα μηνύματα που βρίσκονται σε μια ούρα αποστολής στον ΔΔ σε ένα μήνυμα πριν τα στείλει στον αντίστοιχο ΔΑ (T_A)
- Ρυθμαπόδοση αποθήκευση δεδομένων στο δίσκο (T_D): Η ρυθμαπόδοση στο δίσκο μετριέται στου διακομιστές αποθήκευσης και ορίζεται ως ο συνολικός αριθμός bytes που έχει αποθηκευτεί στους δίσκους προς το συνολικό χρόνο που χρειάστηκε για αυτήν την αποθήκευση

6.2.1. Ρυθμαπόδοση Δικτύου

Στο πείραμα αυτό μετράμε τη ρυθμαπόδοση του δικτύου, κατά την αποστολή των δεδομένων από τον διακομιστή διανομής στους διακομιστές αποθηκεύσεις.



Σχήμα 6.1: Υπολογίζεται η ρυθμαπόδοση του δικτύου σε συνάρτηση με το πλήθος των ταυτόχρονων ροών δεδομένων που υπάρχουν στο σύστημα. Επίσης, εμφανίζονται για κάθε πλήθος ροών δεδομένων μια ράβδο για κάθε διαφορετική τιμή μπλοκ που επιλέχθηκε.

Στο σχήμα 6.1 φαίνεται η επίδραση της ρυθμαπόδοσης του δικτύου σε συνάρτηση με το πλήθος των ροών δεδομένων που πρέπει να στείλει το σύστημα μας για αποθήκευση αλλά και σε σχέση με τη τιμή του μπλοκ που επιλέγεται.

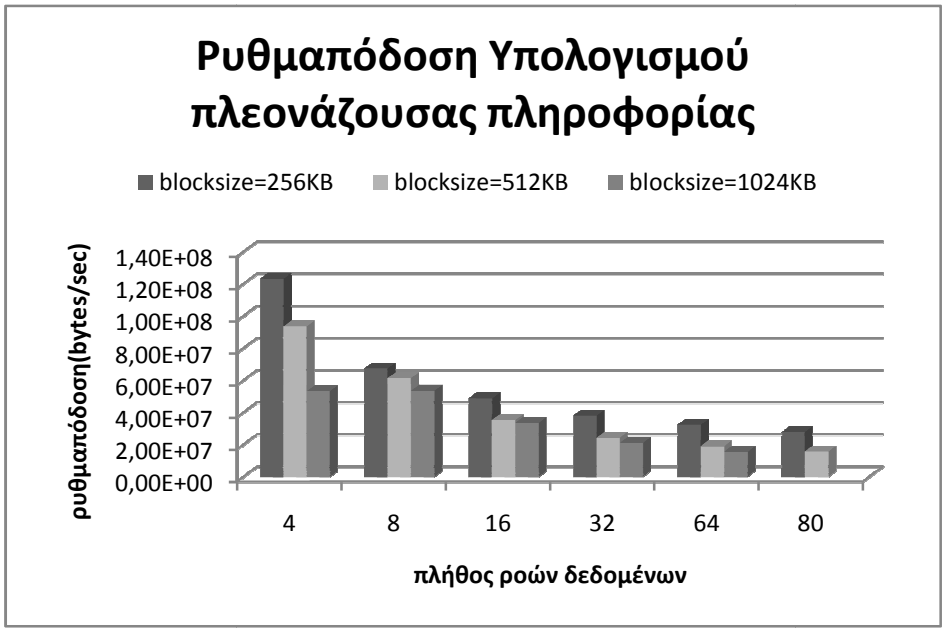
Γενικά παρατηρούμε ότι η ρυθμαπόδοση του δικτύου δεν εξαρτάται από το πλήθος των ροών δεδομένων αλλά από το μέγεθος του μπλοκ.

Με βάση την ταχύτητα του δικτύου για να στείλουμε 256KB χρειαζόμαστε 2 msec, για 512KB 4msec και για 1024KB 8msec. Όταν έχουμε 4 ταυτόχρονες ροές στο σύστημα

μας, χρειάζονται 125msec για να φθάσουν 256 KB στις ουρές των υποσυστημάτων του διακομιστή διανομής, 250 msec για 512KB και 500 msec για 1024KB.

Η επικοινωνία μεταξύ των κόμβων χρησιμοποιεί συνδέσεις TCP. Το TCP χρησιμοποιεί έλεγχο ροής, ο οποίος επιτρέπει στο ένα άκρο επικοινωνία να στέλνει πολλαπλά ταυτόχρονα πακέτα περιμένοντας μια επιβεβαίωση για κάθε πακέτο που έχει στείλει. Ο αριθμός των ταυτόχρονων πακέτων που μπορεί να στείλει εξαρτάται από το μέγεθος του *παραθύρου* που χρησιμοποιεί. Δηλαδή, το μέγεθος του παραθύρου ορίζει το μέγιστο αριθμό ταυτόχρονων πακέτων που μπορούν να σταλούν. Στην αρχή της σύνδεσης το μέγεθος αυτό είναι μικρό (1) και αυξάνεται αριθμητικά (πολλαπλασιάζουμε κάθε φορά το μέγεθος επί 2), με το χρόνο ανάλογα με τον ρυθμό που φτάνουν οι επιβεβαιώσεις από το άκρο που δέχεται την πληροφορία. Έτσι, το TCP αξιοποιεί σταδιακά το δίκτυο λόγο του ελέγχου ροής που χρησιμοποιεί.

6.2.2. Ρυθμαπόδοση αλγορίθμου πλεονάζουσας πληροφορίας



Σχήμα 6.2: Υπολογίζεται η ρυθμαπόδοση του αλγορίθμου παραγωγής της πλεονάζουσας πληροφορίας σε συνάρτηση με το πλήθος των ταυτόχρονων ροών δεδομένων που υπάρχουν στο σύστημα. Επίσης, εμφανίζονται για κάθε πλήθος ροών δεδομένων μια ράβδος για κάθε διαφορετική τιμή μπλοκ που επιλέχθηκε.

Στο σχήμα 6.2 φαίνεται πως επηρεάζεται η ρυθμαπόδοση του αλγορίθμου παραγωγής της πλεονάζουσας πληροφορίας σε συνάρτηση με το πλήθος των ροών δεδομένων που πρέπει να στείλει το σύστημα μας για αποθήκευση αλλά και σε σχέση με τη τιμή του μπλοκ που επιλέγεται.

Όπως έχουμε αναφέρει για κάθε νέα ροή δεδομένων που αναλαμβάνει να αποθηκεύσει ο διακομιστής διανομής δημιουργεί ένα νήμα για να υπολογίσει την πλεονάζουσα πληροφορία της ροής. Επίσης, όλα τα νήματα που εκτελούνται στον ΔΔ ανήκουν στη ίδια διεργασία. Έτσι, όλα τα νήματα που θα υπολογίζουν την πλεονάζουσα πληροφορία των ροών δεδομένων που θα φτάνουν στον ΔΔ θα τα αναλαμβάνει ο ίδιος επεξεργαστής.

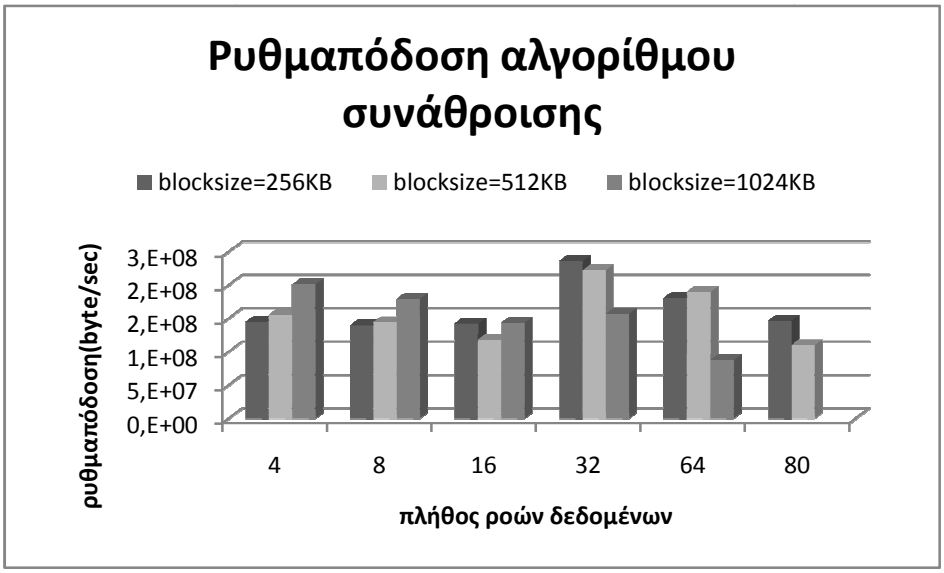
Γενικά παρατηρούμε ότι όσο περισσότερες ροές δεδομένων υπάρχουν στο σύστημα τόσο μειώνεται η ρυθμαπόδοση του αλγορίθμου παραγωγής της πλεονάζουσας πληροφορίας. Αυτό οφείλεται στο γεγονός ότι στην ίδια διεργασία θα εκτελούνται περισσότερα νήματα τα οποία θα διαμοιράζονται τον ίδιο επεξεργαστή με αποτέλεσμα ο υπολογισμός αυτός να παίρνει περισσότερο χρόνο και άρα να μειώνεται η ρυθμαπόδοση του αλγορίθμου.

Επίσης, παρατηρούμε ότι όσο μεγαλύτερο μέγεθος μπλοκ έχουμε τόσο μειώνεται η ρυθμαπόδοση του συστήματος. Αυτό οφείλεται στο γεγονός ότι όσο μεγαλύτερο μπλοκ έχουμε τόσο το εκάστοτε νήμα θα χρησιμοποιεί για περισσότερο χρόνο τον επεξεργαστή και άρα ο ρυθμός παραγωγής της πλεονάζουσας πληροφορίας θα μειώνεται.

6.2.3. Ρυθμαπόδοση αλγορίθμου συνάθροισης μηνυμάτων

Στον διακομιστή αποθήκευσης υπάρχουν κάποια υποσυστήματα τα οποία αναλαμβάνουν να στείλουν τα δεδομένα των ροών δεδομένων στους διακομιστές αποθήκευσης. Κάθε υποσύστημα έχει μια ουρά δεδομένων στην οποία δέχεται δεδομένα και αυτό αναλαμβάνει να τα μεταδώσει. Όπως έχουμε προαναφέρει κάθε υποσύστημα αποστολής σε κάθε του εκτέλεση παίρνει όλα τα δεδομένα που βρίσκονται στην ουρά του τα συναθροίζει σε ένα μήνυμα το οποίο θα στείλει στον διακομιστή αποθήκευσης. Άρα, αφού στέλνει όλη την ουρά του το τελικό μήνυμα αποστολής που δημιουργεί είναι μεταβλητού μεγέθους. Επίσης, κάθε στοιχείο αυτής της ουράς έχει μέγιστο μέγεθος μπλοκ.

Σε αυτό το πείραμα μετράμε την απόδοση αυτού του αλγορίθμου συνάθροισης των μηνυμάτων σε ένα μήνυμα.



Σχήμα 6.3: Υπολογίζεται η ρυθμαπόδοση του αλγορίθμου συνάθροισης σε συνάρτηση με το πλήθος των ταυτόχρονων ροών δεδομένων που υπάρχουν στο σύστημα. Επίσης, εμφανίζονται για κάθε πλήθος ροών δεδομένων μια ράβδο για κάθε διαφορετική τιμή μπλοκ που επιλέχθηκε.

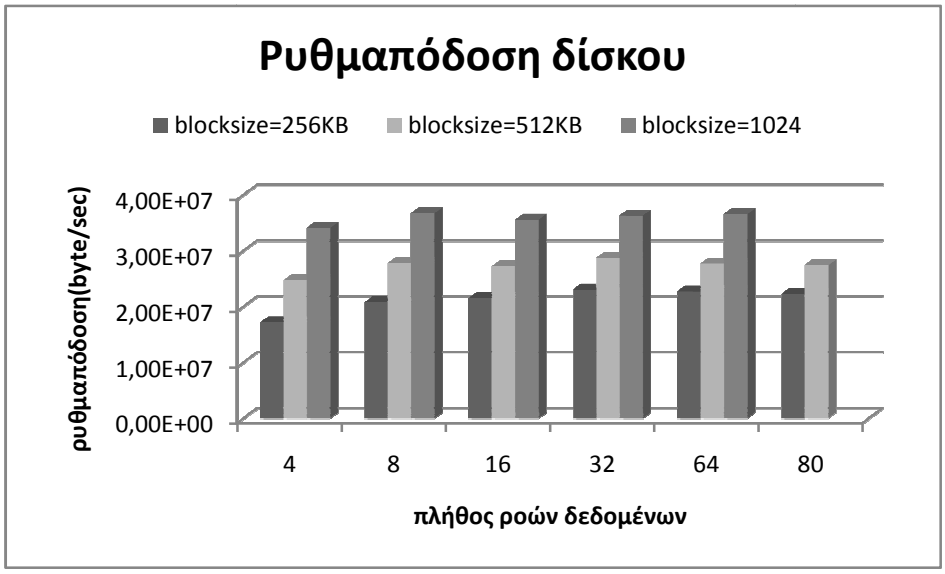
Στο σχήμα 6.3 απεικονίζεται η ρυθμαπόδοση του αλγορίθμου που αναλαμβάνει να συναθροίσει όλα τα μηνύματα της ουράς σε ένα μήνυμα.

Σε πειράματα που είχαν γίνει για να διαπιστωθεί, πως το RPC χρησιμοποιεί το δίκτυο, διαπιστώθηκε ότι η μικρότερη μονάδα που μπορεί να μεταδώσει μέσο του δικτύου είναι 32bit, δηλαδή ένας ακέραιος. Έτσι, όταν δοκιμάσαμε να στείλουμε ένα πίνακα από χαρακτήρες διαπιστώσαμε ότι το RPC εισήγαγε ένα κόστος που ήταν τέσσερις φορές μεγαλύτερο από τα δεδομένα που στέλναμε. Αυτό συνέβαινε, επειδή το RPC χρησιμοποιεί πίνακες ακεραίων στο δίκτυο. Έτσι, όταν έπρεπε να μεταδώσει ένα πίνακα από χαρακτήρες, πρόσθετε ένα χαρακτήρα σε μια θέση του πίνακα ακεραίων που αντιστοιχεί στο δίκτυο. Όμως, με αυτό τον τρόπο χρησιμοποιούσε μόνο τα 8 από τα 32 bit του πίνακα του δικτύου. Λόγω αυτού του κόστους, αναγκαστήκαμε να

χρησιμοποιήσουμε πίνακες ακεραίων για τη μετάδοση των δεδομένων στο δίκτυο, περνώντας σε κάθε θέση του πίνακα ακεραίων τέσσερις χαρακτήρες. Έτσι, ο αλγόριθμος συνάθροισης δέχεται ως είσοδο ένα αριθμό από πίνακές χαρακτήρων, όσα και τα στοιχεία της ουράς, και τους συναθροίζει σε ένα πίνακα χαρακτήρων. Ουσιαστικά, αυτό που κάνει ο αλγόριθμός συνάθροισης είναι να εκτελεί πολλά μικρά memcpy (αντιγράφει 4 χαρακτήρων σε ένα ακέραιο) και μετράμε το χρόνο για να ολοκληρωθούν αυτές οι μικρές αντιγραφές. Η ρυθμαπόδοση του αλγορίθμου προκύπτει από τον αριθμό των χαρακτήρων που αντιγράφτηκαν στο τελικό πίνακα ακεραίων προς τον παραπάνω χρόνο.

Από σχήμα 6.3, αυτό που παρατηρούμε είναι ότι η ρυθμαπόδοση δεν επηρεάζεται από μικρό πλήθος ροών, ενώ αντίθετα σε μεγάλο αριθμό ροών μειώνεται. Το πλήθος των ροών επηρεάζει τη ρυθμαπόδοση της μνήμης και του επεξεργαστεί του συστήματος. Όλα τα νήματα του διακομιστή διανομής ανήκουν στην ίδια διεργασία, άρα όσο περισσότερα νήματα (περισσότερες ροές) έχω τόσο θα μειώνεται η ρυθμαπόδοση του επεξεργαστή και άρα θα καθυστερούν να πραγματοποιηθούν οι μικρές αντιγραφές δεδομένων και άρα θα μειώνεται η απόδοση του αλγορίθμου.

6.2.4. Ρυθμαπόδοση αποθήκευσης δεδομένων στο δίσκο



Σχήμα 6.4: Υπολογίζεται η ρυθμαπόδοση της αποθήκευσης των δεδομένων στο δίσκο σε συνάρτηση με το πλήθος των ταυτόχρονων ροών δεδομένων που υπάρχουν στο σύστημα. Επίσης, εμφανίζονται για κάθε πλήθος ροών δεδομένων μια ράβδο για κάθε διαφορετική τιμή μπλοκ που επιλέχθηκε.

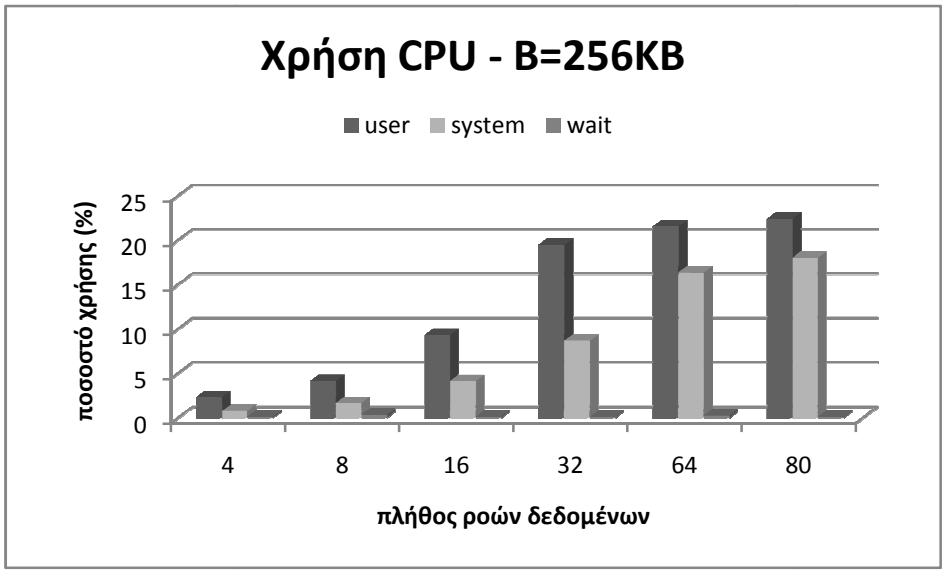
Στο σχήμα 6.4 φαίνεται πως επηρεάζεται η ρυθμαπόδοση της αποθήκευσης των δεδομένων στο δίσκο σε συνάρτηση με το πλήθος των ροών δεδομένων που πρέπει να στείλει το σύστημα μας για αποθήκευση αλλά και σε σχέση με τη τιμή του μπλοκ που επιλέγεται. Οι μετρήσεις αυτές έχουν γίνει στους διακομιστές αποθήκευσης οι οποίοι είναι υπεύθυνοι να αποθηκεύουν τις ροές δεδομένων στο δίσκο τους.

Γενικά παρατηρούμε ότι η ρυθμαπόδοση της αποθήκευσης των δεδομένων στο δίσκο δεν επηρεάζεται από τον αριθμό των ροών δεδομένων. Επίσης παρατηρούμε ότι για το ίδιο πλήθος ροών δεδομένων η ρυθμαπόδοση αυξάνεται. Τα δεδομένα κάθε ροής δεδομένων για να αποθηκευτούν στο δίσκο θα πρέπει πρώτα να βρεθεί το αντίστοιχο μπλοκ στο δίσκο για να αποθηκευτεί η πληροφορία (seek time). Αλλά όσο μεγαλύτερο μέγεθος μπλοκ έχουμε τόσο περισσότερα δεδομένα αποθηκεύονται στο δίσκο ανά χρόνο εύρεσης του αντίστοιχου μπλοκ, άρα αυξάνεται και η ρυθμαπόδοση.

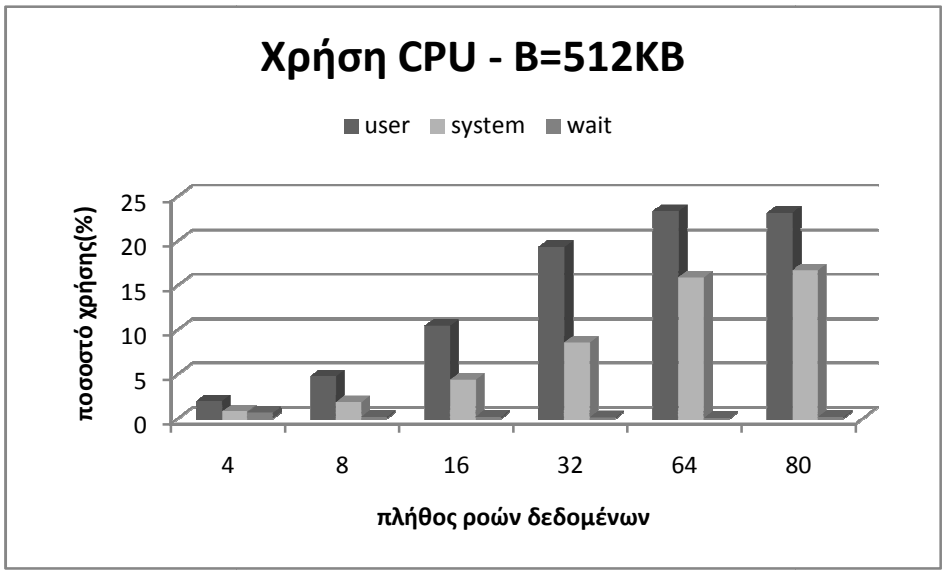
6.3. Ποσοστό χρήσης επεξεργαστή

Σε αυτό το πείραμα υπολογίζουμε το ποσοστό χρήσης του επεξεργαστή του κόμβου στον οποίο τρέχει ο διακομιστής διανομής. Σε αυτό το κόμβο υπάρχουν τέσσερις επεξεργαστές και τα ποσοστά που παρουσιάζονται είναι συνολικά και για τους τέσσερις. Έτσι, για ένα επεξεργαστή το μέγιστο ποσοστό χρήσης είναι 25% του συνολικού.

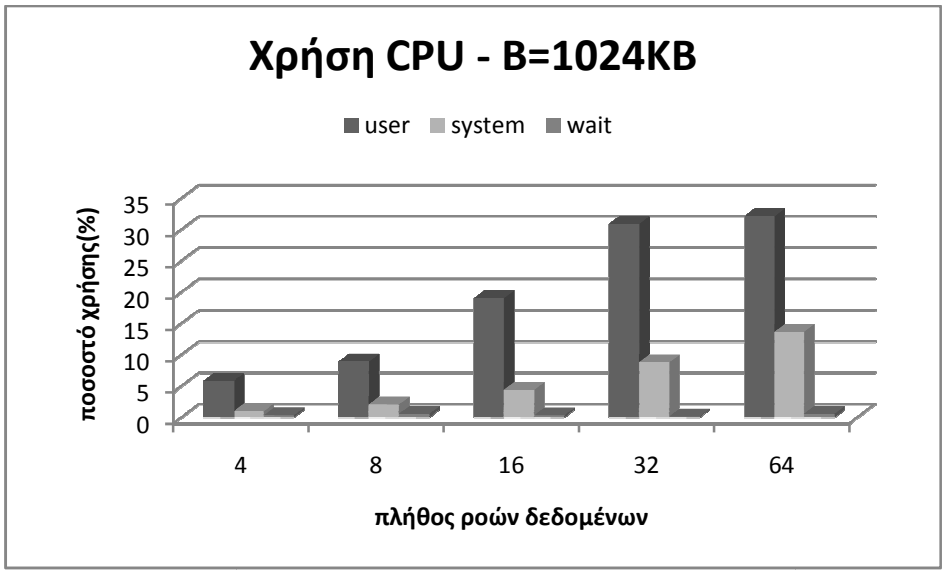
Η χρήση του επεξεργαστή μετρήθηκε όταν ο διακομιστής διανομής αποθηκεύει ροές δεδομένων με μέγεθος αρχείου 10 MB και ρυθμού μετάδοσης 1Mbyte/sec.



Σχήμα 6.5: Υπολογίζεται το ποσοστό χρήσης του επεξεργαστή. Υπολογίζεται το ποσοστό χρήσης του από το χρήστη, από το σύστημα και πόσο αναμένει με μέγεθος μπλοκ 256KB



Σχήμα 6.6: Υπολογίζεται το ποσοστό χρήσης του επεξεργαστή. Υπολογίζεται το ποσοστό χρήσης του από το χρήστη, από το σύστημα και πόσο αναμένει με μέγεθος μπλοκ 512KB



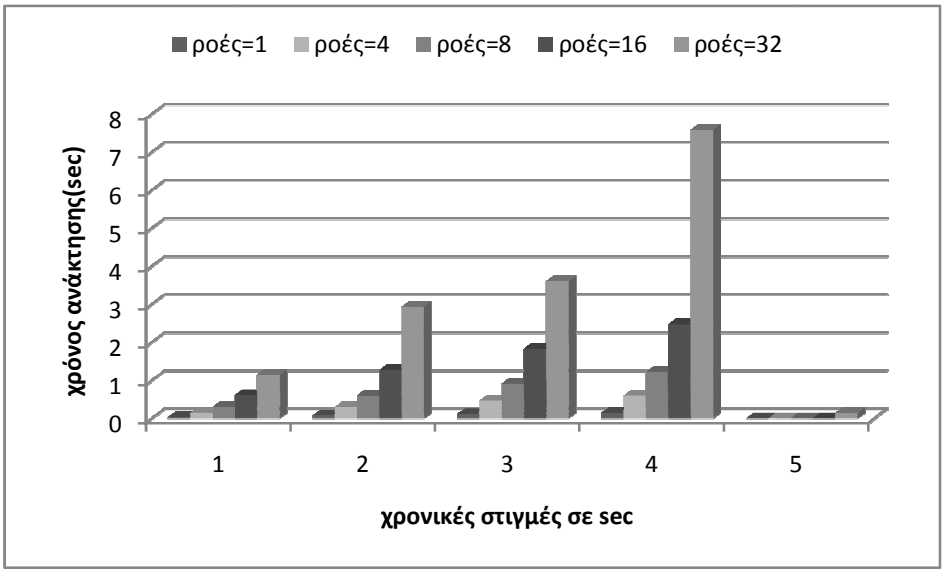
Σχήμα 6.7: Υπολογίζεται το ποσοστό χρήσης του επεξεργαστή. Υπολογίζεται το ποσοστό χρήσης του από το χρήστη, από το σύστημα και πόσο αναμένει με μέγεθος μπλοκ 1024KB

Παραπάνω παρουσιάζονται τα ποσοστά χρησιμοποίησης του επεξεργαστή για τρεις τιμές μπλοκ σε συνάρτηση του πλήθους των ροών δεδομένων.

Όπως έχουμε αναφέρει για να εκτελεστεί ο διακομιστής διανομής χρησιμοποιείται μια μόνο διεργασία η οποία δημιουργεί ένα αριθμός νημάτων επιπέδου χρήστη. Άρα, όλες τις εντολές επιπέδου χρήστη της αναλαμβάνει ένας πυρήνας από τους τέσσερις, οπότε το ποσοστό χρήσης σε επίπεδο χρήστη δε θα πρέπει να ξεπερνά το 25% (ποσοστό που αναλογεί στον ένα πυρήνα) και αυτό φαίνεται από τις παραπάνω γραφικές παραστάσεις.

Βέβαια παρατηρούμε ότι το ποσοστό χρήσης σε επίπεδο χρήστη και επίπεδο συστήματος συνολικά ξεπέρνα το 25%. Αυτό συμβαίνει επειδή ο πυρήνας του λειτουργικού συστήματος μπορεί να αναθέτει τις εργασίες του σε περισσότερους από ένα πυρήνες του επεξεργαστή.
Τέλος, καθώς αυξάνεται το πλήθος των ροών δεδομένων και το μέγεθος του μπλοκ αυξάνεται και το ποσοστό χρήσης του επεξεργαστή, πράγμα λογικό καθώς ο πυρήνας θα έχει περισσότερα δεδομένα να επεξεργαστεί.

6.4. Χρόνος Ανάκτησης Μεταδεδομένων



Σχήμα 6.8: Υπολογίζεται ο χρόνος ανάκτησης των μεταδεδομένων ενός διακομιστή διανομής που έχει καταρρεύσει

Σε αυτό το είδος πειραμάτων μετρήσαμε το χρόνο που χρειάζεται για να ανακτηθούν τα μεταδεδομένα ενός διακομιστή διανομής, ο οποίος έχει καταρρεύσει σε συνάρτηση με τη χρονική στιγμή που έκανε τη διαδικασία περιοδική αποθήκευση μεταδεδομένων, αλλά και με βάση τον αριθμό των ροών δεδομένων που βρίσκονται στο σύστημα. Ο διακομιστής διανομής που έχει καταρρεύσει εκτελούσε τη διαδικασία περιοδικής αποθήκευσης μεταδεδομένων ανά πέντε λεπτά και μετράμε το χρόνο που χρειάζεται για να ανακτηθούν τα δεδομένα του τις χρονικές στιγμές 1,2,3,4 και 5 (σε λεπτά). Ουσιαστικά τις χρονικές στιγμές 1,2,3 και 4 μετράμε το χρόνο που χρειαζόμαστε για να ανακτήσουμε τις εγγραφές δεδομένων από το αρχείο καταγραφής συναλλαγών (BerkeleyDB). Τη χρονική στιγμή 5 έχει πραγματοποιηθεί η διαδικασία περιοδικής αποθήκευσης των μεταδεδομένων και μετράμε το χρόνο για να ανακτήσουμε τα μεταδεδομένα από το αρχείο μεταδεδομένων και μόνο. Τη διαδικασία ανάκτησης την πραγματοποιεί ένα νήμα στον ίδιο διακομιστή διανομής τον οποίο θεωρούμε ότι έχει καταρρεύσει.

Γενικά, στο σχήμα 6.8 παρατηρούμε ότι με το πέρασμα του χρόνου χρειαζόμαστε μεγαλύτερο χρονικό διάστημα για να ανακτήσουμε τα μεταδεδομένα του κόμβου, καθώς τόσο περισσότερες εγγραφές θα πρέπει να ανακτήσουμε από τη βάση δεδομένων. Τη χρονική στιγμή 5 ο χρόνος ανάκτησης είναι πολύ μικρός καθώς πρέπει να διαβάσουμε μόνο το αρχείο μεταδεδομένων αφού η βάση δεδομένων είναι άδεια (έχει γίνει checkpoint).

Επίσης, παρατηρούμε ότι μαζί με τον αριθμό των ροών δεδομένων, αυξάνεται και ο χρόνος ανάκτησης των μεταδεδομένων. Αυτό οφείλεται στο ότι όσες περισσότερες ροές έχουμε τόσες περισσότερες εγγραφές θα προστίθενται στη βάση δεδομένων στη μονάδα του χρόνου και άρα θα απαιτείται περισσότερος χρόνος για να ανακτηθούν αυτές οι εγγραφές.

ΚΕΦΑΛΑΙΟ 7. ΣΥΜΠΕΡΑΣΜΑΤΑ

7.1 Συμπεράσματα

7.1. Συμπεράσματα

Στην παρούσα εργασία σχεδιάστηκε και υλοποιήθηκε ένα σύστημα το οποίο αποθηκεύει ροές δεδομένων με διάφορους ρυθμούς μετάδοσης σε πραγματικό χρόνο.

Το σύστημα αντιμετωπίζει αποτυχίες κόμβων και δίσκων χωρίς να επηρεάζονται οι υπηρεσίες που προσφέρει στους πελάτες που το χρησιμοποιούν.

Το σύστημα μας είναι κλιμακώσιμο καθώς κατανέμει τα μεταδεδομένα του συστήματος σε πολλούς κόμβους και διατηρεί αντίγραφα των μεταδεδομένων του εκάστοτε κόμβο σε άλλους κόμβους.

Το σύστημα μας μπορεί και συναθροίζει δεδομένα διαφορετικών ροών δεδομένων στο ίδιο μήνυμα εξοικονομώντας έτσι εύρος ζώνης για το σύστημα.

ΑΝΑΦΟΡΕΣ

[1] S. V. Anastasiadis, K. C. Sevcik, and M. Stumm.Scalable and fault-tolerant support for variable bit-rate data in the exedra streaming server. *ACM Transactions on Storage*, 1(4):419–456,2005..

[2] S. Berson, L.Golubchik, and R. R. Muntz. Fault tolerant design of multimedia servers. In *ACM SIGMOD Conference*, pages 364–375, San Jose,CA, June 1995.

[3] A. D. Birrel and B. J. Nelson. Implementing remote procedure calls. *ACM Transactions on Computer Systems*, 2(1):39–59, Feb. 1984.

[4] J. Bloomer. *Power Programming with RPC*. O'Reilly and Associates, Inc, Sebastopol, CA, 1992.

[5] W. J. Bolosky, W. J. Bolosky, III, J. S. Barrera, J. S. Barrera, R. P. Draves, R. P. Draves, R. P. Fitzgerald, R. P. Fitzgerald, G. A. Gibson, G. A. Gibson, M. B. Jones, M. B. Jones, S. P. Levi, S. P. Levi, N. P. Myhrvold, N. P. Myhrvold, R. F. Rashid, and R. F. Rashid. The tiger video fileserver. In *Intl Workshop on Network and Operating System Support for Digital Audio and Video*, pages 97–104, Apr. 1996.

[6] M. A. el malek, W. V. C. Ii, C. Cranor, G. R. Ganger, J. Hendricks, A. J. Klosterman, M. Mesnier, M. Prasad, B. Salmon, O. Salmon, R. R. Sambasivan, S. Sinnamohideen, J. D.Strunk, E. Thereska, M.Wachs, and J. Y.Wylie. Ursa minor: versatile cluster-based storage. In *USENIX Conference on File and Storage Technologies*, pages 59–72, San Francisco, CA, 2005.

[7] M. Mesnier, G. R. Ganger, and E. Reidel. Object-based storage. *IEEE Communications Magazine*, pages 84–90, Aug. 2003.

[8] T. K. Moon. *Error Correction Coding*. Wiley-Interscience, Hoboken, NJ, 2005.

[9] R. H. Morelos-Zaragoza. *The Art of Error Correcting Coding*. John Wiley, Hoboken, NJ, second edition edition, 2006.

[10] A. Mourad. Doubly-striped disk mirroring: Reliable storage for video servers. *Multimedia Tools and Applications*, 2:273–297, 1996.

[11] B. Ozden, R. Rastogi, P. Shenoy, and A. Silberschatz. Fault-tolerant architectures for continuous media servers. In ACM SIGMOD Conference, pages 79–90, June 1996.
[12]J. S. Plank. A tutorial on reed-solomon coding for fault-tolerance in raid-like systems. *Software–Practice and Experience*, 27(9):995–1010, Sept. 1997.

[13] S. Quinlan and S. Dorward. Venti: a new approach to archival storage. In *USENIX Conference on File and Storage Technologies*, pages 89–101, Monterey, CA, Jan. 2002.

[14] Y. Saito, S. Frolund, A. Veitch, A. Merchant, and S. Spence. Fab: Building distributed enterprise disk arrays from commodity components. In *ACM Conference on Architectura Support for Programming Languages and Operating Systems*, pages 48–58, Boston, MA, Oct. 2004.

[15] J. Satran, K. Meth, C. Sapuntzakiss, M. Chadalapaka, and E. Zeidner. Rfc3270: Internet small computer systems interface (iscsi).

[16] F. A. Tobagi, J. Pang, R. Baird, and M. Gang.Streaming Raid: A disk array management system for video files. In *ACM Multimedia*, pages 393–400, Anaheim, CA, 1993.

[17] B. Welch, M. Unangst, Z. Abbasi, G. Gibson, B. Mueller, J. Small, J. Zelenka, and B. Zhou. Scalable performance of the panasas parallel file system. In *USENIX Conference on File and Storage Technologies*, pages 17–33, San Jose, CA, Feb. 2008.

[18] B. Zhu, K. Li, and H. Patterson. Avoiding the disk bottleneck in the data domain Deduplication file system. In *USENIX Conference on File and Storage Technologies*, pages 269–282, San Jose, CA, Feb. 2008.

ΣΥΝΤΟΜΟ ΒΙΟΓΡΑΦΙΚΟ

Ο Μενέλαος Μαρκουλάκης γεννήθηκε το 1985 στην Κρήτη όπου και ολοκλήρωσε τις λυκειακές σπουδές του το 2002. Το 2002 ξεκίνησε τις σπουδές του στο Τμήμα Πληροφορικής του Πανεπιστημίου Ιωαννίνων τις οποίες ολοκλήρωσε το 2006. Από το Σεπτέμβριο του 2006 είναι μεταπτυχιακός φοιτητής του Τμήματος Πληροφορικής του Πανεπιστημίου Ιωαννίνων. Τα ερευνητικά του ενδιαφέροντα εστιάζονται σε θέματα αποθήκευσης ροών δεδομένων.