ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ
&
ΜΗΧΑΝΙΚΩΝ Η/Υ
ΤΟΜΕΑΣ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΠΛΗΡΟΦΟΡΙΑΣ ΚΑΙ ΔΙΚΤΥΩΝ

Διπλωματική Εργασία

"*Study of the Impact of Cache Replacement Algorithms Associated With Different Video Segmentation Strategies on Video Caching*"

Χριστοφόρου Χριστόφορος
A.M.: 9830497

Εξεταστική επιτροπή:

καθ. Πατεράκης Μιχάλης *(επιβλέπων)*
καθ. Διγαλάκης Βασίλης
καθ. Σιδηρόπουλος Νίκος

Χανιά Ιούλιος 2003

ΠΡΟΛΟΓΟΣ

Στις μέρες μας, εφαρμογές που αφορούν ζήτηση και μετάδοση video αρχείων (video retrieval and streaming), τόσο σε τοπικά δίκτυα όσο και σε δίκτυα ευρείας γεωγραφικής κάλυψης, αποτελούν ένα από τους πιο γρήγορα αναπτυσσόμενους αλλά και πιο απαιτητικούς κλάδους των δικτύων τηλεπικοινωνιών.

Η παρουσία των Proxy Servers καθίσταται αναγκαία για την αποδοτικότερη λειτουργία του δικτύου και την καλύτερη εξυπηρέτηση των χρηστών. Η εργασία αυτή μελετά τρόπους αποθήκευσης (caching) video δεδομένων σε μία Proxy Cache. Συγκεκριμένα παρουσιάζονται και μελετούνται δύο πολιτικές τοποθέτησης δεδομένων στην cache και δύο πολιτικές αντικατάστασης των δεδομένων από αυτή. Κατά την πρώτη πολιτική τοποθέτησης δεδομένων, “Fixed Chunk Size”, στην cache τοποθετούνται πάντα κομμάτια των video σταθερού μεγέθους ενώ σύμφωνα με την δεύτερη πολιτική, “Variable Chunk Size”, τα καινούργια κομμάτια ενός video που τοποθετούνται στην cache έχουν μεταβλητό μέγεθος το οποίο αυξάνεται ανάλογα με την ποσότητα του video που υπάρχει ήδη αποθηκευμένο σε αυτήν. Εφαρμόζοντας την πρώτη μέθοδο αντικατάστασης, “Least Recently Used”, τα video απομακρύνονται από την cache σύμφωνα με τον χρόνο κατά τον οποίο έγινε η τελευταία αίτηση ένα video. Το video που δεν ζητήθηκε για μεγαλύτερο χρονικό διάστημα επιλέγεται ως θύμα. Χρησιμοποιώντας την δεύτερη μέθοδο αντικατάστασης, “Least Frequently Least Recently Used”, θύμα επιλέγεται το video για το οποίο έχουν γίνει οι λιγότερες αιτήσεις. Αν υπάρχουν περισσότερα από ένα video με τις ίδιες αιτήσεις, θύμα μεταξύ αυτών επιλέγεται το video που δεν έχει ζητηθεί για μεγαλύτερο χρονικό διάστημα.

Όλοι οι συνδυασμοί των παραπάνω τεχνικών αποθήκευσης και αντικατάστασης αξιολογούνται μέσω προσομοίωσης. Εξετάζονται οι παρακάτω δύο περιπτώσεις : i) η κατανομή των πιθανοτήτων σύμφωνα με τις οποίες ζητούνται τα video από τους πελάτες του συστήματος είναι σταθερή σε όλη την διάρκεια της προσομοίωσης και ii) η κατανομή αυτή μεταβάλλεται δυναμικά μετά την πάροδο ορισμένου χρόνου. Η αξιολόγηση της απόδοσης του συστήματος γίνεται με βάση τις παρακάτω δύο μετρικές: i) το “Byte Hit Ratio” (κλάσμα του όγκου των ζητούμενων video δεδομένων που βρίσκονται στην cache) και ii) το “Delay Start” (κλάσμα των αιτήσεων των χρηστών που δεν βρίσκουν κανένα τμήμα του ζητούμενου video στην cache).

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΕΦΑΛΑΙΟ 1 ΕΙΣΑΓΩΓΗ

Η καινούργια αντίληψη του χώρου και χρόνου που εισήγαγε το διαδίκτυο στην ζωή των ολοένα αυξανόμενων χρηστών του είναι αναμφισβήτητη. Ο χώρος συμπιέστηκε στις διαστάσεις μιας οθόνης υπολογιστή και ο χρόνος σε κλάσματα δευτερολέπτου.

Η δυνατότητα ανταλλαγής δεδομένων πληροφορίας ανάμεσα σε χρήστες του διαδικτύου που βρίσκονται σε διαφορετικές τοποθεσίες, ανεξαρτήτως χωρικών αποστάσεων, σε χρόνους σχεδόν μηδενικούς οδήγησε στην δημιουργία νέων εφαρφογών που αποσκοπούν στην καλύτερη εξυπηρέτηση των αναγκών των χρηστών. Οι εφαρμογές αυτές αφορούν θέματα ενημέρωσης, ψυχαγωγίας, εκπαίδευσης καθώς και θέματα επαγγελματικού ενδιαφέροντος. Επιπρόσθετα, το σχετικά χαμηλό κόστος χρησιμοποίησης του διαδικτύου, οδήγησαν στην γρήγορη εξάπλωσή του σε μεγάλο αριθμό χρηστών.

Το γεγονός αυτό είχε ως αποτέλεσμα την δημιουργία σημείων συμφόρησης τόσο στις «πηγές» δεδομένων όσο και κατά μήκος των «μονοπατιών» του δικτύου. Με την πάροδο του χρόνου το φαινόμενο αυτό παίρνει μεγάλες διαστάσεις και περιορίζει την ποιότητα της εξυπηρέτησης που προσφέρεται στους χρήστες του δικτύου.

1.1 Web Caches και Proxy Servers

Κάθε χρήστης που συνδέεται στο διαδίκτυο, σε κάποιο σημείο επικοινωνεί με ένα διακομιστή μεσολάβησης. Οι διακομιστές μεσολάβησης έχουν σκοπό τη μείωση της κίνησης στις ζεύξεις του δικτύου και την αύξηση της απόδοσης και της ασφάλειας των υπηρεσιών που παρέχονται. Επιχειρήσεις, οργανισμοί και ιδρύματα που χρειάζονται πρόσβαση στο διαδίκτυο αλλά παράλληλα απαιτούν να διατηρήσουν την ασφάλεια του ιδιωτικού εσωτερικού τους δικτύου χρησιμοποιούν ειδικούς διακομιστές μεσολάβησης όπως Web Caches και Proxy Servers.

1.1.1 Τι είναι οι Web Caches

Γενικά cache είναι μια περιοχή μνήμης όπου αποθηκεύεται πληροφορία στην οποία μπορεί να έχει πρόσβαση πολύ γρήγορα ο εκάστοτε χρήστης που θέλει να τη χρησιμοποιήσει. Η cache συνήθως τοποθετείται μεταξύ μιας γρήγορης και μιας αργής

συσκευής. Συγκεκριμένα μια Web Cache είναι εγκατεστημένη ανάμεσα σε Web Servers (Origin Servers) και χρήστες (Clients). Η λογική του Web Caching είναι σχετικά απλή : κάθε φορά που ένας χρήστης κάνει μια αίτηση για ένα Web αντικείμενο το οποίο βρίσκεται σε ένα Web Server διαμέσου μιας Web Cache, η Web Cache αποθηκεύει ένα αντίγραφο του αντικειμένου αυτού με σκοπό την επόμενη φορά που θα ζητηθεί το ίδιο αντικείμενο από τον ίδιο ή άλλο χρήστη, το αντικείμενο να παραδοθεί στον χρήστη από την cache χωρίς να είναι αναγκαίο ο Origin Server να ξαναστείλει το αντικείμενο αυτό.

Υπάρχουν διάφορα είδη από Web Caches ανάλογα με το τι θεωρεί κάθε cache σημαντικό για να αποθηκευτεί και να διαχειριστεί. Κάποιες caches αποθηκεύουν μόνο γραφικά, άλλες μόνο ιστοσελίδες, άλλες αρχεία πολυμέσων (Video & Audio Files) και άλλες τα πάντα. Συνήθως οι Web caches αποθηκεύουν αντικείμενα για περιορισμένο χρονικό διάστημα που μπορεί να είναι λίγες ώρες μια μέρα ή μια εβδομάδα. Όταν αυτό το διάστημα τελειώσει και το αντικείμενο ζητηθεί, η cache θα ζητήσει από τον origin server να επικυρώσει αν το αντικείμενο έχει αλλάξει (ενημερωθεί) από την τελευταία φορά που αυτό στάλθηκε από τον origin server στην cache. Αν έχει ενημερωθεί, ο origin server το ξαναστέλνει στην cache. Το χρονικό αυτό διάστημα μεταβάλλεται ανάλογα με το είδος του αντικειμένου και την συχνότητα που αυτό συνήθως ενημερώνεται.

1.1.2 Τι είναι οι Proxy Servers

Ο proxy server είναι μια διαδικασία η οποία ανασύρει σελίδες από το διαδίκτυο και τις αποθηκεύει ώστε να είναι διαθέσιμες αργότερα στον ίδιο ή σε χρήστες του δικτύου που θα τις αναζητήσουν. Ο proxy server αναλαμβάνει να παίξει το ρόλο του ενδιάμεσου ανάμεσα στον χρήστη και τον origin server, από τον οποίο ο πρώτος ζητά να δεί κάποια σελίδα ή να κατεβάσει κάποιο αρχείο. Έτσι, προκειμένου να διεκπεραιώσει το αίτημα του χρήστη, ο proxy server, λειτουργώντας ως διεργασία αντιπρόσωπος, αρχικά ελέγχει αν διαθέτει την πληροφορία που ζητείται. Αν την διαθέτει, τη στέλνει κατευθείαν χωρίς να χρειάζεται ο χρήστης να περιμένει να πάρει την πληροφορία από τον origin server. Αν δεν έχει διαθέσιμη την πληροφορία αυτή, τότε αναλαμβάνει να την κατεβάσει αυτός και ακολούθως τη στέλνει στον χρήστη, ενώ ταυτόχρονα την αποθηκεύει προσωρινά για την περίπτωση που του ζητηθεί ξανά η ίδια πληροφορία στο εγγύς μέλλον.

1.1.3 Πλεονεκτήματα του Proxy Caching

Δύο είναι οι κύριοι λόγοι επιτυχίας του proxy caching :

- Αυξάνει την ταχύτητα περιήγησης στον παγκόσμιο ιστό αφού οι διάφορες αναζητήσεις ικανοποιούνται σε σημαντικό βαθμό από τον proxy cache ο οποίος βρίσκεται πιο κοντά στον χρήστη, αντί από τον εκάστοτε origin server. Γι' αυτό το λόγο ο χρήστης παίρνει τα δεδομένα που ζητά πολύ πιο γρήγορα και οι ιστοσελίδες φαίνεται να ανταποκρίνονται γρήγορα στις διάφορες αιτήσεις.
- Μειώνεται η κίνηση στις γραμμές του δικτύου, διότι για ένα αριθμό αιτήσεων για ένα αντικείμενο το τελευταίο κατεβαίνει από τον origin server μόνο μια φορά. Έτσι το bandwidth που χρησιμοποιείται από κάθε χρήστη για την ικανοποίηση των απαιτήσεών του είναι λιγότερο με αποτέλεσμα η εκμετάλλευση του bandwidth του δικτύου να είναι γίνεται με πιο αποδοτικό τρόπο.

1.2 Δομή της διπλωματικής εργασίας

Παράλληλα με την εκρηκτική εξάπλωση του Internet αυξήθηκαν σημαντικά και οι απαιτήσεις των χρηστών που αφορούν video εφαρμογές. Η τάση αυτή αναμένεται να συνεχιστεί με αποτέλεσμα στο μέλλον η συμφόρηση που θα παρατηρείται στο διαδίκτυο να οφείλεται σε μεγάλο βαθμό σε εφαρμογές μετάδοσης video αρχείων. Αμφισβήτητα, για την αντιμετώπιση του προβλήματος αυτού σημαντικό ρόλο θα παίξουν οι proxy caches. Για αυτό το λόγο αρκετοί ερευνητές τα τελευταία χρόνια εξέτασαν τρόπους για καλύτερη εκμετάλλευση του χώρου που προσφέρει μία cache, που υποστηρίζει video εφαρμογές.

Πολλά σχήματα αποθήκευσης έχουν αναπτυχθεί για τη διαχείριση εφαρμογών video. Αρχικά τα video αντιμετωπίστηκαν ως οντότητες οι οποίες είτε αποθηκεύονταν ολόκληρες ή καθόλου [1, 2, 3]. Πρόσφατα αναπτύχθηκε η τεχνική του καταμερισμού των βίντεο σε κομμάτια. Στο [4, 5] αποθηκεύεται στην cache μόνο το αρχικό κομμάτι των βίντεο ενώ το υπόλοιπο ζητείται συνήθως από τον origin video server ή λαμβάνεται δια μέσου μιας multicast μετάδοσης που εξυπηρετεί παραπάνω από ένα χρήστες τη δεδομένη στιγμή. Επίσης, μελετήθηκε η αποθήκευση video τα οποία είναι κωδικοποιημένα σε επίπεδα (layers) [6, 7]. Στο [8], τα video

καταμερίζονται σε εκθετικά κομμάτια και σύμφωνα με κάποια κριτήρια αντικατάστασης τοποθετούνται στην cache ή απομακρύνονται από αυτή.

Η διπλωματική αυτή εργασία, η οποία έχει άμεση σχέση με τις εργασίες στα [9, 10], μελετά τρόπους αποθήκευσης (caching) και αντικατάστασης video δεδομένων σε μία Proxy Cache με στόχο την αποδοτικότερη λειτουργία του δικτύου και την καλύτερη εξυπηρέτηση των χρηστών. Στο κεφάλαιο που ακολουθεί παρουσιάζεται η σχέση και ο τρόπος αλληλεπίδρασης μεταξύ του χρήστη, της cache και του origin server. Όπως στο [9], χρησιμοποιούμε δύο τεχνικές αποθήκευσης video δεδομένων στην cache: αυτή με Fixed Chunk Size (FCS) και αυτή με Variable Chunk Size (VCS). Χρησιμοποιούμε επίσης δύο αλγόριθμους απομάκρυνσης δεδομένων από την cache: Least Recently Used (χρησιμοποιήθηκε στο [9]) και Least Frequently Least Recently Used. Στο ίδιο κεφάλαιο παρουσιάζονται το μοντέλο και οι μετρικές απόδοσης του συστήματος. Στο κεφάλαιο 3 παρουσιάζονται τα αποτελέσματα προσομοιώσεων του συστήματος όταν η κατανομή των πιθανοτήτων σύμφωνα με την οποία ζητούνται τα video από τους χρήστες του δικτύου είναι στατική. Η αξιολόγηση των αποτελεσμάτων επικεντρώνεται στην επίδραση που έχουν οι δύο αλγόριθμοι αντικατάστασης LRU και LFLRU όταν εφαρμόζονται σε συνδυασμό με την τεχνική τοποθέτησης FCS, ή με την τεχνική VCS. Το κεφάλαιο 4 ασχολείται με αποτελέσματα προσομοιώσεων όπου κατά την διάρκεια της προσομοίωσης η κατανομή των πιθανοτήτων αναζήτησης των video υφίσταται αλλαγές, με σκοπό να προσεγγίσομε σε μεγαλύτερο βαθμό τις πραγματικές συνθήκες λειτουργίας μιας proxy cache. Παράλληλα αξιολογείται η συμπεριφορά των δύο αλγορίθμων αντικατάστασης σε σχέση με τις τεχνικές αποθήκευσης. Τέλος, στο κεφάλαιο 5 συνοψίζονται τα κύρια συμπεράσματα που εξάγονται από την διπλωματική εργασία και παρατίθενται μερικές ιδέες για μελλοντική μελέτη με στόχο την καλύτερη εκμετάλλευση των δυνατοτήτων της proxy cache που υλοποιεί video caching.

ΚΕΦΑΛΑΙΟ 2
Video Caching Και Τρόποι Υλοποίησης Του

Ερευνητές έχουν ασχοληθεί με τον σχεδιασμό τεχνικών και αλγορίθμων που αποσκοπούν στη βελτίωση της απόδοσης του παγκόσμιου ιστού (World Wide Web) από την αρχή της δεκαετίας του '90. Η αντιγραφή δημοφιλών αντικειμένων (caching) σε θέσεις κοντά στους χρήστες (clients) αποδείχθηκε μια πολύ αποτελεσματική λύση, που μειώνει την κυκλοφορία της πληροφορίας στο διαδίκτυο και κατανέμει το φόρτο εργασίας σε περισσότερους υπολογιστές/εξυπηρετητές (servers). Η αντιγραφή δεδομένων έχει ως αποτέλεσμα όχι μόνο την καλύτερη χρησιμοποίηση του εύρους ζώνης του δικτύου αλλά και τη μείωση της καθυστέρησης πρόσβασης που βιώνουν οι χρήστες. Η αποθήκευση αυτή έχει φτηνότερο κόστος υλοποίησης σε σύγκριση με την αύξηση του εύρους ζώνης του δικτύου που διαφορετικά θα απαιτείτο, κάτι που καθιστά την ιδέα του caching πολύ ελκυστική.

Εντούτοις, τα αντικείμενα πολυμέσων (multicast objects) διαφέρουν σημαντικά από τα παραδοσιακά αντικείμενα που αφορούν κείμενο ή στατικές εικόνες. Το μέγεθος των αρχείων video είναι συνήθως αρκετές τάξεις μεγέθους μεγαλύτερο από το μέσο μέγεθος των παραδοσιακών αρχείων κειμένου (HTML). Επίσης, η φύση των video δεδομένων και οι απαιτήσεις των αντίστοιχων εφαρμογών προσδίδουν ένα αρκετά υψηλό βαθμό δυσκολίας και πολυπλοκότητας στην σχεδίαση και υλοποίηση τόσο των κεντρικών υπολογιστών (servers) όσο και των αποθηκευτικών χώρων (proxy caches).

2.1 Γενικά για τις καταστάσεις μιας Video Proxy Cache

Η κυκλοφορία πολυμέσων στο διαδικτύο έχει αλλάξει εντυπωσιακά τους παραδοσιακούς τρόπους λειτουργίας των proxy cache servers. Τα αντικείμενα πολυμέσων έχουν τέτοιες ιδιότητες που απαιτούν διαφορετική προσέγγιση: το μέγεθος τους συνήθως κάνει αδύνατη την αποθήκευση ολόκληρου του αντικείμενου στην τοπική proxy cache, μια τεχνική που ακολουθείται από τη λογική Web Caching. Επιπλέον, η εξυπηρέτηση μιας αίτησης χρήστη για ένα μερικώς αποθηκευμένο αντικείμενο πολυμέσων αναγκάζει τον proxy να έρθει σε επαφή με τον απομακρυσμένο κεντρικό υπολογιστή (origin server) που έχει ολόκληρο το αντικείμενο προκειμένου να ανακτηθούν τα ελλείποντα μέρη. Συνήθως απαιτείται

όλες οι ενέργειες που γίνονται από τον proxy cache server να μην έχουν επιπτώσεις στην εξυπηρέτηση του αίτησης του χρήστη και να παραμένουν μη φανερές σε αυτόν, αυξάνοντας κατά συνέπεια την πολυπλοκότητα του σχεδιασμού του συστήματος.

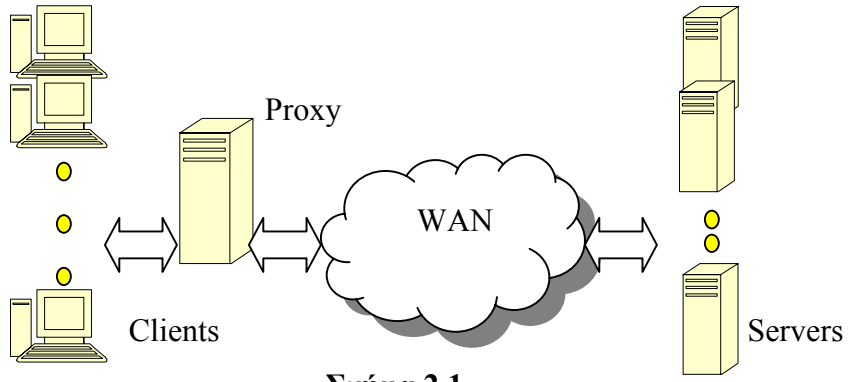
Στη συνέχεια ακολουθεί μια σύντομη και απλοποιημένη περιγραφή της λειτουργίας ενός video proxy cache. Ο πελάτης αρχικά έρχεται σε επαφή με τον proxy cache server, ζητώντας ένα αντικείμενο πολυμέσων όπως θα το ζητούσε από τον μακρινό origin server όπου βρίσκεται αποθηκευμένο ολόκληρο το ζητούμενο αντικείμενο. Τρία είναι τα πιθανά σενάρια που μπορούν να συμβούν με τη λήψη ενός τέτοιου αιτήματος :

- *Ολόκληρο το video βρίσκεται στην cache (cache hit):* σε αυτή την περίπτωση ο proxy cache server θα λειτουργήσει σαν να ήταν ο πραγματικός origin server και θα ξεκινήσει την παράδοση του video στον χρήστη που το ζήτησε με την ελάχιστη δυνατή καθυστέρηση.
- *Το video δεν βρίσκεται στην cache (cache miss):* ο proxy πρέπει να προσκομίσει το ζητούμενο video από το μακρινό server για λογαριασμό του πελάτη. Προς τούτο, ο proxy server εγκαθιδρύει μια σύνδεση με τον απομακρυσμένο κεντρικό υπολογιστή και ο origin server μεταχειρίζεται τον proxy cache server σαν ένα πελάτη. Ο proxy server διαβιβάζει τα λαμβανόμενα τμήματα του video στον πελάτη, ενώ ταυτόχρονα αποθηκεύει ορισμένα από αυτά στην τοπική cache για πιθανή μελλοντική χρήση από άλλο πελάτη. Η καθυστέρηση απόκρισης του συστήματος στην αίτηση του πελάτη καθορίζεται κυρίως από την καθυστέρηση στο τμήμα του διαδικτύου μεταξύ proxy και origin server.
- *Μερικό τμήμα του video βρίσκεται στη cache (partial hit):* από την άποψη του proxy cache server, αυτό είναι το πιο περίπλοκο σενάριο. Συνήθως το παραπάνω μερικό τμήμα του video αντιστοιχεί στα πρώτα λίγα δευτερόλεπτα του video. Ο proxy server ξεκινά την παράδοση του video στον πελάτη που το ζήτησε ξεκινώντας με το αρχικό κομμάτι που έχει αποθηκευμένο στην cache του, ενώ ταυτόχρονα εγκαθιδρύει σύνδεση με τον απομακρυσμένο κεντρικό υπολογιστή που έχει αποθηκευμένο ολόκληρο το video ώστε να προσκομιστούν έγκαιρα από αυτόν όλα τα τμήματα του video και να παραδοθούν στον πελάτη. Η όλη διαδικασία πρέπει να μην είναι φανερή στον

τελικό χρήστη. Από την στιγμή που θα αρχίσει η παράδοση των τμημάτων του video και η αναπαραγωγή τους (playback) από τον χρήστη, δεν πρέπει ο τελευταίος να βιώσει διακοπή του video playback λόγω αργοπορημένης παράδοσης σ' αυτόν video τμημάτων από τον proxy.

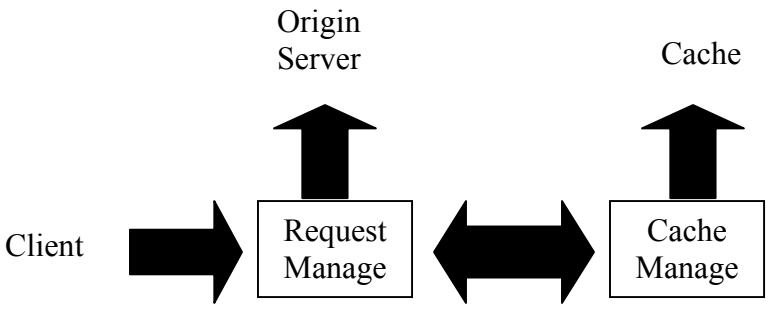
Είναι σαφές ότι ο proxy cache server ενεργεί ως πελάτης (client), εξυπηρετητής (origin server), cache και ως συντονιστής της διαδικασίας παράδοσης των video αντικειμένων στους πελάτες που τα ζητούν.

2.2 Αρχιτεκτονική και λειτουργία ενός συστήματος Client - Video Proxy Cache - Origin Server



Σχήμα 2.1

Το πιο πάνω σχήμα , εξηγεί την τοπολογία του συστήματος διανομής video που συζητάμε. Τα video βρίσκονται αποθηκευμένα σε γεωγραφικά διασκορπισμένους κεντρικούς υπολογιστές (servers). Ένας video proxy server υπολογιστής εγκαθίσταται στο δίκτυο πλησίον μιας ομάδας πελατών. Τα αιτήματα για τα video αυτών των πελατών κατευθύνονται στον proxy, ο οποίος τα ικανοποιεί είτε από την τοπική του cache είτε μέσω σύνδεσής του διά του δικτύου WAN με τους αντίστοιχους κάθε φορά απομακρυσμένους κεντρικούς υπολογιστές (origin servers). Εν γένει, υποτίθεται ότι υπάρχει αρκετό διαθέσιμο εύρος ζώνης στις συνδέσεις μεταξύ του proxy και των πελατών για να υποστηριχθεί η ροή του video (streaming). Η μετάδοση video ροών από τους origin servers μέσω του δικτύου ευρείας περιοχής προς τον proxy υποτίθεται ακριβή, δεδομένου ότι καταναλώνει τμήμα του εύρους ζώνης του διαδικτύου. Ο proxy cache server αποθηκεύει τμήματα των video προσπαθώντας αφενός να βελτιώσει το χρόνο απόκρισης σε αιτήσεις των πελατών και αφετέρου να μειώσει τον όγκο των αντικειμένων video που μεταδίδονται στο ευρύτερο δίκτυο.



Σχήμα 2.2

Το σχήμα 2.2 επεξηγεί την εσωτερική αρχιτεκτονική ενός proxy server. Αποτελείται από δύο σημαντικές οντότητες: το Request Manager και το Cache Manager. Ο Request Manager είναι αρμόδιος για τη συνεχή ροή του video προς τους πελάτες. Γενικά, η ευθύνη του είναι να δρομολογεί τη μετάδοση του prefix (πρόθεμα, το αρχικό κομμάτι του video που βρίσκεται ήδη στην cache) προς τον πελάτη και να διαβιβάσει ένα αίτημα για το suffix(επίθεμα, το υπόλοιπο κομμάτι του video) στον αντίστοιχο origin server. Επιπλέον, είναι απαραίτητο να εξασφαλίσει τη συνεχή ροή του video στον πελάτη. Αυτό επιτυγχάνεται με την χρησιμοποίηση προσωρινών αποθηκευτικών χώρων (buffers) στις συνδέσεις server-proxy και proxy-clients. Η κύρια ευθύνη του Cache Manager είναι να διαθέσει αποτελεσματικά τους πόρους αποθήκευσης του proxy στα ζητούμενα video. Η εργασία αυτή εστιάζει στη λειτουργικότητα του Cache Manager. Υποτίθεται ότι ο Cache Manager αρχίζει αμέσως τη μετάδοση του προθέματος (αν υπάρχει) στον πελάτη και ζητά το επίθεμα από τον κεντρικό υπολογιστή προέλευσης (origin server).

Ο Cache Manager λαμβάνει τα εισερχόμενα δεδομένα από τον origin server και αποφασίζει εάν θα αποθηκευθούν ή όχι στην cache. Όταν η απόφαση είναι να αποθηκευθούν τα πρόσφατα ανακτημένα τμήματα του video, ο Cache Manager αποφασίζει επίσης (α) πόση διάρκεια του εισερχόμενου video θα αποθηκευτεί, (β) ποια από τα νέα εισερχόμενα τμήματα του video θα κρατήσει και (γ) ποια από τα ήδη αποθηκευμένα δεδομένα θα απομακρύνει από την cache για να δημιουργήσει ελεύθερο χώρο για τα νέα τμήματα του video.

Η τρίτη απόφαση είναι στην ουσία η πολιτική αντικατάστασης. Στα παραδοσιακά Web caching σχήματα καθώς και σε μερικά video caching σχήματα, η λειτουργία του Cache Manager περιορίζεται στην απόφαση σχετικά με τις ενέργειες αντικατάστασης. Αυτά τα σχήματα δεν υποθέτουν κατάτμηση του αντικειμένου και συνεπώς δεν ειναι απαραίτητο να αποφασίσουν σχετικά με τα προαναφερθέντα

ζητήματα (α) και (β), τα οποία είναι συγκεκριμένα για αποθήκευση τεμαχισμένων αντικειμένων. Αντίθετα, η εργασία αυτή εστιάζει την μελέτη της σε δύο διαφορετικούς τρόπους κατάτμησης και επικεντρώνει την προσοχή της στην επίδραση που έχουν δύο διαφορετικές λειτουργίες αντικατάστασης σε σχέση με τις δύο τεχνικές κατάτμησης.

Η πρώτη λειτουργία, δηλαδή η απόφαση για το διάστημα που διατίθεται σε κάθε video, όπως θα παρουσιαστεί στη συνέχεια, έχει σημαντικές επιπτώσεις στην απόδοση του proxy. Ο δεύτερος ρόλος του Cache Manager, που είναι η επιλογή των συγκεκριμένων τμημάτων ενός ορισμένου συνολικού μεγέθους του video που θα αποθηκευτούν, δεν έχει επιπτώσεις στην απόδοση από την άποψη του cache hit ratio δεδομένου ότι δεν έχει επιπτώσεις στον όγκο των δεδομένων που αποθηκεύονται στην cache. Εντούτοις, μπορεί να έχει επιπτώσεις στην ποιότητα του video και στην καθυστέρηση αναπαραγωγής του που αντιλαμβάνεται ο χρήστης. Τα παραπάνω τμήματα μπορούν να είναι διαδοχικά πλαίσια του video (video frames) εν γένει σε οποιαδήποτε απόσταση από την αρχή του video. Αυτή η επιλογή έχει επιπτώσεις στην καθυστέρηση του χρήστη (αρχική καθυστέρηση-startup delay, καθυστέρηση λόγω των διαδικασιών VCR που πιθανά χρησιμοποιεί ο χρήστης κ.λ.π.). Σε αυτή την εργασία ο Cache Manager μεταχειρίζεται προνομιακά τα αρχικά διαδοχικά τμήματα ενός video προκειμένου να μειωθεί η αρχική καθυστέρηση αναπαραγωγής του video στον χρήστη. Ένα video αποθηκεύεται αρχίζοντας πάντα από τα αρχικά του τμήματα και προχωρώντας σε σειριακά μέχρι να αποθηκευτεί και το τελευταίο τμήμα του. Κωδικοποίηση του video σε επίπεδα (layers) και διαδικασίες VCR των χρηστών δεν εξετάζονται.

2.3 Τεχνικές Κατάτμησης/Αποθήκευσης

Οι προαναφερθείσες λειτουργίες κατάτμησης ενός video που χρησιμοποιούνται από τον Cache Manager, εκτελούνται ως εξής. Ο Cache Manager χρησιμοποιεί μια μονάδα αντικατάστασης, αποκαλούμενη Chunk. Όταν ένα video που δεν είναι στην cache ζητείται, προσκομίζεται από τον αντίστοιχο server και τα αρχικά τμήματά του, μέχρι το μέγεθος του Chunk αποθηκεύονται στην proxy cache. Σε περίπτωση που δεν υπάρχει αρκετός ελεύθερος χώρος στην cache ο αλγόριθμος αντικατάστασης επιλέγει ένα από τα ήδη αποθηκευμένα video ως θύμα. Το τελευταίο Chunk του video θύματος αφαιρείται από την cache. Κάθε πρόσθετο αίτημα για ένα

video οδηγεί στην αποθήκευση ενός πρόσθετου (διαδοχικού) Chunk. Αυτό εγγυάται ότι μόνο το πρόθεμα (αρχικά διαδοχικά μέρη) κάθε video αποθηκεύεται στην cache. Το μέγεθος του Chunk καθορίζει την επίδοση της διαδικασίας αντικατάστασης και αποθήκευσης και κατά συνέπεια την γενική απόδοση της cache. Στη συνέχεια παρουσιάζονται δύο τεχνικές κατάτμησης που σχετίζονται με την επιλογή του μεγέθους του Chunk: i) Fixed Chunk Size και ii) Variable Chunk Size.

2.3.1 Το Fixed Chunk Size σχήμα

Το Fixed Chunk Size (FCS) σχήμα [9, 10] έχει το πλεονέκτημα ότι είναι ιδιαίτερα απλό. Σύμφωνα με αυτό το μέγεθος του Chunk μπορεί να ποικίλει από μερικά πλαίσια video μέχρι και το πλήρες video. Εφόσον επιλεγεί, παραμένει σταθερό και ίδιο για όλα τα video και για όλες τις αιτήσεις. Όπως προαναφέρθηκε κάθε αίτηση για ένα video οδηγεί στην αποθήκευση ενός πρόσθετου Chunk στην cache. Όταν το ελλείπον μέρος ενός ζητούμενου video είναι μικρότερο από το μέγεθος του Chunk, όλα τα ελλείποντα τμήματα αποθηκεύονται στην cache. Όταν το μέγεθος του Chunk επιλέγεται ίσο με ένα πλήρες video, τότε το Fixed Chunk Size σχήμα μετατρέπεται στο παραδοσιακό Web Caching σχήμα.

2.3.2 Το Variable Chunk Size σχήμα

Σύμφωνα με το Variable Chunk Size (VCS) σχήμα [9, 10], το μέγεθος του Chunk καθορίζεται από τη χρονική στιγμή που υποβάλλεται η αίτηση για το συγκεκριμένο video και η τιμή του από την ποσότητα του ήδη αποθηκευμένου στην cache μέρους του video. Αυτό σημαίνει ότι το μέγεθος του Chunk δεν είναι το ίδιο για όλες τις αιτήσεις για ένα δεδομένο video και είναι εν γένει διαφορετικό για διαφορετικά video. Συγκεκριμένα, το μέγεθος του Chunk μετρούμενο σε units για ένα ζητούμενο video i, εξαρτάται από το K_i, τον αριθμό των units του video i που ήδη βρίσκονται στην cache τη χρονική στιγμή που υποβάλλεται η αίτηση για το video i. Έτσι, το μέγεθος του Chunk(K_i) = g * K_i units, $K_i \neq 0$. Ο παράγοντας, g>0, αποκαλείται παράγοντας επιτάχυνσης (Acceleration Factor). Εάν το ζητούμενο video i λείπει από την cache, ($K_i = 0$), τότε αποθηκεύεται στην cache το πρώτο Chunk με μέγεθος 1 unit. Σκοπός αυτού του μηχανισμού είναι (α) να επιτρέψει στα δημοφιλή video να αποθηκεύσουν στην cache μεγάλο μέγεθος Chunk και συνεπώς να τους δώσει την δυνατότητα μεγαλύτερο μέρος τους να βρίσκεται στην cache και (β) να

δώσει την ευκαιρία video κατέστησαν δημοφιλή πρόσφατα και που απουσιάζουν από την cache να αυξήσουν την αποθηκευμένη μερίδα τους σε σύντομο χρόνο αυξάνοντας το μέγεθος των chunk που αποθηκεύουν σε κάθε νέο αίτημα για αυτά.

2.4 Τεχνικές Αντικατάστασης

Στην εργασία αυτή μελετήσαμε δύο τεχνικές απομάκρυνσης δεδομένων από την cache με σκοπό να αξιολογήσουμε τα αποτελέσματα που δίδουν οι δύο τεχνικές αποθήκευσης (FCS και VCS) όταν χρησιμοποιούνται σε συστήματα που υποστηρίζουν διαφορετικούς αλγόριθμους αντικατάστασης. Οι δύο αλγόριθμοι αντικατάστασης που θεωρήσαμε ακολουθούν παρακάτω.

2.4.1 Αλγόριθμος Least Recently Used

Ο αλγόριθμος Least Recently Used (LRU) είναι απλός και εύκολα υλοποιήσιμος. Ως κριτήριο για την επιλογή του video θύματος χρησιμοποιεί μόνο τις χρονικές στιγμές κατά τις οποίες κάθε video που βρίσκεται στην cache είχε την πιο πρόσφατη αναζήτηση από κάποιο χρήστη. Θύμα επιλέγεται εκείνο το video που δεν έχει ζητηθεί για μεγαλύτερο χρονικό διάστημα.

2.4.2 Αλγόριθμος Least Frequently Least Recently Used

Η πολυπλοκότητα του αλγόριθμου Least Frequently Least Recently Used (LFLRU) σε σχέση με τον LRU είναι αυξημένη δεδομένου ότι δεν χρησιμοποιεί μόνο ένα αλλά δύο κριτήρια απόφασης για την επιλογή του video θύματος. Πρώτο κριτήριο είναι ο αριθμός των αιτήσεων που έχουν γίνει για τα διάφορα video που βρίσκονται στην cache. Δηλαδή, ο αλγόριθμος αυτός προσπαθεί να λάβει υπόψη του την συχνότητα με την οποία αναζητούνται τα διάφορα video, διατηρώντας τον αριθμό των αναζητήσεων που έχουν γίνει για κάθε ένα από αυτά. Αν ένα video απομακρυνθεί ολοκληρωτικά από την cache τότε ο αντίστοιχος αριθμός αναζητήσεων μηδενίζεται και όταν το video ξαναμπεί αργότερα στην cache η καταγραφή των αναζητήσεων του θα αρχίσει από το μηδέν. Δεύτερο κριτήριο είναι το κριτήριο που χρησιμοποιεί ο αλγόριθμος LRU. Video-θύμα επιλέγεται αυτό με τις λιγότερες αναζητήσεις ($1^{ο}$ κριτήριο). Αν περισσότερα από ένα video έχουν τον ίδιο αριθμό αναζητήσεων, τότε ως θύμα επιλέγεται μεταξύ τους αυτό που δεν έχει ζητηθεί για το μεγαλύτερο χρονικό διάστημα ($2^{ο}$ κριτήριο).

2.4.3 Περιορισμός Αντικατάστασης ενός Video

Τέλος, πρέπει να σημειώσουμε ότι, θεωρούμε ότι ένα video δεν μπορεί να απομακρυνθεί από την cache (ολικά ή μερικά) όταν θεωρείται "ενεργό"(active). Ένα video θεωρείται ενεργό για το χρονικό διάστημα που ο proxy μεταδίδει στο χρήστη, που αναζήτησε το video αυτό, το μέρος εκείνο του video που βρισκόταν αποθηκευμένο στην cache τη στιγμή της αίτησης του χρήστη.

2.5 Μετρικές απόδοσης συστήματος

Για την αξιολόγηση της συμπεριφοράς μιας Video Proxy Cache η οποία λειτουργεί σε συνθήκες που καθορίζονται από την εφαρμογή των δύο τεχνικών αποθήκευσης, FCS και VCS, και των δύο αλγορίθμων αντικατάστασης, LRU και LFLRU, χρησιμοποιήσαμε δύο μετρικές απόδοσης. Το Byte Hit Ratio και το Delay Start. Οι μετρικές αυτές ορίζονται παρακάτω.

2.5.1 Μετρική Byte Hit Ratio

Όπως προαναφέρθηκε, η κύρια ευθύνη του Cache Manager είναι να εκμεταλλευτεί αποτελεσματικά τους πόρους αποθήκευσης του proxy ώστε να μειωθεί ο όγκος των video αντικειμένων που προσκομίζονται από τους κεντρικούς υπολογιστές προέλευσης (origin servers). Το πόσο αποδοτικός είναι ένας proxy cache server σε αυτή την ευθύνη του φαίνεται από τη μετρική Byte Hit Ratio (BHR). Το BHR αντιστοιχεί στο μέσο ποσοστό των δεδομένων video που εξυπηρετούνται άμεσα από την τοπική proxy cache. Στα συστήματα όπου εφαρμόζεται η μερική αποθήκευση των video ένα cache hit είναι στις περισσότερες περιπτώσεις μερικό και αντιπροσωπεύει την ποσότητα των δεδομένων που βρέθηκαν στην cache του proxy. Συνεπώς, το Byte Hit Ratio για ένα αίτημα για το video i ορίζεται ως:

$$\mathrm{BHR_i} = \frac{\text{Μεγεθος της αποθηκευμενης ποσοτητας του ζητουμενου video i}}{\text{Μεγεθος του πληρους video i}}$$

Το BHR_i παίρνει τιμές μεταξύ 0 και 1. Μηδέν για μια πλήρη αποτυχία αναζήτησης (cache miss), και ένα για μια πλήρη επιτυχία (cache hit). Το μέσο BHR όλων των ζητούμενων video κατά τη διάρκεια ενός χρονικού διαστήματος x συμβολίζεται με ως BHR (x). Το διάστημα x μπορεί να καθορίζεται σε ώρες ή σε ένα αριθμό από αναζητήσεις. Το steady state BHR (ss_BHR) καθορίζεται από το όριο του BHR (x) καθώς το x τείνει στο άπειρο.

2.5.2 Μετρική Delay Start

Η μετρική Delay Start αντιπροσωπεύει το ποσοστό των αιτήσεων video από τους πελάτες για τις οποίες υπήρχε cache miss.

$$\text{Delay Start} = \frac{\text{Αριθμος των αιτησεων που οδηγησαν σε cache miss σε ολη τη διαρκεια της προσομοιωσης}}{\text{Συνολικος αριθμος των αιτησεων στη διαρκεια της προσομοιωσης}}$$

Το Delay Start αντιστοιχεί στο ποσοστό των αιτήσεων πελατών που βιώνουν αρχική καθυστέρηση στην αναπαραγωγή του video, δεδομένου ότι ο proxy πρέπει να εγκαθιδρύσει σύνδεση με τον απομακρυσμένο εξυπηρετητή προέλευσης προτού αρχίσει να παραδίδει το video στον πελάτη.

2.6 Βέλτιστη πολιτική αποθήκευσης

2.6.1 High Popularity First

Αν υποθέσουμε ότι γνωρίζουμε εκ των προτέρων την κατανομή των πιθανοτήτων αναζήτησης των διαφόρων video από τους πελάτες, τότε μπορεί να αποδειχθεί [4] ότι η καλύτερη πολιτική που είναι δυνατό να εφαρμοστεί με στόχο την επίτευξη του μέγιστου ss_BHR είναι η High Popularity First (HPF). Σύμφωνα με την πολιτική αυτή, ο proxy αποθηκεύει στην cache ολόκληρα video ξεκινώντας από το πιο δημοφιλή μέχρι να γεμίσει η cache. Ανάλογα με το μέγεθος της cache, σε αυτήν μπαίνει εξ' ολοκλήρου ένας αριθμός από τα πιο δημοφιλή video. Μόνο το τελευταίο video που βρίσκεται στην cache (λιγότερο δημοφιλές από τα υπόλοιπα που βρίσκονται στην cache) μπορεί να μη βρίσκεται ολόκληρο σε αυτή.

Οι τιμές HPF_ss_BHR και HPF_Delay Start που προκύπτουν για τα διάφορα σενάρια προσομοίωσης που εξετάζονται στη διπλωματική αυτή βρίσκονται στο Παράρτημα Α και θα χρησιμοποιηθούν ως μέτρο σύγκρισης στα επόμενα κεφάλαια.

2.6.2 ss_BHR vs Delay Start

Στο σημείο αυτό θα πρέπει να κάνουμε μία παρατήρηση σχετικά με τις μετρικές ss_BHR και Delay Start, για την καλύτερη κατανόηση της αλληλεπίδρασης που έχουν μεταξύ τους. Αν εξετάσουμε τις δύο μετρικές ανεξάρτητα από τις τεχνικές αποθήκευσης και αντικατάστασης τότε όσο μεγαλύτερο ss_BHR πετύχουμε, κάτι που είναι επιθυμητό από πλευράς αποδοτικής χρησιμοποίησης του δικτύου, τόσο μεγαλύτερο Delay Start θα έχουμε (κάτι που είναι ανεπιθύμητο από την πλευρά των πελατών). Για παράδειγμα, αν μπορέσουμε γεμίσουμε την cache με τα πιο δημοφιλή ολόκληρα video, τότε θα πετύχουμε το υψηλότερο δυνατό ss_BHR αλλά παράλληλα ένα δεδομένο Delay Start. Εναλλακτικά, αν μπορέσουμε να γεμίσουμε την cache με τον διπλάσιο αριθμό από video, όπου το καθένα θα αποτελείται από το μισό μέγεθος του κάθε video τότε θα πετύχουμε χειρότερο ss_BHR από πριν αλλά καλύτερο Delay Start διότι στην cache υπάρχει μεγαλύτερος αριθμός από video, το καθένα όμως με λιγότερα units από πριν.

2.7 Γενικό μοντέλο προσομοίωσης

Υποθέτουμε ότι η κατανομή των αναζητήσεων των video είναι μια κατανομή Zipf. Τυπικά, ο τρόπος αναζήτησης των video σε ένα proxy server παρουσιάζει χρονική τοπικότητα. Εντούτοις, λόγω της έλλειψης πληροφοριών πρόσβασης από πραγματικούς video servers [7, 11], θεωρούμε ότι ισχύει το Independent Reference (IR) μοντέλο. Σύμφωνα με το μοντέλο αυτό, το video i ζητείται με την πιθανότητα p_i ανεξάρτητα από τα προηγούμενα αιτήματα. Το γεγονός ότι διάφορα αποτελέσματα που ελήφθησαν κατόπιν προσομοιώσεων με πραγματικά δεδομένα κίνησης (traces) αποδείχθησαν να είναι ποιοτικά παρόμοια με εκείνα που λαμβάνονται με την προσομοίωση που χρησιμοποιεί το μοντέλο IR [10], δείχνει ότι το μοντέλο IR είναι επαρκές για να προβλέψει τη σχετική απόδοση των προτεινόμενων τεχνικών αποθήκευσης.

Το μοντέλο του συστήματος που προσομοώνεται αποτελείται από ένα κεντρικό video server με N video και ένα proxy server ο οποίος έχει την δυνατότητα να αποθηκεύσει K ολόκληρα video. Η αναλογία K/N αντιπροσωπεύει την σχετική χωρητικότητα του proxy ως προς τον συνολικό αριθμό των video (Relative Cache Size). Για απλότητα υποθέτουμε ότι όλα τα video έχουν την ίδια σταθερή διάρκεια (μέγεθος ίσο με L units). Η μεταβλητή p_i δείχνει την πιθανότητα με την οποία

αναζητείται το video i. Η δημοτικότητα των video ακολουθεί κατανομή Zipf. Δηλαδή η πιθανότητα αναζήτησης για το video i δίδεται από τον τύπο $p_i = C/i^a$, όπου $C = \left(\sum_{i=1}^{N} \frac{1}{i^a} \right)^{-1}$ και το a είναι η παράμετρος της κατανομής Zipf που καθορίζει την πολικότητα της κατανομής. Τέλος, υποθέτουμε ότι οι αφίξεις αιτήσεων ακολουθούν μια διαδικασία Poisson με το μέσο ρυθμό αφίξεων λ. Οι παράμετροι του μοντέλου που προσομοιώνεται παρουσιάζονται στον πίνακα 2.1.

Συμβολισμός	**Επεξήγηση**	**Τιμή**
N	Αριθμός των video που βρίσκονται αποθηκευμένα στον κεντρικό video server	1000
L	Μέγεθος video/διάρκεια	1000units/hour
λ	Ρυθμός αιτήσεων	30req/hour
K	Μέγεθος cache	Μεταβλητή
K/N	Relative Cache Size	Μεταβλητή
a	Παράμετρος κατανομής Zipf	Μεταβλητή

Πίνακας 2.1

Οι προσομοιώσεις υλοποιήθηκαν σε περιβάλλον UNIX (Linux Mandrake 9.0) και γλώσσα προγραμματισμού C++ (με compiler g++).

ΚΕΦΑΛΑΙΟ 3
Σύστημα με Στατική Κατανομή Πιθανοτήτων Αναζήτησης

3.1 Πληροφορίες για το σύστημα προσομοίωσης

Στο κεφάλαιο αυτό παρουσιάζουμε τα αποτελέσματα προσομοίωσης ενός συστήματος όπου η κατανομή των πιθανοτήτων σύμφωνα με τις οποίες ζητούνται τα video που λαμβάνουν μέρος στην προσομοίωση παραμένει στατική καθ' όλη τη διάρκεια της. Δηλαδή οι πιθανότητες εμφάνισης των video υπολογίζονται πριν αρχίσει η προσομοίωση σύμφωνα με μια κατανομή Zipf και παραμένουν αμετάβλητες μέχρι το τέλος της προσομοίωσης. Όπως αναφέρθηκε και στο προηγούμενο κεφάλαιο ο συνολικός αριθμός των video είναι 1000 και έχουν όλα το ίδιο μέγεθος (1000 units/hour). Οι αφίξεις των αιτήσεων των πελατών στο σύστημα ακολουθεί Poisson κατανομή με ρυθμό λ ίσο με 30 αιτήσεις ανά ώρα. Τέλος, ο χρόνος προσομοίωσης είναι 10 000 ώρες.

Αρχίζοντας τη μελέτη αυτή θέλαμε πρώτα να παρατηρήσουμε τη συμπεριφορά του συστήματος όταν αυτό λειτουργεί κάτω από δύσκολες συνθήκες και φτάνει στα όρια λειτουργίας του. Με σκοπό να πετύχουμε αυτό το στόχο ασχοληθήκαμε με τη μεταβολή του μεγέθους της cache σε σχέση με τον συνολικό αριθμό των video (Relative Cache Size) και τη μεταβολή της παραμέτρου a της Zipf κατανομής. Συγκεκριμένα, μελετήσαμε το σύστημα όταν το μέγεθος της cache είναι πολύ μικρό (μικρό RCS) και όταν οι πιθανότητες αναζήτησης των video είναι πολύ πολωμένες (μεγάλο a).

Όλα τα σενάρια που έχουν προσομοιωθεί παρουσιάζονται στον παρακάτω πίνακα :

Τεχνική Αποθήκευσης	Relative Cache Size	Zipf Parameter
FCS	10%	0,8
FCS	10%	1,17
FCS	2%	0,8
FCS	2%	1,17
FCS	0,3%	0,8
FCS	0,3%	1,17
VCS	10%	0,8
VCS	10%	1,17
VCS	2%	0,8
VCS	2%	1,17
VCS	0,3%	0,8
VCS	0,3%	1,17

Πίνακας Σεναρίων Προσομοίωσης

Τα παραπάνω σενάρια έχουν εξεταστεί δύο φορές. Την πρώτη φορά προσομοιώθηκαν εφαρμόζοντας ως τεχνική αντικατάστασης τον αλγόριθμο Least Recently Used ενώ την δεύτερη φορά εφαρμόζοντας τον αλγόριθμο Least Frequently Least Recently Used. Η επίδραση των αλγορίθμων αντικατάστασης στις μετρικές απόδοσης του συστήματος παρουσιάζεται στα τμήματα του κεφαλαίου αυτού που ακολουθούν.

Τέλος, στον παρακάτω πίνακα παρουσιάζονται οι πιθανότητες αναζήτησης των είκοσι πιο δημοφιλών video όταν η κατανομή Zipf δεν είναι έντονα πολωμένη (a=0,8) και όταν είναι έντονα πολωμένη (a=1,17), αντίστοιχα.

Video	**Πιθανότητα Αναζήτησης του video όταν a=0,8**	**Πιθανότητα Αναζήτησης του video όταν a=1,17**
1	0.064642	0.214862
2	0.037127	0.095489
3	0.026842	0.059419
4	0.021324	0.042437
5	0.017838	0.032686
6	0.015417	0.026407
7	0.013628	0.022049
8	0.012247	0.018860
9	0.011146	0.016432
10	0.010245	0.014526
11	0.009493	0.012994
12	0.008855	0.011736
13	0.008305	0.010687
14	0.007827	0.009799
15	0.007407	0.009039
16	0.007034	0.008382
17	0.006701	0.007808
18	0.006402	0.007303
19	0.006131	0.006855
20	0.005884	0.006456
Άθροισμα των πιθανοτήτων αναζήτησης των υπόλοιπων video	0,695505	0,365774

Πίνακας Πιθανοτήτων Αναζήτησης

3.2 Μελέτη συστήματος με Fixed Chunk Size

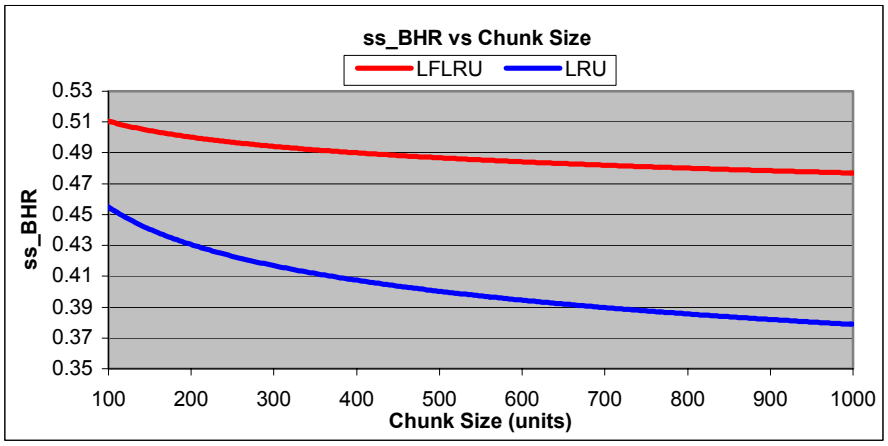
Για την μελέτη του συστήματος στο οποίο εφαρμόζεται η τεχνική αποθήκευσης Fixed Chunk Size προσομοιώσαμε το σύστημα για διαφορετικά μεγέθη Chunk Size. Σε κάθε προσομοίωση υπολογίσαμε το steady state Byte Hit Ratio (ss_BHR) και το αντίστοιχο Delay Start.

3.2.1 Αποτελέσματα μετρικής ss_BHR

Σενάριο Ι : Ξεκινάμε εξετάζοντας την συμπεριφορά του συστήματος όταν το **RCS είναι 10%** και η παράμετρος **a (Zipf) είναι 0,8**. Το σενάριο αυτό χαρακτηρίζεται από σχετικά ικανοποιητικό μέγεθος cache σε συνδυασμό με όχι έντονα πολωμένη κατανομή πιθανοτήτων αναζήτησης video. Τα αποτελέσματα φαίνονται παρακάτω.

Chunk Size(units)	LRU ss_BHR	LFLRU ss_BHR
100	0,451374	0,512805
200	0,431164	0,499162
300	0,419495	0,49274
400	0,409946	0,500053
500	0,40181	0,475932
600	0,395347	0,485734
700	0,389383	0,474528
800	0,385644	0,469341
900	0,378351	0,486426
1000	0,378281	0,48688

Πίνακας 3.1



Γραφική Παράσταση 3.1

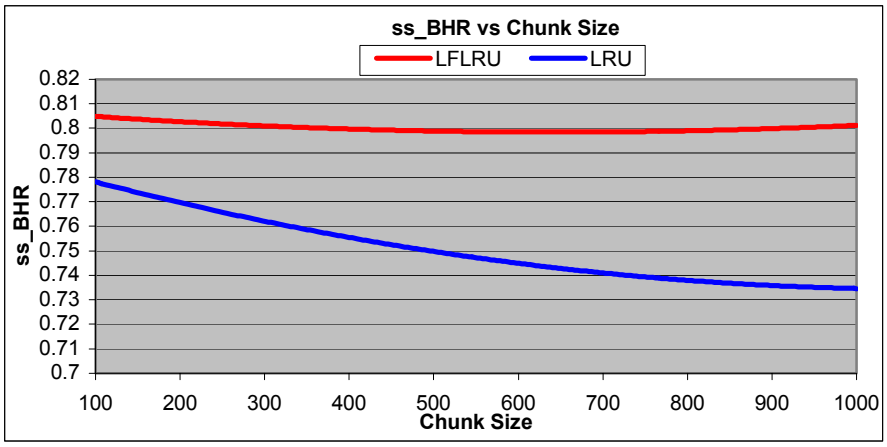
Είναι εμφανής η υπεροχή που παρουσιάζει ο αλγόριθμος αντικατάστασης LFLRU έναντι του αλγορίθμου LRU. Η ποσοστιαία βελτίωση που παρουσιάζει το ss_BHR όταν χρησιμοποιείται ο αλγόριθμος LFLRU αρχίζει από περίπου 13% για chunk size 100 units και καταλήγει σε περίπου 28% για μεγάλα μεγέθη chunk. Αναλύοντας τα αποτελέσματα των προσομοιώσεων για όλα τα μεγέθη Chunk καταλήγουμε σε μια μέση τιμή βελτίωσης περίπου 18%. Είναι σημαντικό να παρατηρήσουμε ότι η μέγιστη τιμή ss_BHR που μπορούμε να πετύχουμε σε αυτό το σενάριο, αν θεωρήσουμε ότι γνωρίζουμε ποιες είναι οι πιθανότητες αναζήτησης και τοποθετήσουμε στην cache ολόκληρα τα πιο δημοφιλή video είναι 0,525 (HPF_ss_BHR). Ο LFLRU χωρίς να γνωρίζει τις πιθανότητες αναζήτησης πετυχαίνει ss_BHR με τιμές μεταξύ 0,51 και 0,47.

Ο αλγόριθμος LFLRU κτίζει σταδιακά την «ιστορία» κάθε video που βρίσκεται στην cache διατηρώντας τον αριθμό των αναζητήσεων που έγιναν για το κάθε ένα από αυτά, και έτσι καταφέρνει να αποκτήσει μια καλή εκτίμηση της δημοτικότητας τους. Χρησιμοποιώντας το πλεονέκτημα αυτό αποθηκεύει στην cache τα πιο δημοφιλή από αυτά. Η αδυναμία που παρουσιάζει ο αλγόριθμος LRU, λόγω έλλειψης πληροφορίας για τις αναζητήσεις που έχουν γίνει στο παρελθόν για το κάθε video που είναι στην cache, γίνεται πιο εμφανής καθώς το μέγεθος του chunk αυξάνεται και κατά συνέπεια η κατάσταση της cache αλλάζει με πιο έντονους ρυθμούς. Συγκεκριμένα με τον LRU η ποσότητα σε units που καταλαμβάνει μετά από μια αναζήτηση ένα μη δημοφιλές video είναι μεγαλύτερη από πριν, κάτι που σημαίνει ότι αντίστοιχη ποσότητα από ένα πιο δημοφιλές video θα απομακρυνθεί αναγκαστικά από την cache.

Σενάριο ΙΙ : Στο σενάριο αυτό διατηρούμε το μέγεθος της cache ίδιο με πριν (δηλ. **RCS=10%**) αλλά πολώνουμε περισσότερο την κατανομή πιθανοτήτων αναζήτησης αυξάνοντας την παράμετρο της Zipf και θέτοντας την ίση με 1,17 (δηλ. **a=1,17**). Με αυτό τον τρόπο πετυχαίνουμε οι πιθανότητες αναζήτησης των πιο δημοφιλών video να είναι πολύ μεγαλύτερες από αυτές των λιγότερο δημοφιλών.

Chunk Size(units)	LRU ss_BHR	LFLRU ss_BHR
100	0,779708	0,806404
200	0,769264	0,801671
300	0,761468	0,798861
400	0,752434	0,801842
500	0,749931	0,797906
600	0,745685	0,795836
700	0,742902	0,803089
800	0,739205	0,797601
900	0,73566	0,80101
1000	0,73333	0,800014

Πίνακας 3.2



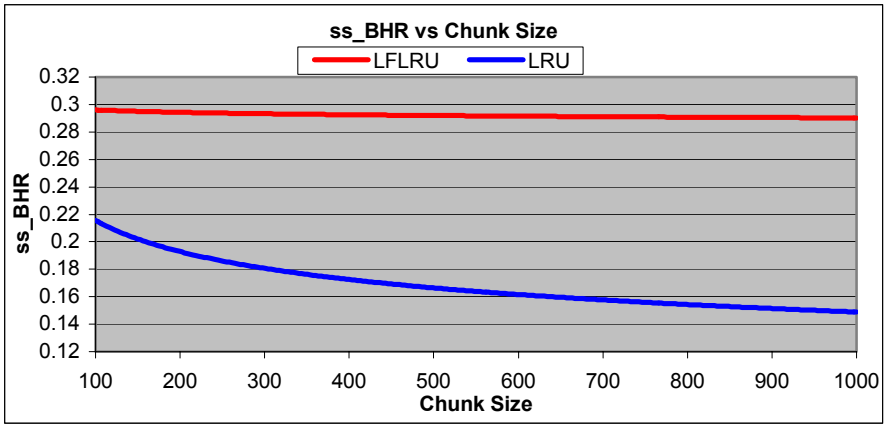
Γραφική Παράσταση 3.2

Από τα αποτελέσματα στον πίνακα 3.2 και στη γραφική παράσταση 3.2 φαίνεται ότι ο αλγόριθμος LFLRU δίνει καλύτερο ss_BHR. Η ποσοστιαία βελτίωση που παρουσιάζει σε σχέση με τον αλγόριθμο LRU είναι περίπου 3,5% για πολύ μικρό μέγεθος chunk ενώ για μεγαλύτερο μέγεθος chunk η βελτίωση είναι περίπου 9%. Είναι φανερό ότι έστω και καλύτερος από τον LRU η βελτίωση που πετυχαίνει ο LFLRU έχει μειωθεί συγκρινόμενη με τη βελτίωση που πετυχαίνει σε ένα σύστημα όπου οι πιθανότητες αναζήτησης των video δεν είναι πολωμένες. Το φαινόμενο αυτό οφείλεται στο γεγονός ότι ο αλγόριθμος LRU επωφελείται από την ύπαρξη της πόλωσης στην κατανομή αναζήτησης. Λόγω της πόλωσης η ομάδα των video που ζητείται πιο συχνά είναι μικρότερη και κατά συνέπεια παρουσιάζονται λιγότερο έντονες αλλαγές στην κατάσταση της cache. Επιπρόσθετα, πρέπει να επισημάνουμε ότι η δυνατότητα του LFLRU να βελτιώσει τις τιμές του ss_BHR είναι περιορισμένη αφού το μέγιστο ss_BHR που μπορεί να επιτευχθεί είναι περίπου 0,813(HPF_ss_BHR).

Σενάριο ΙΙΙ : Θέλοντας να πιέσουμε το σύστημα μειώσαμε την χωρητικότητα της cache θέτοντας **RCS=2%**. Αυτό σημαίνει ότι στην cache του proxy μπορούν να αποθηκευτούν μόνο 20 ολόκληρα video από τα 1000 video που βρίσκονται στον κεντρικό video server. Παράλληλα θέτουμε την παράμετρο της κατανομής Zipf ίση με 0,8 (**a=0,8**).

Chunk Size(units)	LRU ss_BHR	LFLRU ss_BHR
100	0,217092	0,297958
200	0,194373	0,296058
300	0,181664	0,290157
400	0,170387	0,291182
500	0,164377	0,28776
600	0,157731	0,295668
700	0,156006	0,286178
800	0,153985	0,292332
900	0,153615	0,292001
1000	0,153014	0,293105

Πίνακας 3.3



Γραφική Παράσταση 3.3

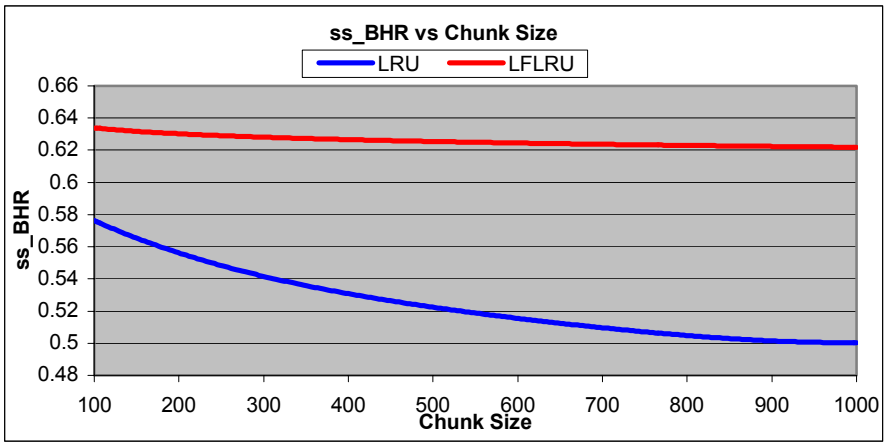
Ελαττώνοντας αρκετά την δυνατότητα της cache να αποθηκεύει κλάσμα του συνολικού περιεχομένου του κεντρικού video server (μόνο 2%) διαπιστώνουμε ότι ο αλγόριθμος LFLRU καταφέρνει να ανταποκριθεί πολύ καλά στη δοκιμασία αυτή κάτι που δεν ισχύει για τον αλγόριθμο LRU. Η παρατήρηση αυτή επιβεβαιώνεται υπολογίζοντας τις ποσοστιαίες βελτιώσεις που πετυχαίνει ο LFLRU σε σύγκριση με τις επιδόσεις του LRU. Οι τιμές που παρουσιάζουν οι βελτιώσεις αυτές αρχίζουν περίπου από 30% για μικρό Chunk μέγεθος (100 units) και αγγίζουν την τιμή 90% καθώς το μέγεθος του chunk που χρησιμοποιείται στο σύστημα αυξάνεται. Η μέση ποσοστιαία αύξηση που προσδίδει ο αλγόριθμος LFLRU στο ss_BHR υπολογίζεται σε 62%.

Αναμφισβήτητα, τα αποτελέσματα της προσομοίωσης δείχνουν την σαφή υπεροχή του αλγορίθμου LFLRU σε σχέση με τον αλγόριθμο LRU. Ο LFLRU χρησιμοποιώντας τις αναζητήσεις των video καταφέρνει να οργανώσει καλύτερα το περιεχόμενο της μικρής cache του proxy. Η τεχνική αντικατάστασης LRU αντιμετωπίζει με μεγαλύτερη δυσκολία το γεγονός ότι η cache είναι μικρή, γι' αυτό και για σχετικά μεγάλα chunks επιτυγχάνει πολύ μικρό ss_BHR. Αντίθετα ο αλγόριθμος LFLRU παρουσιάζει μια σχετική σταθερότητα για όλα τα μεγέθη chunk και καταφέρνει να πλησιάσει κοντά στη μέγιστη τιμή ss_BHR που μπορεί να επιτευχθεί (HPF_ss_BHR=0,303).

Σενάριο IV : Στην συνέχεια εξετάζουμε την συμπεριφορά της cache όταν θέσουμε **RCS=2%** και **a=1,17**. Στην περίπτωση αυτή με το ίδιο μικρό μέγεθος της cache που είχαμε στο προηγούμενο σενάριο συνδυάζουμε μια πολωμένη κατανομή πιθανοτήτων αναζήτησης. Τα αποτελέσματα που παίρνουμε φαίνονται παρακάτω.

Chunk Size(units)	LRU ss_BHR	LFLRU ss_BHR
100	0,576172	0,633531
200	0,556439	0,631702
300	0,541361	0,628352
400	0,530269	0,624534
500	0,523511	0,627437
600	0,515268	0,621957
700	0,509103	0,622493
800	0,504857	0,623487
900	0,501817	0,624112
1000	0,500302	0,621938

Πίνακας 3.4



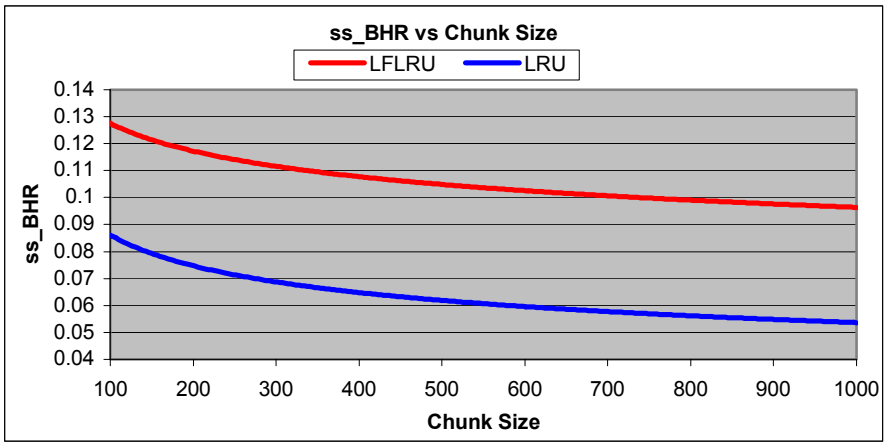
Γραφική Παράσταση 3.4

Και σ'αυτό το σενάριο ο αλγόριθμος LFLRU υπερέχει σε σχέση με το αλγόριθμο LRU. Η ποσοστιαία αύξηση που προσδίδει ο LFLRU στο ss_BHR αρχίζει από το 10% για chunk ίσο με 100 units και φτάνει το 24% για μεγάλα chunk. Η μείωση στην ποσοστιαία βελτίωση που παρατηρείται σε σχέση με το προηγούμενο σενάριο οφείλεται στο γεγονός ότι τα video που ζητούνται συχνότερα είναι λιγότερα λόγω της πόλωσης που χαρακτηρίζει την κατανομή των πιθανοτήτων αναζήτησης, κάτι που βοηθά τον αλγόριθμο LRU να διατηρήσει στην cache περισσότερα δημοφιλή video. Αντίθετα ο LFLRU καταφέρνει να εκμεταλλεύεται ιδανικά το χώρο της cache πετυχαίνοντας τιμές που είναι πολύ κοντά στο HPF_ss_BHR (0,634).

Σενάριο V : Μία ακραία κατάσταση που μπορεί να βρεθεί το εξεταζόμενο σύστημα είναι η ακόλουθη. Η cache να χωράει μερικά μόνο ολόκληρα video (π.χ.,τρία) και η κατανομή των πιθανοτήτων αναζήτησης να μην είναι έντονα πολωμένη. Έτσι, θέτοντας **RCS=0,3%** και **a=0,8** πήραμε τα παρακάτω αποτελέσματα ss_BHR για διάφορα μεγέθη chunk.

Chunk Size(units)	LRU ss_BHR	LFLRU ss_BHR
100	0.083452	0.124175
200	0.076415	0.119753
300	0.069587	0.115352
400	0.066422	0.106803
500	0.061236	0.103558
600	0.059785	0.10339
700	0.058324	0.101279
800	0.057596	0.098184
900	0.053574	0.097308
1000	0.051997	0.095475

Πίνακας 3.5



Γραφική Παράσταση 3.5

Αν και οι τιμές του ss_BHR που επιτυγχάνονται σε αυτό το σενάριο είναι οι πιο μικρές συγκρινόμενες με όλα τα υπόλοιπα σενάρια, οι διαφορές που παρατηρούνται μεταξύ των επιδόσεων των δύο αλγορίθμων αντικατάστασης είναι αρκετά έντονες. Ο αλγόριθμος LFLRU καταφέρνει να βελτιώσει τις τιμές ss_BHR που παράγει ο LRU κατά 48% περίπου όταν το chunk έχει μέγεθος 100 units και καθώς αυξάνεται το chunk size η ποσοστιαία αυτή αύξηση φτάνει το 83%. Οι πολύ χαμηλές τιμές του επιτυχανόμενου ss_BHR οφείλονται στον υπερβολικά μικρό χώρο που προσφέρει η cache για αποθήκευση video δεδομένων. Παρόλα αυτά, η τεχνική αντικατάστασης LFLRU καταφέρνει να δώσει έντονα καλύτερα αποτελέσματα από την τεχνική LRU.

Σενάριο VI : Διατηρώντας το μέγεθος της cache το ίδιο με το προηγούμενο σενάριο (**RCS=0.3%**) θέσαμε την παράμετρο της κατανομής Zipf ίση με 1,17 (**a=1,17**). Τα αποτελέσματα που πήραμε δεν παρουσιάζονται για λόγους εξοικονόμησης αλλά διέπονται από την ίδια λογική με τα προηγούμενα. Ο αλγόριθμος LFLRU υπερέχει έναντι του αλγορίθμου LRU για όλα τα μεγέθη chunk προσδίδοντας στο σύστημα μια μέση βελτίωση της τάξης του 6,5%. Η μέγιστη και ελάχιστη τιμή ss_BHR που δίνει ο LFLRU είναι 0,363 και 0,338 αντίστοιχα ενώ το ιδανικό ss_BHR που μπορεί να επιτευχθεί υπό αυτές τις συνθήκες είναι περίπου 0,37.

3.2.2 Αποτελέσματα μετρικής Delay Start

Ως δεύτερη μετρική αξιολόγησης της συμπεριφοράς του συστήματος χρησιμοποιήσαμε το Delay Start θέλοντας να αποκτήσουμε σαφή εικόνα για το ποσοστό των πελατών που υπόκεινται σε αρχική καθυστέρηση αφού υποβάλουν το

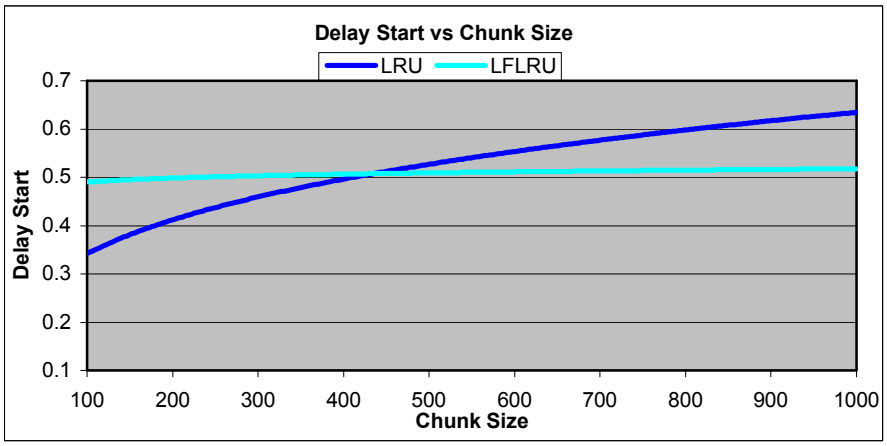
αίτημά τους στον proxy. Επιθυμητό είναι να έχουμε όσο το δυνατό μικρότερο Delay Start.

Κατά τη διάρκεια της μελέτης μας υπολογίσαμε την μετρική Delay Start για όλα τα σενάρια που παρουσιάστηκαν πιο πριν. Παρακάτω παρουσιάζουμε αναλυτικά μόνο μερικά αντιπροσωπευτικά σενάρια ενώ στα αποτελέσματα των υπολοίπων θα αναφερόμαστε σε συντομία.

Σενάριο Ι : Αρχίζουμε την μελέτη του Delay Start παρουσιάζοντας τα αποτελέσματα του συστήματος με συνθήκες προσομοίωσης **RCS=10%** και **a=0,8**.

Chunk Size(units)	LRU Delay Start	LFLRU Delay Start
100	0.330743	0.48618
200	0.417648	0.498055
300	0.467549	0.501538
400	0.506697	0.513213
500	0.538619	0.509067
600	0.55622	0.522444
700	0.575528	0.516787
800	0.591578	0.515149
900	0.607526	0.505921
1000	0.623136	0.511779

Πίνακας 3.6



Γραφική Παράσταση 3.6

Παρατηρούμε ότι για μικρά μεγέθη chunk το Delay Start του αλγορίθμου LFLRU είναι μεγαλύτερο από αυτό που παρουσιάζει ο LRU. Καθώς όμως τα μεγέθη των chunks αυξάνονται, αρχίζει να αυξάνεται και το Delay Start της τεχνικής αντικατάστασης LRU. Η αύξηση αυτή εξελίσσεται με γοργό ρυθμό έχοντας ώς αποτέλεσμα να ξεπεράσει, από κάποιο μέγεθος chunk και μετά, το Delay Start του αλγόριθμου LFLRU. Αντίθετα ο αλγόριθμος LFLRU παρουσιάζει μια σχετική σταθερότητα στις τιμές του Delay Start. Ενώ αρχικά, ο LFLRU παρουσιάζει τιμές της

μετρικής κατά 47% μεγαλύτερες από αυτές του LRU, για μέγεθος chunk 100 units, καταφέρνει καθώς μεγαλώνουν τα μεγέθη chunk να παρουσιάσει τιμές μετρικής μέχρι και 18% μικρότερες από αυτές του LRU.

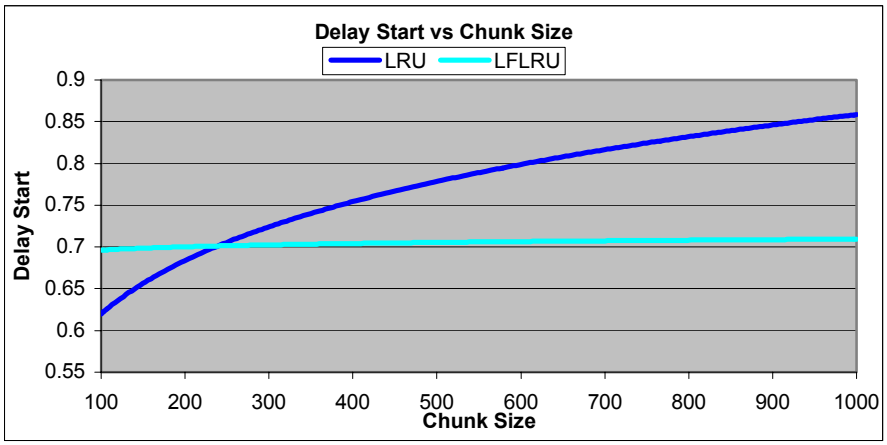
Ο λόγος στον οποίο οφείλεται το καλύτερο Delay Start που παρουσιάζει ο αλγόριθμος LRU για μικρά chunk είναι ο εξής : η οργάνωση των δεδομένων της cache ενός proxy που χρησιμοποιεί LRU δεν γίνεται με τόσο καλό τρόπο όσο αυτού που χρησιμοποιεί LFLRU αφού ο δεύτερος λαμβάνει υπόψη και το αριθμό τον αναζητήσεων και την χρονική στιγμή που γίνεται η πιο πρόσφατη αναζήτηση για κάθε video. Ο LFLRU καταφέρνει να εκτιμήσει με μεγαλύτερη επιτυχία πια είναι τα πιο δημοφιλή video και να τα κρατήσει στην cache. Κατά συνέπεια ο LFLRU διατηρεί μικρότερο αριθμό video στην cache που το καθένα αποτελείται από περισσότερα units. Αντίθετα ο LRU διατηρεί περισσότερα video, αναγκαστικά με λιγότερα chunk το καθένα.

Το φαινόμενο που μόλις περιγράφηκε πιο πάνω γίνεται πολύ πιο έντονο όταν το μέγεθος του chunk είναι μικρό-π.χ. 10 units. Με τόσο μικρό chunk ο LRU επιτρέπει την παραμονή στην cache ενός πολύ μεγάλου αριθμού από video πολλά από τα οποία απαρτίζονται από πολύ λίγα units ενώ LFLRU αναγνωρίζει τα πιο δημοφιλή (λίγα) και παραχωρεί στο καθένα από αυτά μεγάλη ποσότητα χώρου (πολλά units).

Στο σημείο αυτό πρέπει επίσης να παρατηρήσουμε ότι το Delay Start που δίνει ο LFLRU προσεγγίζει την τιμή HPF_Delay Start (0,47) που είναι η ελάχιστη τιμή που μπορούμε να πετύχουμε αν επιθυμούμε μέγιστο ss_BHR.

Θέτοντας το **a=1,17**, γενικά οι τιμές του Delay Start μειώνονται περίπου κατά 50% ενώ τα συμπεράσματα της σύγκρισης μεταξύ των δύο αλγορίθμων αντικατάστασης παραμένουν ουσιαστικά τα ίδια.

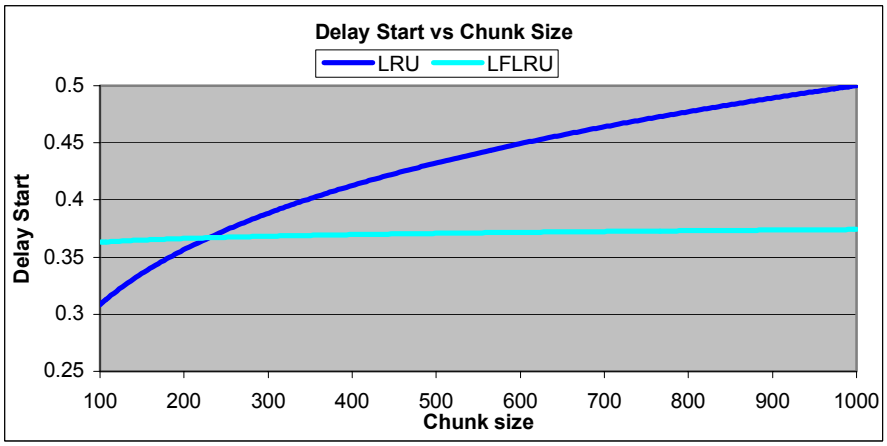
Σενάριο ΙΙ : Στην συνέχεια παρουσιάζουμε τις τιμές που παίρνει το Delay Start στην περίπτωση **RCS=2%** και **a=0.8**. Οι συνθήκες προσομοίωσης του συστήματος αυτή τη φορά είναι πιο δύσκολες από το σενάριο Ι αφού στην cache μπορούν να αποθηκευτούν μόνο 20 ολόκληρα video.



Γραφική Παράσταση 3.7

Από τα αποτελέσματα του γραφήματος 3.7 παρατηρούμε ότι ο αλγόριθμος αντικατάστασης LRU δίνει καλύτερα αποτελέσματα στο διάστημα από 100 μέχρι περίπου 300 units ενώ στο υπόλοιπο διάστημα η αυξητική τάση που παρουσιάζει το Delay Start όταν εφαρμόζεται ο LRU αποβαίνει μοιραία για το σύστημα. Έτσι ο αλγόριθμος LFLRU επιδεικνύοντας μια πιο σταθερή συμπεριφορά καταφέρνει να πετύχει, για μεγέθη μεγαλύτερα των 300 units, καλύτερο Delay Start από αυτό του LRU. Η ποσοστιαία βελτίωση που πετυχαίνει ο LFLRU σε σχέση με τον LRU φτάνει το 16%.

Σενάριο ΙΙΙ : Στο τελευταίο σενάριο που παραθέτουμε αναλυτικά εξετάζουμε το σύστημα όταν η cache είναι μικρή και η κατανομή των πιθανοτήτων αναζήτησης πολωμένη. Έτσι στην περίπτωση αυτή έχουμε **RCS=2%** και **a=1,17**.



Γραφική Παράσταση 3.8

Παρατηρούμε ότι και αυτή τη φορά η συμπεριφορά των δύο τεχνικών αντικατάστασης είναι η ίδια με τα προηγούμενα σενάρια. Ο LRU παρουσιάζει μια έντονη αυξητική τάση στις τιμές του Delay Start καθώς το μέγεθος του chunk μεγαλώνει. Αντίθετα, η συμπεριφορά του LFLRU προσδίδει μια σχετικά σταθερή τιμή στο Delay Start για όλα τα μεγέθη chunk. Επίσης διαπιστώνουμε ότι στο σενάριο αυτό, συγκρινόμενο με το σενάριο ΙΙ όπου a=0,8, η ποσοστιαία μείωση του Delay Start για chunk μεγαλύτερα ή ίσα των 300 units που πετυχαίνει ο LFLRU είναι μεγαλύτερη, με μέγιστη τιμή μείωσης 24%.

Υπόλοιπα σενάρια : Για τα σενάρια όπου RCS=0.3% και a=0.8 ή 1,17 η συμπεριφορά των δύο αλγορίθμων παραμένει η ίδια με την διαφορά ότι η μέγιστη μείωση Delay Start που πετυχαίνει ο LFLRU είναι της τάξης του 3% με 5% κάτι που οφείλεται στην υπερβολικά μικρή χωρητικότητα της cache. Επίσης, η μέση τιμή της ποσοστιαίας προσέγγισης του Delay Start που πετυχαίνει ο LFLRU για a=0,8 είναι 0,88% ενώ για a=1,17 είναι 0,64%. Όπως και στα προηγούμενα σενάρια έτσι και σ'αυτό οι τιμές που πετυχαίνει ο LFLRU προσεγγίζουν το HPF_Delay Start.

3.2.3 Γενικά σχόλια

Το σύστημα που εξετάσαμε στην παράγραφο 3.2 υλοποιήθηκε εφαρμόζοντας την τεχνική αποθήκευσης Fixed Chunk Size και διατηρώντας την Zipf κατανομή πιθανοτήτων αναζήτησης των video σταθερή καθ' όλη την διάρκεια τις προσομοίωσης. Μεταβάλαμε το μέγεθος της cache και την τιμή της παραμέτρου Zipf με σκοπό να δούμε πια θα είναι η επίδραση που θα έχουν στην απόδοση του συστήματος οι δύο τεχνικές αντικατάστασης Least Recently Used και Least Frequently Least Recently Used. Ως μετρικές αξιολόγησης χρησιμοποιήσαμε το Byte Hit Ratio και το Delay Start.

Αν εξετάσουμε τους δύο αλγορίθμους αντικατάστασης ξεχωριστά, παρατηρούμε ότι ο LFLRU παρουσιάζει μια σταθερή συμπεριφορά για όλα τα σενάρια και όλα τα μεγέθη chunk τόσο στη μετρική ss_BHR όσο και στη μετρική Delay Start. Καταφέρνει να πετύχει σχεδόν το μέγιστο δυνατό ss_BHR και το αντίστοιχο ελάχιστο Delay Start που μπορεί να επιτευχθεί, αν γνωρίζουμε τις πιθανότητες αναζήτησης και αποθηκεύαμε στην cache ολόκληρα τα πιο δημοφιλή video.

Αντίθετα, ο LRU αντιμετωπίζει προβλήματα και στις δύο μετρικές. Σε όλα τα σενάρια καταφέρνει να πετύχει υψηλό ss_BHR και χαμηλό Delay Start μόνο όταν το μέγεθος του chunk είναι μικρό. Καθώς το μέγεθος του chunk μεγαλώνει, το ss_BHR πέφτει (πολλές φορές πολύ γρήγορα) ενώ το Delay Start αυξάνεται έντονα. Το φαινόμενο αυτό οφείλεται στο γεγονός ότι ο LRU δεν μπορεί να αξιολογήσει τη δημοτικότητα των video αφήνοντας στην cache video τμήματα που δεν θα έπρεπε. Κατά συνέπεια, για μικρά μεγέθη chunk έχει πολλά video στην cache και χαμηλό Delay Start. Η επίπτωση της λανθασμένης αυτής επιλογής του LRU δεν φαίνεται έντονα όταν η cache είναι μεγάλη, διότι σε αυτή την περίπτωση δίνεται η δυνατότητα να αποθηκεύονται πολλά units γι' αυτό και παρουσιάζεται σχετικά ψηλό ss_BHR. Αντίθετα, όταν η cache είναι μικρή, ο LRU δίνει χαμηλό ss_BHR. Επιπρόσθετα, με LRU για μεγάλα μεγέθη chunk το ss_BHR είναι μικρό και το Delay Start μεγάλο διότι στην cache δεν υπάρχει ούτε μεγάλος αριθμός video ούτε τα πιο δημοφιλή λόγω της αδυναμίας του LRU να τα αναγνωρίσει και να τα επιλέξει και λόγω του μεγάλου μεγέθους που έχουν τα chunks.

Παρατηρήσαμε ότι για όλα στα σενάρια που προσομοιώθηκαν ο αλγόριθμος LFLRU καταφέρνει να επιτύχει πολύ καλύτερο ss_BHR. Ειδικά για τα σενάρια όπου η κατανομή των πιθανοτήτων αναζήτησης δεν είναι έντονα πολωμένη (a=0,8) η ποσοστιαία βελτίωση που παρουσιάζει το ss_BHR όταν εφαρμόζεται ο LFLRU σε σχέση με την περίπτωση που εφαρμόζεται ο LRU είναι εντυπωσιακή. Στα σενάρια όπου η χωρητικότητα της cache είναι μικρή παρατηρείται βελτίωση που φτάνει το 80% και 90%. Επίσης, για σενάρια όπου η Zipf κατανομή πιθανοτήτων είναι πολωμένη, και πάλι η υπεροχή του LFLRU είναι αισθητή αφού πετυχαίνει ποσοστιαία βελτίωση του ss_BHR που φτάνει για RCS=10% και RCS=2% , το 9% και το 24%, αντίστοιχα. Η μείωση της ποσοστιαίας διαφοράς επίδοσης μεταξύ του LFLRU και του LRU δεν οφείλεται σε μη καλή λειτουργία του LFLRU αλλά στη βοήθεια που δέχεται ο LRU από την πόλωση της κατανομής αναζήτησης κάτι που σημαίνει ότι τα πιο δημοφιλή video είναι πολύ λιγότερα και ζητούνται πιο συχνά.

Όσο αφορά την μετρική Delay Start η κατάσταση είναι λίγο πιο περίπλοκη. Ο αλγόριθμος LRU παρουσιάζει καλύτερα αποτελέσματα για μικρά μεγέθη chunk ενώ καθώς τα μεγέθη chunk μεγαλώνουν το Delay Start που ο LRU προσδίδει στο σύστημα χειροτερεύει όλο και περισσότερο. Αντίθετα ο αλγόριθμος LFLRU παρουσιάζει μια σταθερή συμπεριφορά με αποτέλεσμα για μεγαλύτερα μεγέθη chunk να επιτυγχάνει καλύτερο Delay Start από τον LRU. Αυτό συμβαίνει για μέγεθος

chunk μεγαλύτερο των 400 units όταν ισχύει RCS=10% και για μέγεθος chunk μεγαλύτερο ή ίσο των 300 units όταν RCS=2%.

Κατά συνέπεια, συμπεραίνουμε ότι για τις πιο πάνω τιμές μεγέθους chunk ο αλγόριθμος αντικατάστασης Least Frequently Least Recently Used πετυχαίνει ταυτόχρονα και καλύτερο Byte Hit Ratio και καλύτερο Delay Start από τον αλγόριθμο αντικατάστασης Least Recently Used.

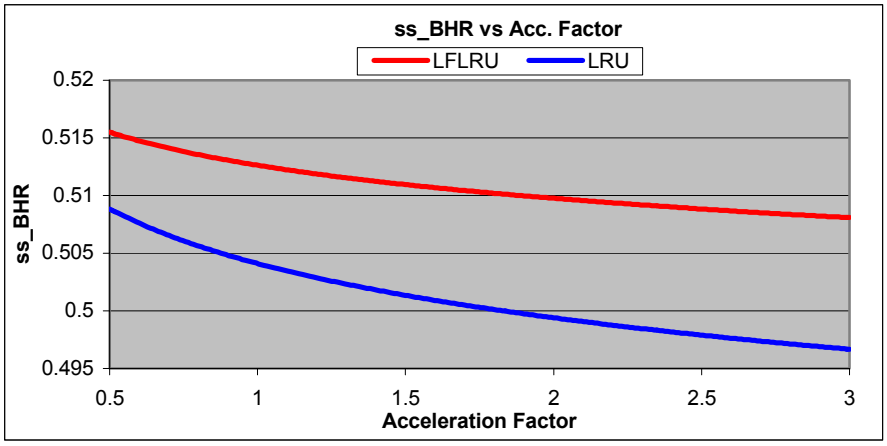
3.3 Μελέτη συστήματος με Variable Chunk Size

Για την αξιολόγηση ενός συστήματος στο οποίο εφαρμόζεται η τεχνική αποθήκευσης Variable Chunk Size πραγματοποιήσαμε διάφορες προσομοιώσεις μεταβάλλοντας την τιμή του Acceleration Factor "g". Ως μέτρο σύγκρισης χρησιμοποιήσαμε και πάλι τις μετρικές Byte Hit Ratio και Delay Start.

3.3.1 Αποτελέσματα μετρικής ss_BHR

Στο μέρος αυτό επιλέξαμε για λόγους οικονομίας χώρου να παρουσιάσουμε αναλυτικά μόνο δύο σενάρια, αρκετά για να αποτυπώσουν την εικόνα του συστήματος όταν αυτό χρησιμοποιεί VCS.

Σενάριο Ι : Στην περίπτωση αυτή, οι συνθήκες προσομοίωσης είναι **RCS=10%** και **a=0,8**.

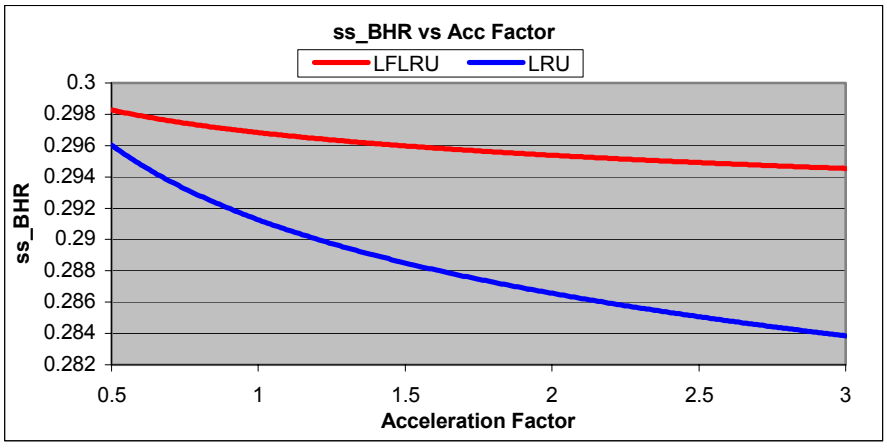


Γραφική Παράσταση 3.9

Παρατηρούμε ότι ο αλγόριθμος LFLRU καταφέρνει να επιτύχει καλύτερες τιμές ss_BHR για όλες τις τιμές του Acceleration Factor g που εξετάστηκαν. Οι ποσοστιαίες βελτιώσεις που εμφανίζει ο LFLRU έναντι του LRU δεν είναι μεγάλες αφού η μέγιστη παρατηρείται για το μεγαλύτερο g και είναι 2,3%. Οι μικρές διαφορές που υφίστανται οφείλονται στο γεγονός ότι η αδυναμία που παρουσιάζει ο αλγόριθμος LRU στον τομέα των κριτηρίων απόφασης για το πια video θα παραμείνουν στην cache, αντισταθμίζεται από τη δυνατότητα της τεχνικής αποθήκευσης VCS να τοποθετεί όλο και μεγαλύτερα chunk στην cache για τα video που δέχονται διαδοχικές αιτήσεις με αποτέλεσμα να καταφέρνει να αποκαθιστά την διαμόρφωση σωστών περιεχομένων στην cache. Δηλαδή τα πιο δημοφιλή video να καταλαμβάνουν μεγαλύτερη ποσότητα χώρου (units). Με απλά λόγια θα μπορούσαμε να πούμε ότι οι αδυναμίες του αλγόριθμου LRU διορθώνονται από την τεχνική VCS.

Η αλλαγή της τιμής της παραμέτρου Zipf σε 1.17 δίδει την ίδια εικόνα συμπεριφοράς του συστήματος, με τη μόνη διαφορά ότι σε αυτή την περίπτωση αντισταθμίζονται ακόμη περισσότερο οι συνέπειες των λανθασμένων επιλογών του LRU βοηθώντας τον έτσι να πετύχει ακόμη υψηλότερο ss_BHR.

Σενάριο ΙΙ : Τέλος παρουσιάζουμε τα αποτελέσματα της μετρικής ss_BHR όταν το μέγεθος της cache είναι πολύ μικρό και η κατανομή των πιθανοτήτων αναζήτησης δεν είναι έντονα πολωμένη (**RCS=2%** και **a=0.8**).



Γραφική Παράσταση 3.10

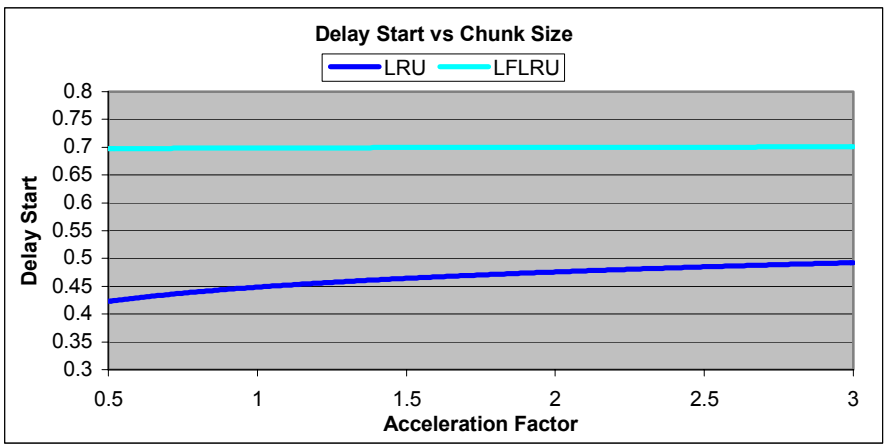
Είναι φανερό ότι ο αλγόριθμος LFLRU δίνει καλύτερο ss_BHR και στην περίπτωση όπου η cache είναι μικρή. Παρόλο που οι ποσοστιαίες διαφορές μεταξύ των τιμών του ss_BHR που παρουσιάζουν οι δύο αλγόριθμοι είναι μικρές

παρατηρούμε ότι σε αυτή την περίπτωση η βελτίωση που πετυχαίνει ο LFLRU φτάνει το 4,4%. Αυτό οφείλεται στο γεγονός ότι το μικρό μέγεθος της cache δυσκολεύει τη τεχνική VCS να αντισταθμίσει τα "σφάλματα" που δημιουργεί ο αλγόριθμος αντικατάστασης LRU λόγω έλλειψης πληροφορίας που διακρίνει τον τρόπο που αποφασίζει για την παραμονή ή απομάκρυνση τμήματος ενός video από την cache.

Επιπρόσθετα, όταν η cache είναι υπερβολικά μικρή οι τάσεις που παρουσιάζουν οι γραφικές παραστάσεις είναι οι ίδιες με τη μέγιστη ποσοστιαία βελτίωση της επίδοσης του LFLRU έναντι του LRU 6%. Όσο πιο μικρή είναι η cache, τόσο καλύτερος εμφανίζεται ο αλγόριθμος LFLRU σε σύγκριση με τον LRU. Τέλος, και για τα δύο μεγέθη μικρής cache που εξετάσαμε (RCS=2% και RCS=0.3%) θεωρώντας a=1,17, οι βελτιώσεις που παρατηρούνται είναι αντίστοιχα πιο μικρές από αυτές με a=0,8.

3.3.2 Αποτελέσματα μετρικής Delay Start

Μετά την προσομοίωση όλων των σεναρίων του πίνακα 3.1 με VCS παρατηρήσαμε ότι για όλα τα σενάρια ο αλγόριθμος αντικατάστασης LFLRU δίνει χειρότερα αποτελέσματα από τον αλγόριθμο LRU όσο αφορά τη μετρική Delay Start. Καθώς το μέγεθος της cache μειώνεται οι διαφορές μεταξύ των αποτελεσμάτων των δύο αλγορίθμων μειώνονται. Για παράδειγμα ενώ για RCS=10% το Delay Start που δίνει ο LFLRU για a=0.8 και a=1,17 είναι περίπου 120% και 100% αντίστοιχα, χειρότερο από αυτό του LRU, καταλήγει για RCS=0.3% να είναι μόνο 8% και 3% αντίστοιχα, χειρότερο. Παρακάτω παρουσιάζουμε ενδεικτικά τα αποτελέσματα του Delay Start όταν **RCS=2%** και **a=0,8**.



Γραφική Παράσταση 3.11

Στην περίπτωση αυτή ο LFLRU παρουσιάζει αποτελέσματα που είναι κατά περίπου 50% χειρότερα από αυτά που πετυχαίνει ο αλγόριθμος LRU.

3.3.3 Γενικά σχόλια

Σίγουρα η υλοποίηση ενός Proxy Cache Server εφαρμόζοντας την τεχνική αποθήκευσης Variable Chunk Size αντί της τεχνικής Fixed Chunk Size αλλάζει σε κάποια σημεία τις τιμές των δύο μετρικών απόδοσης. Σε όλα τα σενάρια που εξετάστηκαν, ο αλγόριθμος αντικατάστασης Least Frequently Least Recently Used καταφέρνει να δώσει λίγο καλύτερα αποτελέσματα από τον LRU στο steady state Byte Hit Ratio αλλά ταυτόχρονα δίνει χειρότερα αποτελέσματα στο Delay Start. Αυτό δεν οφείλεται σε κακή λειτουργία του αλγόριθμου LFLRU αλλά στην μεγάλη επίδραση που έχει το VCS στο σύστημα. Ο LFLRU συνεχίζει να επιτρέπει σε μικρό αριθμό video, των πιο δημοφιλών, να παραμένουν σχεδόν ολόκληρα στην cache πετυχαίνοντας έτσι ένα πολύ ψηλό ss_BHR που αγγίζει το HPF_ss_BHR αλλά και το αντίστοιχο Delay Start (κοντά στο HPF_Delay Start).

Αντίθετα, στην περίπτωση που χρησιμοποιείται ο αλγόριθμος αντικατάστασης LRU το σύστημα παρουσιάζει την εξής συμπεριφορά : το υψηλό ss_BHR που επιτυγχάνεται οφείλεται στην τεχνική αποθήκευσης VCS η οποία βάζει στην cache πολύ γρήγορα ένα σχετικά δημοφιλές video. Παράλληλα, λόγω της αδυναμίας του LRU να αναγνωρίσει τα δημοφιλή video επιτρέπεται σε ένα αριθμό video που δεν είναι δημοφιλή να παραμείνουν στην cache. Δεδομένου ότι τι αρχικό μέγεθος chunk που χρησιμοποιείται από την VCS είναι 1 unit (πολύ μικρό), δίνεται με αυτό τον τρόπο δυνατότητα σε ένα πολύ μεγάλο αριθμό από μη δημοφιλή video να παραμείνουν στην cache με αποτέλεσμα να έχουμε έτσι μικρό Delay Start.

3.4 Γενικά σχόλια για το σύστημα σταθερής κατανομής πιθανοτήτων αναζήτησης

Ένα σύστημα Proxy Cache που εργάζεται κάτω από μια στατική κατανομή πιθανοτήτων αναζήτησης φαίνεται να έχει διαφορετική συμπεριφορά ανάλογα με τις τεχνικές αντικατάστασης και αποθήκευσης που χρησιμοποιεί. Αν χρησιμοποιεί την τεχνική αντικατάστασης LFLRU τότε τα αποτελέσματα των δύο μετρικών ss_BHR και Delay Start δεν επηρεάζονται αισθητά από την επιλογή της τεχνικής

αποθήκευσης. Και για τις δύο τεχνικές FCS & VCS επιτυγχάνονται περίπου οι ίδιες τιμές. Επιπρόσθετα, και οι δύο μετρικές παρουσιάζουν μια σχετική σταθερότητα στις τιμές τους ανεξάρτητα από το μέγεθος του chunk (FCS) ή την τιμή του Acceleration Factor (VCS).

Αντίθετα, αν το σύστημα χρησιμοποιεί την τεχνική αντικατάστασης LRU τότε η επιλογή της τεχνικής αποθήκευσης έχει μεγάλη σημασία. Επιλέγοντας FCS, τότε καθώς αυξάνεται το μέγεθος του chunk το ss_BHR μειώνεται ενώ το Delay Start αυξάνεται. Ωστόσο, για VCS οι μετρικές παρουσιάζουν σχετική σταθερότητα στις τιμές τους.

Τέλος, και για τις δύο τεχνικές αποθήκευσης ο αλγόριθμος LFLRU δίνει πάντα καλύτερο ss_BHR. Αν χρησιμοποιείται FCS τότε παρουσιάζει έντονα καλύτερο ss_BHR από τον LRU, ενώ όταν χρησιμοποιείται VCS δίνει ελαφρώς καλύτερα αποτελέσματα. Όσο αφορά το Delay Start, για FCS ο LRU δίνει καλύτερα αποτελέσματα από τον LFLRU μόνο για μικρά chunk sizes ενώ για VCS ο LRU επιτυγχάνει πάντα καλύτερο Delay Start.

ΚΕΦΑΛΑΙΟ 4
Σύστημα με Μεταβλητή Κατανομή Πιθανοτήτων Αναζήτησης

4.1 Πληροφορίες για το σύστημα προσομοίωσης

Στο προηγούμενο κεφάλαιο εξετάσαμε την συμπεριφορά ενός Proxy Cache θεωρώντας ότι οι πιθανότητες με τις οποίες αναζητούνται τα video από τους χρήστες είναι σταθερές σε όλη την διάρκεια της προσομοίωσης του συστήματος. Θέλοντας να προσομοιώσουμε ένα σύστημα που θα βρίσκεται πιο κοντά σε πραγματικές συνθήκες λειτουργίας, προσθέσαμε στο σύστημά μας ακόμη ένα χαρακτηριστικό. Η πιθανότητα αναζήτησης κάθε video αλλάζει με την πάροδο μιας χρονικής περιόδου R. Ο τρόπος σύμφωνα με τον οποίο γίνεται η αλλαγή αυτή περιγράφεται λίγο πιο κάτω. Τα υπόλοιπα χαρακτηριστικά του συστήματος παραμένουν τα ίδια (αριθμός video N, μέγεθος/διάρκεια video L, ρυθμός άφιξης αιτήσεων λ και η διάρκεια προσομοίωσης).

Η αλλαγή των πιθανοτήτων αναζήτησης κάθε R ώρες δίνει την δυνατότητα σε video που δεν ήταν τόσο δημοφιλή μέχρι την δεδομένη στιγμή να αντικαταστήσουν στην cache τα video που πριν την αλλαγή θεωρούνταν πιο δημοφιλή. Η αλλαγή της κατανομής αναζήτησης των video υλοποιείται με την δημιουργία μια δεύτερης Zipf κατανομής η οποία έχει την ίδια παράμετρο a με την πρώτη. Η συσχέτιση των δύο Zipf κατανομών γίνεται με τη χρησιμοποίηση μια ακέραιας μεταβλητής K που μπορεί να πάρει τιμές από 1 μέχρι N. Θεωρούμε ότι τα video κατατάσσονται με τέτοιο τρόπο σε κάθε κατανομή έτσι ώστε το πιο δημοφιλές video να βρίσκεται στην πρώτη θέση ενώ το λιγότερο δημοφιλές να βρίσκεται στην τελευταία. Ξεκινώντας τη διαδικασία αλλαγής από το πιο δημοφιλές video της πρώτης Zipf κατανομής, αυτό τοποθετείται στην r_1 θέση της δεύτερης κατανομής. Η θέση r_1 επιλέγεται τυχαία μεταξύ των θέσεων 1 μέχρι K. Το δεύτερο πιο δημοφιλές video της κατανομής 1 τοποθετείται στην θέση r_2 της κατανομής 2. Η θέση r_2 επιλέγεται τυχαία μεταξύ των θέσεων 1 μέχρι min(N, K+1). Η θέση r_1 εξαιρείται από την επιλογή αυτή διότι είναι ήδη κατελειμένη. Η διαδικασία αυτή συνεχίζεται για όλα τα video. Με άλλα λόγια ένα video που βρίσκεται στην θέση q έχει την δυνατότητα να μετακινηθεί K θέσεις πιο κάτω σε δημοτικότητα, δηλαδή να μειωθεί η πιθανότητα αναζήτησής του, ή να μετακινηθεί q-1 θέσεις πιο πάνω, δηλαδή να αυξηθεί η

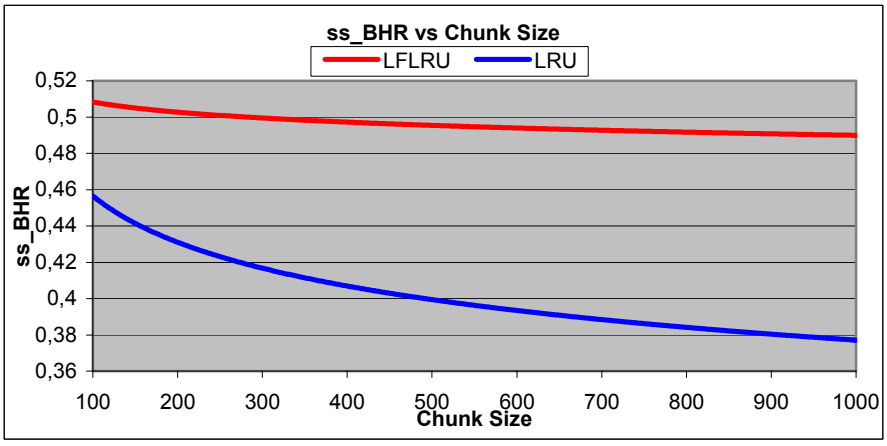
πιθανότητα αναζήτησής του. Το K για όλες τις προσομοιώσεις που έχουμε πραγματοποιήσει έχει την τιμή 10.

4.2 Μελέτη του συστήματος με Fixed Chunk Size

Χρησιμοποιώντας ως τεχνική αποθήκευσης το Fixed Chunk Size προσομοιώσαμε το σύστημα δυναμικής κατανομής πιθανοτήτων αναζήτησης. Εξετάσαμε το σύστημα για RCS=10%, 2%, 0.3% και a=0,8 και 1,17 υλοποιώντας όλους τους δυνατούς συνδυασμούς όπως και στο κεφάλαιο 3. Οι προσομοιώσεις γίνονται αλλάζοντας τις πιθανότητες αναζήτησης (όπως αναφέραμε ήδη) κάθε R ώρες. Έχουμε κάνει προσομοιώσεις για R=24 ώρες, 72 ώρες, 168 ώρες.

4.2.1 Αποτελέσματα μετρικής ss_BHR

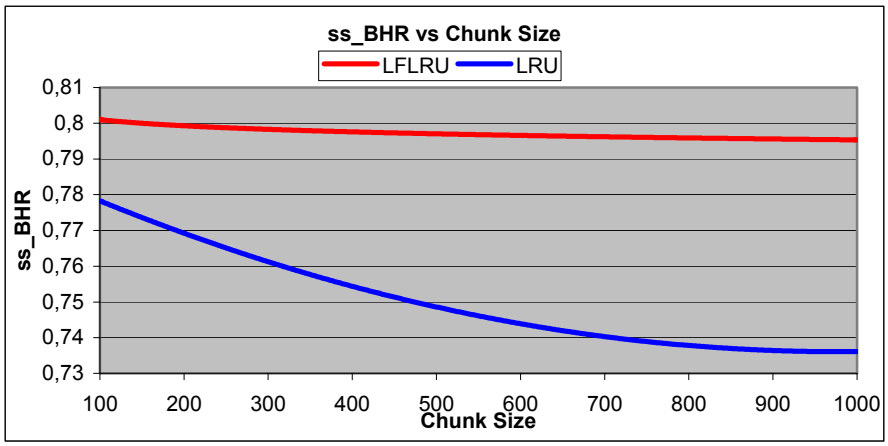
Σενάριο Ι : Αρχίζουμε την εξέταση του συστήματος με την περίπτωση που οι πιθανότητες αναζήτησης αλλάζουν κάθε 24 ώρες (**R=24h**), το **RCS=10%** και το **a=0,8**.



Γραφική Παράσταση 4.1

Παρατηρούμε ότι οι τιμές του ss_BHR που παράγονται εφαρμόζοντας την τεχνική αντικατάστασης LFLRU είναι κατά πολύ καλύτερες από αυτές που δίνει ο αλγόριθμος LRU. Οι ποσοστιαίες βελτιώσεις που πετυχαίνει ο LFLRU αρχίζουν από το 12,5% για μέγεθος chunk 100 units και φτάνουν την τιμή του 31% για μέγεθος chunk 1000 units.

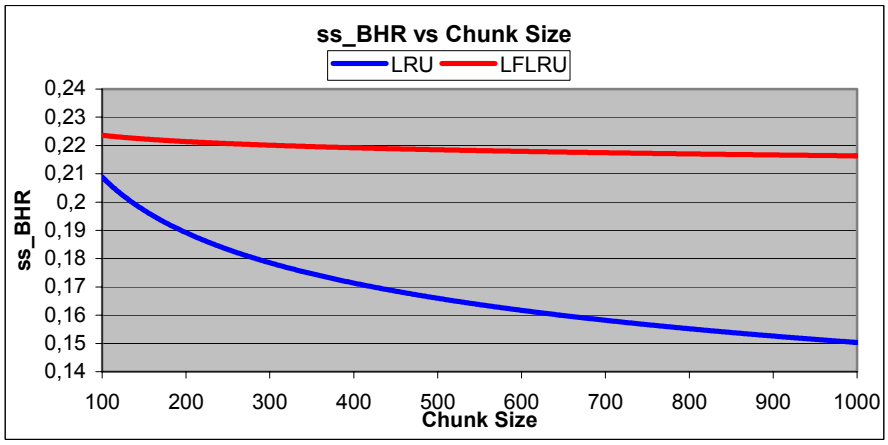
Σενάριο ΙΙ : Αν πολώσουμε την κατανομή των πιθανοτήτων (**a=1,17**) διατηρώντας το ίδιο μέγεθος cache (**RCS=10%**) και **R=24h** τότε παίρνουμε τα παρακάτω αποτελέσματα.



Γραφική Παράσταση 4.2

Όπως αναμέναμε, η γενική εικόνα που παρουσιάζει το γράφημα είναι ίδια με αυτή του γραφήματος 4.1 όπου a=0.8 με την μόνη διαφορά ότι τάξη της ποσοστιαίας βελτίωσης που πετυχαίνει ο αλγόριθμος LFLRU είναι μικρότερη. Αρχίζει από 3% για το μικρότερο μέγεθος chunk και καταλήγει περίπου στο 8% για το μεγαλύτερο. Αυτό οφείλεται στην πόλωση της κατανομής πιθανοτήτων αναζήτησης, κάτι που βοηθά πολύ το έργο του LRU (όπως εξηγήσαμε ήδη στο Κεφ. 3).

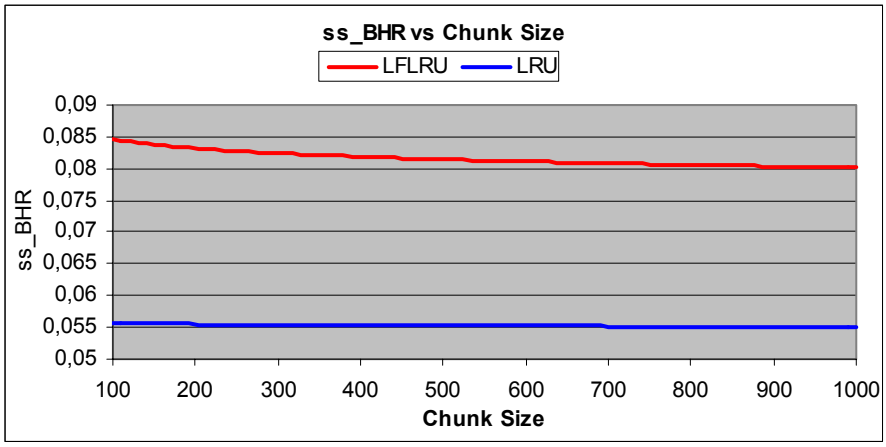
Σενάριο ΙΙΙ : Στη συνέχεια εξετάζουμε το σύστημα σε πιο δύσκολες συνθήκες λειτουργίας όπου το μέγεθος της cache είναι σχετικά μικρό. Έτσι θέτουμε **RCS=2%, a=0.8** και **R=24h**.



Γραφική Παράσταση 4.3

Έχοντας μικρή cache ο αλγόριθμος LFLRU όχι μόνο δίνει καλύτερα αποτελέσματα από τον αλγόριθμο LRU αλλά παράλληλα η ποσοστιαία διαφορά μεταξύ των τιμών του LFLRU και LRU είναι μεγαλύτερη από αυτή που επιτυγχάνει όταν ισχύει RCS=10%. Για μέγεθος chunk 100 units πετυχαίνει 6% βελτίωση σε σχέση με την τιμή του ss_BHR που δίνει ο LRU, ενώ καθώς το μέγεθος του chunk αυξάνεται καταφέρνει να δώσει βελτιώσεις της τάξης του 40%. Συμπεραίνουμε ότι για μικρή cache η δυσλειτουργία του LRU γίνεται πιο έντονη.

Σενάριο IV : Στο τελευταίο σενάριο με **R=24h** υποθέτουμε πολύ μικρή cache (**RCS=0.3%**) και όχι έντονα πολωμένη κατανομή πιθανοτήτων αναζήτησης (**a=0,8**).



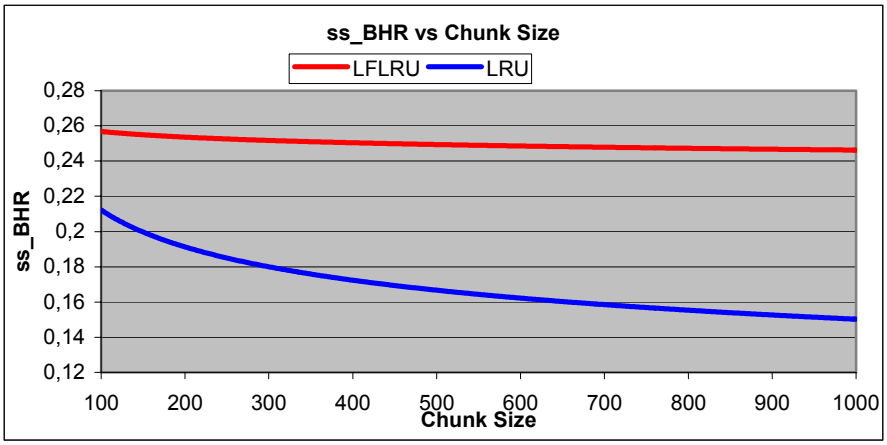
Γραφική Παράσταση 4.4

Αν και οι τιμές του ss_BHR που επιτυγχάνουν οι δύο αλγόριθμοι είναι πάρα πολύ μικρές λόγω του υπερβολικά μικρού μεγέθους της cache, συγκριτικά ο αλγόριθμος LFLRU δίνει περίπου 40% με 50% καλύτερα αποτελέσματα από τον αλγόριθμο LRU.

Στη συνέχεια μελετήσαμε τη συμπεριφορά του συστήματος όταν το χρονικό διάστημα, μετά την παρέλευση του οποίου αλλάζει η κατανομή των πιθανοτήτων αναζήτησης, αυξηθεί σε 72 ώρες. Η γενική εικόνα που παρουσιάζει το σύστημα είναι ίδια. Δηλαδή ο αλγόριθμος αντικατάστασης LFLRU δίνει πάντα καλύτερες τιμές ss_BHR από τον αλγόριθμο αντικατάστασης LRU σε όλα τα δυνατά σενάρια. Παράλληλα όμως επισημαίνουμε ότι οι ποσοστιαίες διαφορές μεταξύ των τιμών ss_BHR που δίνουν οι δύο τεχνικές αντικατάστασης, παρουσιάζουν μια αυξητική τάση όταν R=72. Με άλλα λόγια, ο αλγόριθμος LFLRU επιτυγχάνει μεγαλύτερες

τιμές όταν το χρονικό διάστημα μεταξύ της αλλαγής της κατανομής των πιθανοτήτων αναζήτησης είναι μεγαλύτερο. Στη συνέχεια παρουσιάζουμε ένα σενάριο με R=72 ώρες.

Σενάριο V : Επιλέξαμε να παρουσιάσουμε το σενάριο προσομοίωσης με **R=72h, RCS=2%** και **a=0.8** διότι αντιστοιχεί σε μία από τις δύσκολες περιπτώσεις που έχει να αντιμετωπίσει ένας proxy cache server λόγω του μικρού μεγέθους της cache.

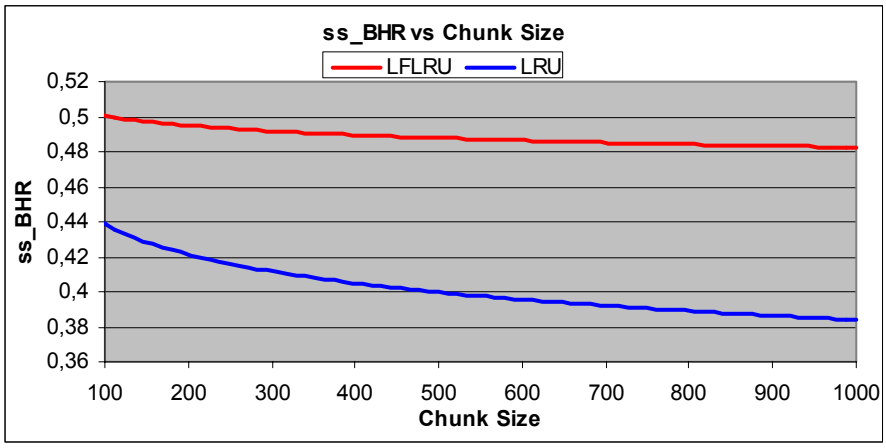


Γραφική Παράσταση 4.5

Και σ'αυτό το σενάριο ο αλγόριθμος LFLRU δίνει καλύτερα αποτελέσματα από τον LRU, οι ποσοστιαίες αυξήσεις ποικίλουν μεταξύ 21% και 57%. Αν συγκρίνουμε τα αποτελέσματα αυτά με τα αποτελέσματα στην γραφική παράσταση 4.3, θα παρατηρήσουμε ότι στην παρούσα περίπτωση ενώ οι τιμές του ss_BHR που δίνει ο LRU παραμένουν περίπου ίδιες, οι τιμές που λαμβάνονται όταν εφαρμόζεται ο LFLRU παρουσιάζουν αύξηση.

Θέλοντας να επιβεβαιώσουμε την τάση αύξησης που παρουσιάζει ο αλγόριθμος αντικατάστασης στις τιμές του ss_BHR καθώς αυξάνεται το διάστημα χρόνου R που μεσολαβεί μεταξύ των διαδοχικών αλλαγών των πιθανοτήτων αναζήτησης, προχωράμε στην παρουσίαση κάποιων αποτελεσμάτων όταν το R είναι ίσο με 168 ώρες. Επιπλέον, στόχος μας σε αυτή την περίπτωση είναι να εξετάσουμε την συμπεριφορά του Proxy Cache σε ένα περιβάλλον λειτουργίας όπου οι προτιμήσεις που έχουν οι χρήστες όσον αφορά τα διαθέσιμα δεν αλλάζουν πολύ συχνά.

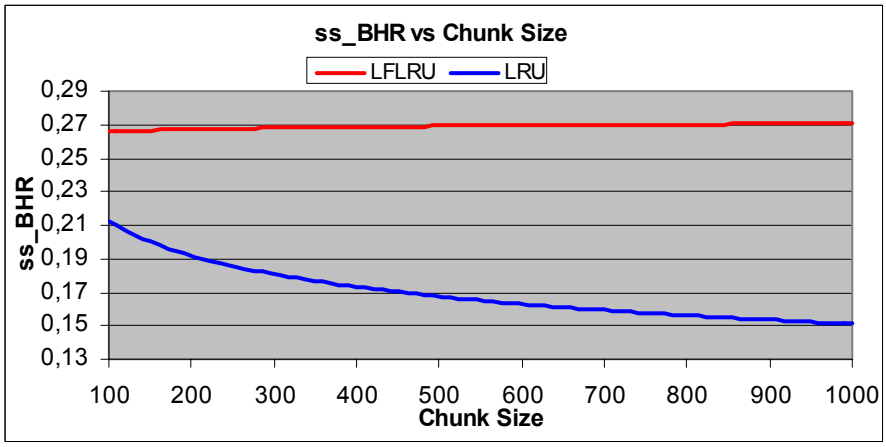
Σενάριο VI : Στο πρώτο σενάριο όπου **R=168h** θεωρούμε την ύπαρξη μιας cache χωρητικότητας 100 ολόκληρων video (**RCS=10%**) και παράμετρο της κατανομής Zipf **a=0,8**.



Γραφική Παράσταση 4.6

Παρατηρούμε ότι ο LFLRU αυξάνει το ss_BHR σε σχέση με τα αποτελέσματα που δίνει LRU το λιγότερο κατά 13% στα 100 units, ενώ η μεγαλύτερη βελτίωση που πετυχαίνει φτάνει το 28%.

Σενάριο VII : Ακολουθούν τα αποτελέσματα της προσομοίωσης στην περίπτωση που **R=168h**, **RCS=2%** και **a=0,8.**

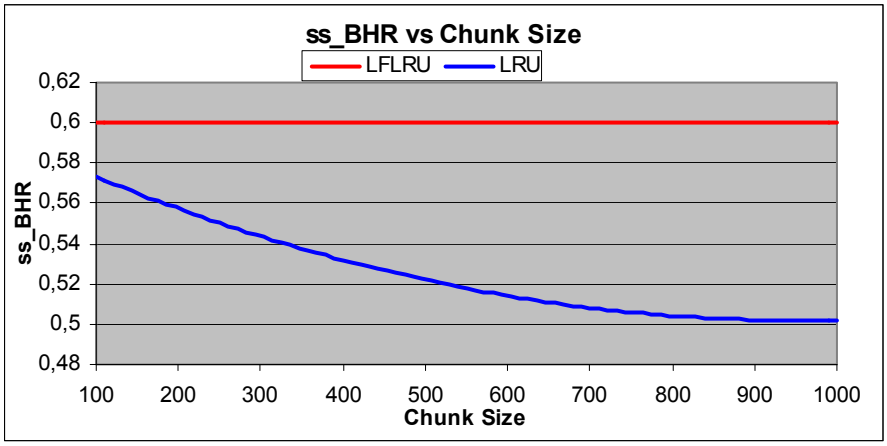


Γραφική Παράσταση 4.7

Από τα αποτελέσματα στο γράφημα 4.7 παρατηρούμε την σαφή υπεροχή του αλγορίθμου LFLRU έναντι του LRU, αφού ο LFLRU καταφέρνει να πετύχει μέχρι και 70% αύξηση στις τιμές του ss_BHR έναντι αυτών του LRU. Σημαντικό είναι να

επισημάνουμε τη σχετική σταθερότητα που παρουσιάζει το ss_BHR του LFLRU για όλες τις τιμές του μεγέθους chunk.

Σενάριο VIII : Τέλος, παρουσιάζουμε τα αποτελέσματα της προσομοίωσης ενός συστήματος στο οποίο οι αλλαγές των πιθανοτήτων γίνονται κάθε 168 ώρες, η cache είναι μικρή και η Zipf κατανομή έντονα πολωμένη (**R=168h, RCS=2%, a=1,17**).



Γραφική Παράσταση 4.8

Ο LFLRU παρουσιάζει ποσοστιαίες αυξήσεις στο ss_BHR που αρχίζουν από 4.7% και φτάνουν το 21%. Για όλα τα μεγέθη chunk επιτυγχάνει ουσιαστικά σταθερή τιμή ss_BHR ενώ αντίθετα ο LRU καθώς μεγαλώνουν τα μεγέθη των chunkς παρουσιάζει πτώση.

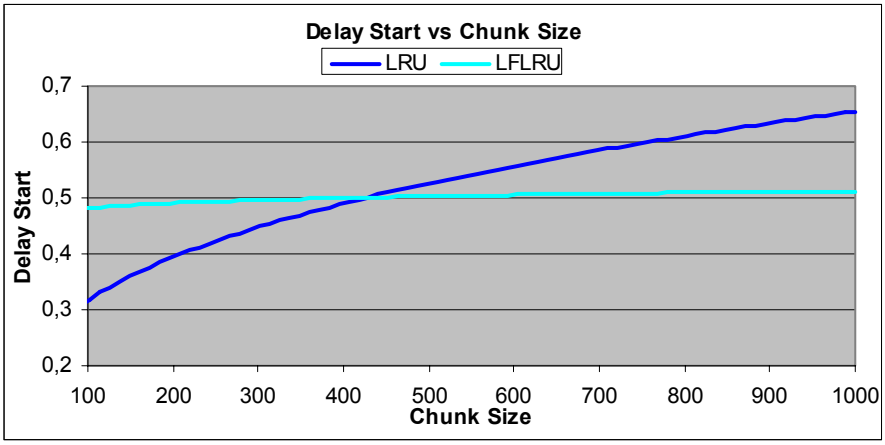
Η σύγκριση των σεναρίων με R=168h με αυτά των αντίστοιχων σεναρίων με R=72h και R=24h δείχνει ότι καθώς μεγαλώνει το R και για μικρό μέγεθος cache (RCS=2%), αυξάνονται οι τιμές του ss_BHR που δίνει ο αλγόριθμος LFLRU. Όταν η cache είναι σχετικά μεγάλη (RCS=10%), το ss_BHR δεν επηρεάζεται από τη μεταβολή του R παρουσιάζοντας σταθερή συμπεριφορά. Παράλληλα, αν και σε όλες τις περιπτώσεις ο LFLRU δίνει πολύ καλύτερο ss_BHR από τον LRU, μόνο όταν η cache είναι μεγάλη η επίδοση του LFLRU πλησιάζει την τιμή του HPF_ss_BHR.

4.2.2 Αποτελέσματα μετρικής Delay Start

Στην παράγραφο αυτή εκτιμούμε το κλάσμα των αιτήσεων των χρηστών που υφίστανται αρχική καθυστέρηση. Η εκτίμηση αυτή γίνεται μέσω της μετρικής Delay

Start. Για την παρουσίαση των αποτελεσμάτων επιλέγουμε σενάρια όπου οι αλλαγές των πιθανοτήτων αναζήτησης γίνονται κάθε σχετικά μικρά (R=24h) και μεγάλα (R=168h) χρονικά διαστήματα. Παράλληλα εξετάζονται caches που έχουν χωρητικότητα 100 000 units (RCS=10%) και χωρητικότητα 20 000 units (RCS=2%). Πρέπει να προσθέσουμε ότι βασικός στόχος μας είναι να καταλάβουμε την επίδραση των τεχνικών αντικατάστασης την λειτουργία του Proxy Cache Server.

Σενάριο Ι : Αρχίζουμε την μελέτη των αποτελεσμάτων του Delay Start για ένα σύστημα που οι πιθανότητες αναζήτησης των video μεταβάλλονται κατά την διάρκεια της προσομοίωσης και η τεχνική αποθήκευσης που χρησιμοποιείται είναι Fixed Chunk Size υλοποιώντας το σενάριο με **R=24h**, **RCS=10%** και **a=0,8**.



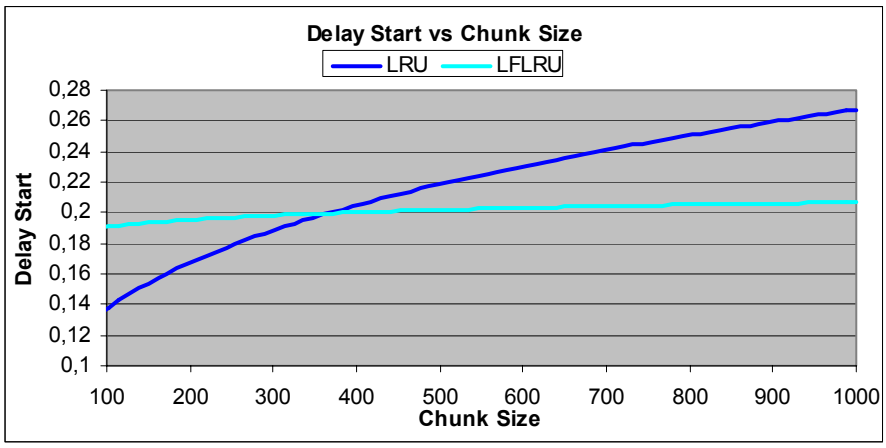
Γραφική Παράσταση 4.9

Παρατηρούμε ότι η γραφική παράσταση που περιγράφει τα αποτελέσματα του Delay Start όταν εφαρμόζεται ο αλγόριθμος LFLRU παρουσιάζει σταθερή συμπεριφορά για όλα τα μεγέθη chunk, ενώ αντίθετα το Delay Start του αλγορίθμου LRU αυξάνεται καθώς μεγαλώνει το μέγεθος του chunk που χρησιμοποιείται στο σύστημα. Το γεγονός αυτό έχει ως συνέπεια το φαινόμενο που συναντήσαμε στο κεφάλαιο 3 (παράγραφος 3.2.2), κατά το οποίο ο αλγόριθμος LRU δίνει καλύτερο Delay Start από αυτό του LFLRU όταν το μέγεθος του chunk είναι σχετικά μικρό. Καθώς όμως, το μέγεθος chunk μεγαλώνει, η τεχνική αντικατάστασης LFLRU πετυχαίνει μέχρι και 18% καλύτερα αποτελέσματα από αυτά της LRU.

Διατηρώντας το χρονικό διάστημα R=24h παρουσιάζουμε τα αποτελέσματα δύο ακόμη σεναρίων. Στο πρώτο πολώνουμε την κατανομή των πιθανοτήτων

αναζήτησης θέτοντας την παράμετρο της κατανομής Zipf a=1,17, ενώ στο δεύτερο μειώνουμε την χωρητικότητα της cache σε 20 000 units (RCS=2%).

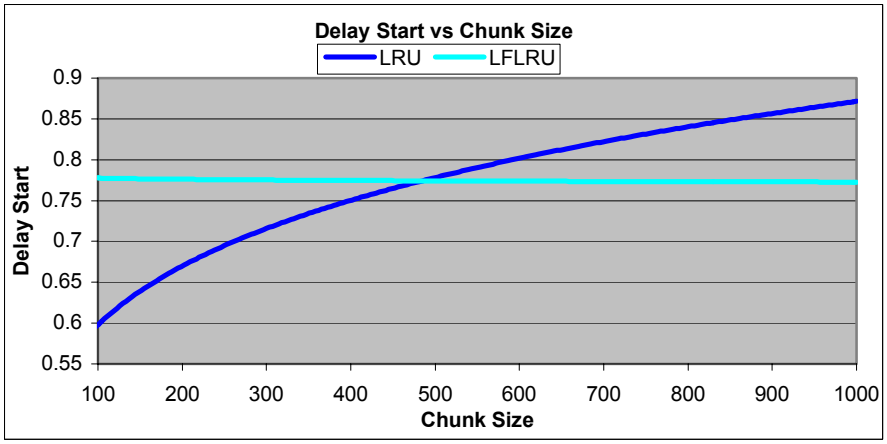
Σενάριο ΙΙ : R=24h, **RCS=10%** και **a=1,17**.



Γραφική Παράσταση 4.10

Παρατηρούμε ότι η συμπεριφορά των δύο αλγορίθμων παραμένει η ίδια με αυτή στο προηγούμενο σενάριο με κάποιες διαφορές στις τιμές των αποτελεσμάτων. Συγκρινόμενο με το σενάριο Ι το Delay Start μειώνεται και στους δύο αλγορίθμους λόγω της πόλωσης που υφίσταται η κατανομή των πιθανοτήτων αναζήτησης αλλά παράλληλα ο LFLRU αλγόριθμος καταφέρνει να πετύχει καλύτερη ποσοστιαία βελτίωση στις τιμές της μετρικής σε σχέση με τις τιμές που πετυχαίνει ο LRU που φτάνει μέχρι και 22% (στο σενάριο Ι η μέγιστη τιμή ποσοστιαίας βελτίωσης ήταν 18%).

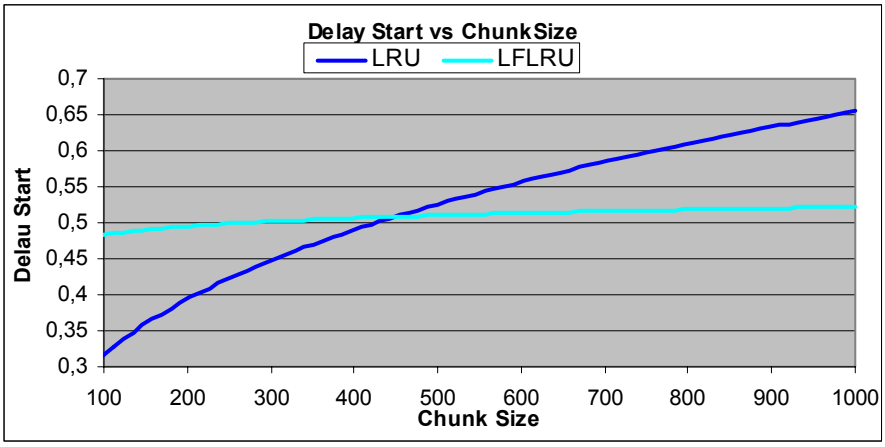
Σενάριο ΙΙΙ : Στο σενάριο αυτό ο της cache ελαττώνεται. Έτσι έχουμε **R=24h**, **RCS=2%** και **a=0,8**.



Γραφική Παράσταση 4.11

Ο αλγόριθμος LFLRU καταφέρνει παρόλο που η cache είναι μικρή, να παρουσιάσει μια σταθερή συμπεριφορά για όλα τα μεγέθη των chunks επιτυγχάνοντας ποσοστιαίες βελτιώσεις για μεγάλα μεγέθη chunk που φτάνουν το 6.5%. Το μικρό μέγεθος της ποσοστιαίας βελτίωσης οφείλεται στο μικρή χωρητικότητα της cache. Στη συνέχεια παρουσιάζουμε δύο επιπλέον σενάρια , όπου η αλλαγή των πιθανοτήτων αναζήτησης πραγματοποιείται κάθε 168 ώρες.

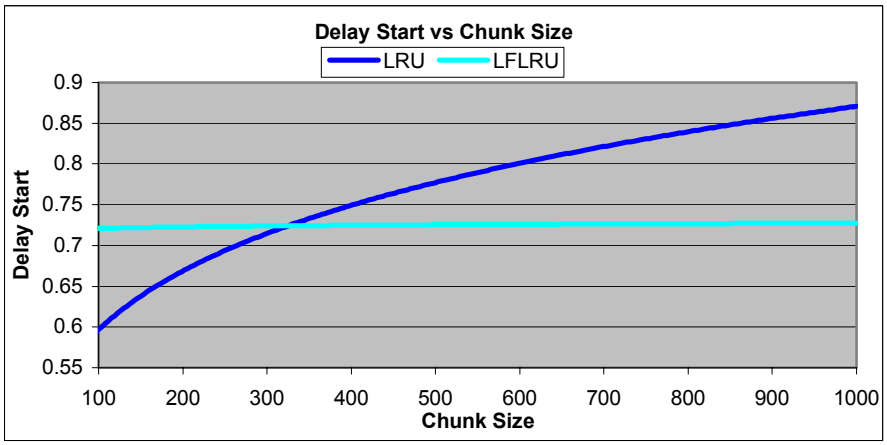
Σενάριο IV : Η προσομοίωση πραγματοποιείται χρησιμοποιώντας **R=168h**, **RCS=10%** και **a=0,8**.



Γραφική Παράσταση 4.12

Γενικά, βλέπουμε ότι και πάλι ο αλγόριθμος LFLRU δίνει καλύτερο Delay Start για μεγάλα μεγέθη chunk. Συγκρίνοντας τα αποτελέσματα αυτά με εκείνα της γραφικής παράστασης 4.9 παρατηρούμε ότι για RCS=10% και a=0,8 το χρονικό διάστημα R δεν επηρεάζει τις επιδόσεις των δύο αλγορίθμων. Αυτό οφείλεται στο σχετικά μεγάλο μέγεθος της cache. Όπως θα δούμε στο επόμενο σενάριο αυτό δεν ισχύει όταν η cache είναι μικρή.

Σενάριο V : Η παράμετρος R παραμένει ίδια (**R=168h**) ενώ η cache ελαττώνεται (**RCS=2%**). Επίσης, **a=0,8**.



Γραφική Παράσταση 4.13

Λόγω του μικρού μεγέθους της cache η ανάδειξη της υπεροχής του LFLRU έναντι του LRU καθώς το μέγεθος chunk μεγαλώνει είναι σημαντική φτάνοντας σε ποσοστιαία βελτίωση της τάξης του 12%. Αξίζει τον κόπο να ρίξομε ξανά μια ματιά στα αποτελέσματα της γραφικής παράσταση 4.11 και να παρατηρήσουμε ότι η βελτίωση που πετυχαίνει ο LFLRU στο παρόν σενάριο (R=168h) είναι σχεδόν διπλάσια από αυτή που πετυχαίνει όταν R=24h. Βλέπουμε ότι για μικρό μέγεθος cache το χρονικό διάστημα R επηρεάζει σε σημαντικό βαθμό την απόδοση της τεχνικής αντικατάστασης LFLRU.

4.2.3 Γενικά σχόλια

Στην παράγραφο αυτή εξετάσαμε ένα Proxy Cache Server, ο οποίος χρησιμοποιεί ως τεχνική αποθήκευσης τη FCS, όταν η κατανομή των πιθανοτήτων αναζήτησης μεταβάλλεται κάθε R ώρες.

Παρατηρήσαμε ότι αν το σύστημα χρησιμοποιεί την τεχνική αντικατάστασης LRU τότε τα αποτελέσματα των μετρικών ss_BHR και Delay Start δεν επηρεάζονται από τη μεταβολή του χρονικού διαστήματος R. Αντίθετα επηρεάζονται από τη μεταβολή του μεγέθους του chunk. Καθώς το μέγεθος του chunk μεγαλώνει το ss_BHR παρουσιάζει έντονη μείωση ενώ το Delay Start παρουσιάζει έντονη αύξηση.

Όταν ο αλγόριθμος LFLRU χρησιμοποιείται σαν τεχνική αντικατάστασης, τότε η τιμή του χρονικού διαστήματος R επηρεάζει τα αποτελέσματα των μετρικών απόδοσης όταν το μέγεθος της cache είναι μικρό. Στην περίπτωση αυτή, καθώς το R μεγαλώνει τόσο το ss_BHR όσο και το Delay Start βελτιώνονται. Ωστόσο, όταν η cache είναι μεγάλη η μεταβολή της τιμής του R δεν επηρεάζει αισθητά τις δύο

μετρικές. Επιπλέον, τόσο η μετρική ss_BHR όσο και η μετρική Delay Start παρουσιάζουν μια σταθερή συμπεριφορά για όλα τα μεγέθη chunk.

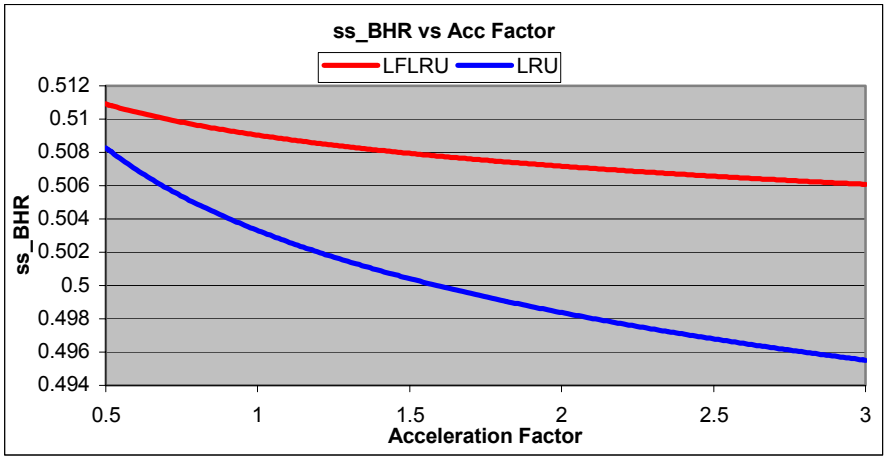
Συγκρίνοντας τους δύο αλγορίθμους αντικατάστασης, ο LFLRU δίνει πάντα καλύτερο ss_BHR για όλα τα σενάρια που εξετάστηκαν ενώ παράλληλα δίνει και καλύτερο Delay Start καθώς το μέγεθος του chunk μεγαλώνει.

4.3 Μελέτη του Συστήματος με Variable Chunk Size

Στην παράγραφο αυτή μελετάμε το σύστημα εφαρμόζοντας την τεχνική αποθήκευσης Variable Chunk Size. Οι προσομοιώσεις και πάλι γίνονται τόσο για μεγάλο μέγεθος cache όσο και για μικρό ενώ παράλληλα εφαρμόζεται ή όχι έντονα πολωμένη ή έντονα πολωμένη κατανομή πιθανοτήτων αναζήτησης. Η προσοχή μας επικεντρώνεται στην επίδραση που έχουν οι τεχνικές αντικατάστασης Least Frequently Least Recently Used και Least Recently Used στην απόδοση του συστήματος.

4.3.1 Αποτελέσματα Μετρικής ss_BHR

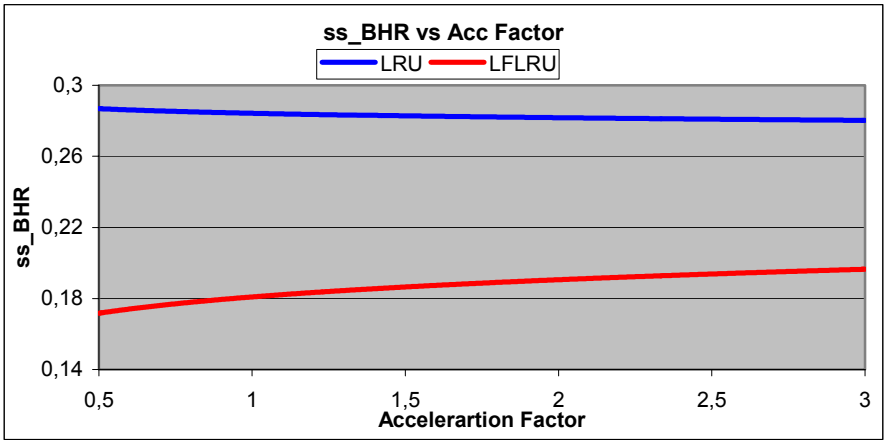
Σενάριο Ι : Στο πρώτο σενάριο προσομοίωσης που παρουσιάζουμε με VCS και δυναμικά μεταβαλλόμενη κατανομή πιθανοτήτων αναζήτησης χρησιμοποιούμε **R=24h**, **RCS=10%** και **a=0,8**.



Γραφική Παράσταση 4.13

Παρατηρούμε ότι αλγόριθμος LFLRU δίνει ελάχιστα καλύτερες τιμές ss_BHR από τον LRU (μέση ποσοστιαία βελτίωση 1,29%). Αν πολώσουμε την Zipf κατανομή θέτοντας α=1,17 η μέση ποσοστιαία βελτίωση γίνεται ακόμη πιο μικρή (0,15%).

Σενάριο ΙΙ : Προσομοιώνοντας το σύστημα για μικρή cache, η εικόνα που παρουσιάζει ο αλγόριθμος LFLRU είναι εντελώς διαφορετική. Έτσι για **R=24h**, **RCS=2%** και **a=0,8** τα αποτελέσματα είναι τα παρακάτω.

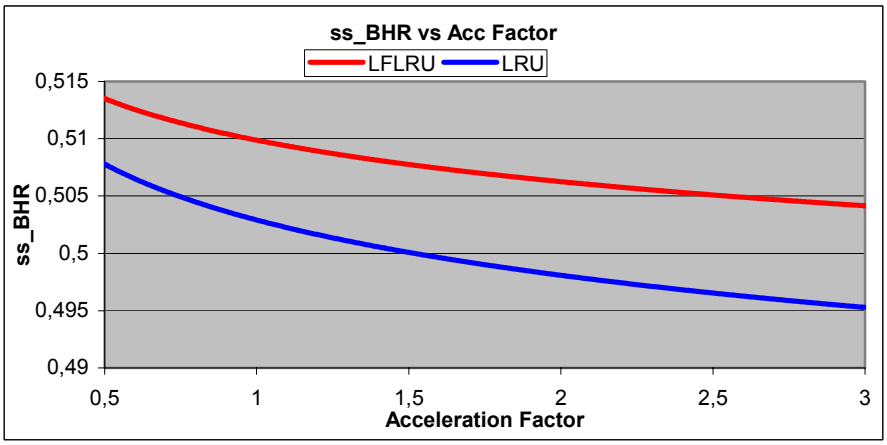


Γραφική Παράσταση 4.14

Όπως είναι φανερό από το πιο πάνω γράφημα όταν το μέγεθος της cache μειωθεί ο αλγόριθμος LRU δίνει πολύ καλύτερα αποτελέσματα από τον αλγόριθμο LFLRU. Τα αποτελέσματα του δεύτερου είναι κατά μέσο όρο 30% χειρότερα.

Στη συνέχεια παρουσιάζουμε δύο σενάρια προσομοίωσης με R=168h θέλοντας να παρατηρήσουμε την επίδραση που θα έχει στο ss_BHR η αύξηση του χρονικού διαστήματος με την παρέλευση του οποίου αλλάζουν οι πιθανότητες αναζήτησης των video.

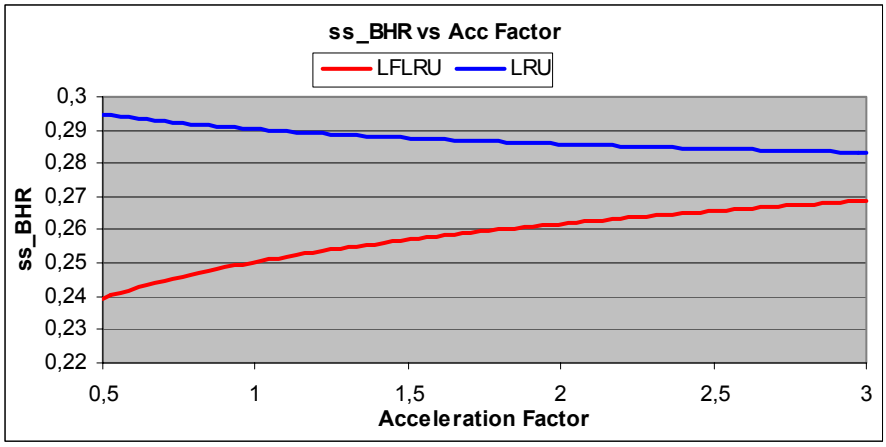
Σενάριο ΙΙΙ : R=168h, **RCS=10%** και **a=0,8**.



Γραφική Παράσταση 4.15

Το σύστημα παρουσιάζει γενικά παρόμοια συμπεριφορά με αυτή για μικρότερο διάστημα R. Οι τιμές του ss_BHR του LFLRU είναι 1% με 2% καλύτερες από αυτές του LRU. Επίσης παρατηρούμε μία μικρή αύξηση των τιμών του LFLRU σε σχέση με αυτές που ισχύουν όταν R=24h (γραφική παράσταση 4.13). Τα πράγματα γίνονται πιο ξεκάθαρα όταν το μέγεθος της cache ελαττωθεί.

Σενάριο IV : Στην περίπτωση αυτή διατηρήσαμε το **R=168h** και θέσαμε **RCS=2%** και **α=0,8**.



Γραφική Παράσταση 4.16

Αν και ο αλγόριθμος LFLRU συνεχίζει να δίνει χειρότερα αποτελέσματα από αυτά του LRU παρατηρούμε ότι οι τιμές του ss_BHR έχουν αυξηθεί περισσότερο από 40% σε σχέση με τις αντίστοιχες τιμές όταν R=24h. Αυτό συμβαίνει διότι ο αλγόριθμος LFLRU έχει σε αυτή την περίπτωση περισσότερο χρόνο για να αξιολογήσει καλύτερα τις δημοτικότητες των διαφόρων video.

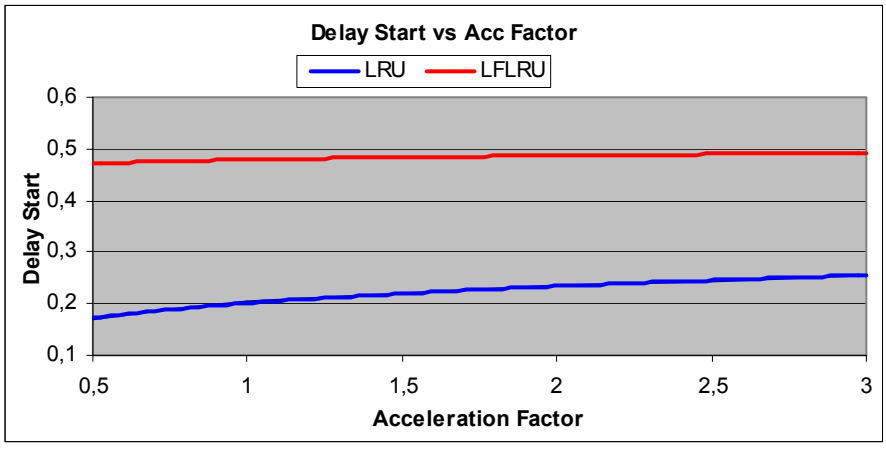
Μετά τα αποτελέσματα που παρουσιάσαμε τέσσερα γραφήματα (4.13 - 4.16) είναι αναγκαίο να δώσουμε κάποια εξήγηση για την συμπεριφορά του συστήματος. Τόσο για μικρό όσο και για μεγάλο R η γενική εικόνα παραμένει η ίδια. Για μεγάλη cache ο LFLRU δίνει καλύτερο ss_BHR από τον LRU πλησιάζοντας πολύ κοντά την τιμή HPF_ss_BHR ενώ αντίθετα για μικρή cache ο αλγόριθμος LFLRU δίνει πολύ χειρότερα αποτελέσματα.

Η αποτυχία που παρουσιάζει ο LFLRU οφείλεται σε δύο λόγους : α) στο πρώτο κριτήριο απόφασης που χρησιμοποιεί ο αλγόριθμος αντικατάστασης και β) στον τρόπο λειτουργίας του VCS. Ας θεωρήσουμε ότι τη χρονική στιγμή t_1 έγινε αλλαγή στην κατανομή των πιθανοτήτων αναζήτησης. Αμέσως μετά ένα καινούργιο

δημοφιλές video που προσπαθεί να εισέλθει και να μείνει στην cache θα πρέπει να ξεπεράσει σε αριθμό αναζητήσεων τα video που ήδη βρίσκονται στην cache. Παράλληλα το καινούργιο δημοφιλές video «κτίζεται» στην cache με βάση την τεχνική VCS αρχίζοντας με πολύ μικρά chunks (αρχικό μέγεθος chunk ίσο με 1 unit, βλ.Κεφ.2.2). Αν κατά την διάρκεια του «κτισίματος» του νέου δημοφιλούς video στην cache και πριν ο αριθμός των αναζητήσεων του αυξηθεί αρκετά, έρθει μια αναζήτηση για ένα «παλιό» δημοφιλές video, που λόγω της πρότερος κατάστασης θα έχει αποθηκευμένα αρκετά units στην cache, τότε το «παλιό» video θα ζητήσει ένα επιπλέον μεγάλο chunk προς αποθήκευση με αποτέλεσμα τα units αυτά να αφαιρεθούν πιθανότατα από το καινούργιο δημοφιλές video για το οποίο ο αριθμός των αναζητήσεών του δεν έχει ακόμα αυξηθεί αρκετά. Πολύ πιθανό το καινούργιο δημοφιλές video να απομακρυνθεί εντελώς από την cache. Κατά συνέπεια το ss_BHR μειώνεται σημαντικά.

4.3.2 Αποτελέσματα Μετρικής Delay Start

Μετά από μελέτη όλων των σεναρίων στην περίπτωση όπου οι πιθανότητες αναζήτησης μεταβάλλονται κάθε κάποιο R χρονικό διάστημα διαπιστώσαμε ότι σε όλες τις περιπτώσεις οι τιμές του Delay Start που δίνει ο αλγόριθμος αντικατάστασης LFLRU είναι κατά πολύ χειρότερες από αυτές που δίνει ο LRU. Ένα παράδειγμα δίνεται παρακάτω στην περίπτωση **R=168h**, **RCS=10%** και **a=0,8**.



Γραφική Παράσταση 4.17

Όπως βλέπουμε από τα αποτελέσματα στην γραφική παράσταση 4.17 ο αλγόριθμος LFLRU επιτυγχάνει περίπου την ίδια τιμή με αυτή της πολιτικής HPF όταν RCS=10%, ενώ για μικρή cache δίνει μεγαλύτερο Delay Start.

4.3.3 Τεχνική Zero References

Μετά τη μελέτη στις παραγράφους 4.3.1 και 4.3.2 οδηγηθήκαμε στην αναζήτηση μιας τεχνικής η οποία θα αντιμετωπίζει την αδυναμία που παρουσίασε ο αλγόριθμος LFLRU όταν εφαρμόζεται σε ένα σύστημα που λειτουργεί με βάση VCS και δυναμικά μεταβαλλόμενη κατανομή πιθανοτήτων αναζήτησης. Μετά από τη σχετική έρευνα καταλήξαμε στην εφαρμογή της τεχνικής Zero References.

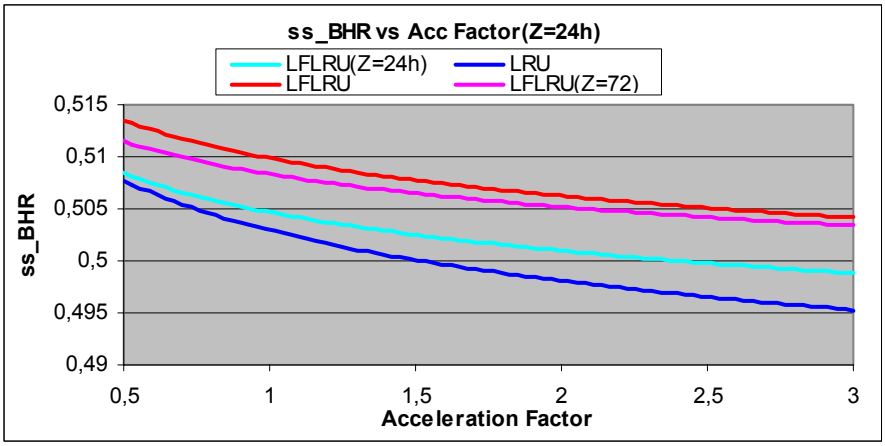
Σύμφωνα με την τεχνική αυτή η ιστορία των video που βρίσκονται αποθηκευμένα στην cache διαγράφεται κάθε χρονικό διάστημα Z ωρών. Δηλαδή, ο αριθμός των αναζητήσεων κάθε video που διατηρεί ο Cache Manager μηδενίζεται. Με άλλα λόγια, αν θεωρήσουμε ότι τα video πριν το μηδενισμό των μετρητών αναζήτησης των διαφόρων video στην cache βρίσκονται καταταγμένα με ιδανικό τρόπο σύμφωνα με τη δημοτικότητά τους, τότε στην πρώτη θέση βρίσκεται το πιο δημοφιλές video (με τον μεγαλύτερο αριθμό αναζητήσεων) και στην τελευταία το λιγότερο δημοφιλές (με το μικρότερο αριθμό αναζητήσεων). Αν ακριβώς μετά το μηδενισμό έρθει ένα αίτημα για το τελευταίο λιγότερο δημοφιλές video, τότε ο αριθμός αναζητήσεών του θα τεθεί ίσος με τη μονάδα και θα μετακινηθεί στην πρώτη θέση της κατάταξης σπρώχνοντας τα υπόλοιπα μια θέση πιο κάτω.

Με τον τρόπο αυτό βοηθείται ο αλγόριθμος LFLRU, μετά την αλλαγή της κατανομής των πιθανοτήτων αναζήτησης, να εκτιμήσει γρηγορότερα τις νέες πιθανότητες αναζήτησης που έχουν τα διάφορα video. Το μόνο θέμα που απομένει να γίνει προσεκτικά είναι η κατάλληλη επιλογή του διαστήματος Z. Το Z πρέπει να είναι μικρότερο ή ίσο, το τελευταίο είναι και το επιθυμητό, από το χρονικό διάστημα R. Παρακάτω παρουσιάζονται τα αποτελέσματα των μετρικών ss_BHR και Delay Start όταν εφαρμόζεται η τεχνική Zero References χρησιμοποιώντας δύο τιμές για τη διαρκεί Z. Η μία είναι ακέραιο υποπολλαπλάσιο του R ενώ η άλλη όχι.

- **Μελέτη Μετρικής ss_BHR**

R=168h RCS=10% a=0.8

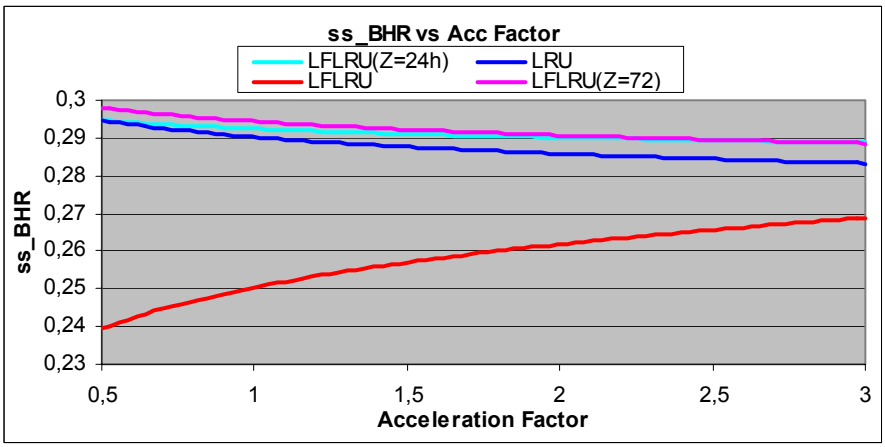
Acceleration Factor (g)	LRU ss_BHR	LFLRU ss_BHR	LFLRU ss_BHR Z=24h	LFLRU ss_BHR Z=72h
0,5	0,50695	0,511604	0,507278	0,510893
1,0	0,503177	0,510613	0,50598	0,508505
1,5	0,501699	0,507901	0,503679	0,507314
2,0	0,498052	0,511882	0,500332	0,505604
2,5	0,49695	0,506125	0,499796	0,504427
3,0	0,493874	0,498571	0,498082	0,502292



Γραφική Παράσταση 4.17

R=168h RCS=2% a=0.8

Acceleration Factor (g)	LRU ss_BHR	LFLRU ss_BHR	LFLRU ss_BHR Z=24h	LFLRU ss_BHR Z=72h
0,5	0,293941	0,231761	0,293459	0,297468
1,0	0,291216	0,261016	0,293454	0,294536
1,5	0,288348	0,261613	0,292955	0,293407
2,0	0,285306	0,263277	0,289435	0,291305
2,5	0,28489	0,263325	0,289555	0,289225
3,0	0,282378	0,262608	0,287461	0,287579



Γραφική Παράσταση 4.18

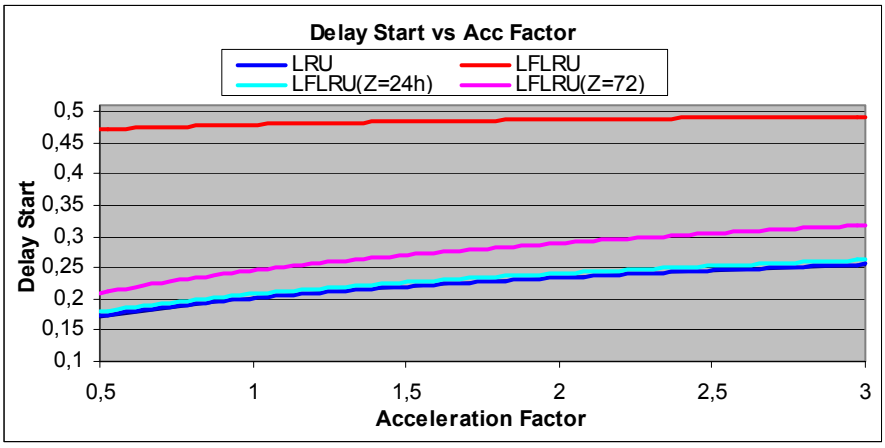
Από τη μελέτη των αποτελεσμάτων στις γραφικές παραστάσεις 4.17 και 4.18 παρατηρούμε ότι στην περίπτωση όπου η cache είναι μεγάλη (RCS=10%) η εφαρμογή της τεχνικής Zero References μειώνει το ss_BHR διότι στερεί από τον LFLRU σημαντική πληροφορία απόφασης. Ωστόσο, πρέπει να σημειώσουμε ότι και πάλι το ss_BHR που δίνει είναι μεγαλύτερο από αυτό που επιτυγχάνεται με τον αλγόριθμο LRU.

Αντίθετα όταν η cache είναι μικρή, η εφαρμογή της τεχνικής Zero References είναι εντυπωσιακά ευεργετική. Όχι μόνο καταφέρνει να βελτιώσει το ss_BHR του LFLRU αλλά επιτυγχάνει μεγαλύτερο ss_BHR από αυτό του LRU. Επίσης, παρατηρούμε ότι και για τις δύο τιμές του Z έχουμε παρόμοια επιθυμητά αποτελέσματα.

- **Μελέτη Μετρικής Delay Start**

R=168h RCS=10% a=0.8

Acceleration Factor (g)	LRU Delay Start	LFLRU DelayStart	LFLRU DelayStart Z=24h	LFLRU DelayStart Z=72h
0,5	0,171616	0,471621	0,178554	0,205324
1,0	0,203730	0,480119	0,210136	0,249697
1,5	0,219407	0,484575	0,225645	0,270478
2,0	0,235727	0,481883	0,242091	0,295277
2,5	0,243652	0,487879	0,250469	0,303007
3,0	0,255165	0,496658	0,263231	0,312843



Γραφική Παράσταση 4.19

Αφού μελετήσαμε την επίδραση της τεχνικής Zero References στη μετρική Delay Start καταλήξαμε ότι σε όλες τις περιπτώσεις βελτιώνει σε πολύ σημαντικό βαθμό το Delay Start που δίνει ο LFLRU. Επίσης διαπιστώσαμε ότι όσο πιο κοντά σε

ακέραιο υποπολλαπλάσιο του R είναι η τιμή του Z, τόσο η τιμή του Delay Start θα προσεγγίζει την τιμή που δίνει ο LRU. Ποτέ όμως δεν θα είναι μικρότερη από αυτή.

4.3.4 Γενικά Σχόλια

Στην παράγραφο 4.3 εξετάσαμε ένα σύστημα Proxy Cache Server το οποίο χρησιμοποιεί VCS και στο οποίο οι πιθανότητες αναζήτησης μεταβάλλονται ανά τακτά χρονικά διαστήματα διάρκειας R ωρών. Τόσο κατά την εφαρμογή της τεχνικής αντικατάστασης LRU όσο και κατά την εφαρμογή της τεχνικής LFLRU παρατηρήσαμε ότι η μεταβολή του Acceleration Factor g δεν επηρεάζει αισθητά τα αποτελέσματα των μετρικών απόδοσης. Επίσης, η μεταβολή του R επηρεάζει αισθητά τις μετρικές απόδοσης μόνο όταν εφαρμόζεται ο LFLRU σε cache που είναι σχετικά μικρή.

Συγκρίνοντας τους δύο αλγορίθμους αντικατάστασης παρατηρούμε ότι ο LFLRU δίνει καλύτερο ss_BHR όταν η cache είναι σχετικά μεγάλη (RCS=10%) ενώ στις υπόλοιπες περιπτώσεις ο LRU υπερέχει. Μελετώντας τα αποτελέσματα της μετρικής Delay Start ο LRU υπερέχει σε όλες τις περιπτώσεις. Χρησιμοποιώντας την τεχνική Zero References ο LFLRU καταφέρνει να πετύχει καλύτερο ss_BHR σε όλα τα σενάρια και να βελτιώσει σε μεγάλο βαθμό το Delay Start.

4.4 Γενικά Σχόλια Για Το Σύστημα Σταθερής Κατανομής Πιθανοτήτων Αναζήτησης

Τελειώνοντας τη μελέτη του συστήματος με δυναμικά μεταβαλλόμενη κατανομή πιθανοτήτων αναζήτησης των video, καταλήξαμε σε μερικά γενικά συμπεράσματα εξετάζοντας τα αποτελέσματα των μετρικών απόδοσης σε συνδυασμό με τις τεχνικές αντικατάστασης και αποθήκευσης.

Η χαμηλότερη τιμή Delay Start που μπορούμε να πετύχουμε είναι με το συνδυασμό VCS και LRU ενώ η μεγαλύτερη τιμή ss_BHR επιτυγχάνεται χρησιμοποιώντας VCS με LFLRU αν έχουμε μεγάλη cache ή χρησιμοποιώντας VCS με Zero References αν έχουμε μικρή cache. Επίσης, παρατηρούμε ότι αν η cache είναι σχετικά μεγάλη οι τιμές του ss_BHR που δίνει ο αλγόριθμος LFLRU σε συνδυασμό με FCS προσεγγίζουν τις μέγιστες τιμές ss_BHR που παίρνουμε όταν εφαρμόσουμε

VCS με LFLRU ή LRU ανεξάρτητα από το R. Ωστόσο, αν η cache είναι μικρή, η πιο πάνω παρατήρηση ισχύει και πάλι μόνο αν το χρονικό διάστημα R είναι σχετικά μεγάλο (π.χ.R=168 hours).

ΚΕΦΑΛΑΙΟ 5 Επίλογος

Σε αυτή τη διπλωματική εργασία μελετήσαμε ένα σύστημα Video Proxy Cache με στόχο να εξετάσουμε την επίδραση που έχουν οι δύο τεχνικές αντικατάστασης, Least Frequently Least Recently Used και Least Recently Used, σε συνεργασία με δύο τεχνικές αποθήκευσης, Fixed Chunk Size και Variable Chunk Size, στην απόδοση της λειτουργίας του συστήματος βασιζόμενοι στις μετρικές αξιολόγησης steady state Byte Hit Ratio (ss_BHR) και Delay Start. Επιπλέον, το σύστημα έχει προσομοιωθεί α) σε περιβάλλον όπου η κατανομή των πιθανοτήτων αναζήτησης παραμένει στατική καθ'όλη τη διάρκεια της προσομοίωσης, και β) σε περιβάλλον όπου η κατανομή των πιθανοτήτων αναζήτησης μεταβάλλεται μετά από παρέλευση σταθερών χρονικών διαστημάτων κατά την διάρκεια της προσομοίωσης. Παράλληλα, μελετήθηκαν περιπτώσεις όπου η χωρητικότητα της cache είχε διαφορετικά μεγέθη (μεγάλη-μικρή-πολύ μικρή) και η κατανομή των πιθανοτήτων αναζήτησης είχε διαφορετική πόλωση (ασθενή πόλωση-έντονη πόλωση).

5.1 Συμπεράσματα

Η εξαγωγή, συζήτηση και παρουσίαση των συμπερασμάτων από τις προσομοιώσεις του συστήματος έχουν αναφερθεί με λεπτομέρεια στα κεφάλαια 3 και 4. Εδώ θα παρουσιάσουμε συνοπτικά και ομαδοποιημένα τα κύρια συμπεράσματα.

1. Αρχίζουμε από την αξιολόγηση των προσομοιώσεων όταν ο Proxy Cache λειτουργεί σε περιβάλλον όπου οι πιθανότητες αναζήτησης των video είναι στατικές.
   - Η τεχνική αντικατάστασης LFLRU δίνει πολύ ψηλό ss_BHR τόσο όταν συνδυάζεται με την τεχνική εναποθήκευσης FCS όσο και με την τεχνική VCS. Οι τιμές του ss_BHR φαίνεται να προσεγγίζουν πάρα πολύ τις τιμές που επιτυγχάνει η βέλτιστη (αλλά μη υλοποιήσιμη) πολιτική High Popularity First, HPF (Κεφ. 2) ενώ παράλληλα δεν επηρεάζονται αισθητά από την μεταβολή του μεγέθους του Chunk και του Acceleration Factor (g), αντίστοιχα. Ίδια σταθερότητα παρουσιάζει και το Delay Start που επίσης προσεγγίζει την αντίστοιχη τιμή της πολιτικής HPF.

- Η τεχνική αντικατάστασης LRU παρουσιάζει έντονη πτώση στο ss_BHR και έντονη αύξηση στο Delay Start όταν εφαρμόζεται με την τεχνική αποθήκευσης FCS καθώς αυξάνεται το μέγεθος του chunk. Αντίθετα όταν συνεργάζεται με την τεχνική VCS τόσο το ss_BHR όσο και το Delay Start παρουσιάζουν σταθερή συμπεριφορά.
- Συγκρίνοντας τις δύο τεχνικές βλέπουμε ότι ο αλγόριθμος LFLRU δίνει πάντα καλύτερο ss_BHR. Ειδικά για FCS είναι έντονα καλύτερος. Παράλληλα δίνει και καλύτερο Delay Start όταν το μέγεθος του chunk στο FCS είναι σχετικά μεγάλο. Αντίθετα όταν εφαρμόζεται τεχνική αποθήκευσης VCS ο αλγόριθμος LRU δίνει πάντα καλύτερο Delay Start.

2. Στη συνέχεια παραθέτουμε τις παρατηρήσεις που κάναμε για το σύστημα όταν οι πιθανότητες αναζήτησης μεταβάλλονται δυναμικά μετά από τακτά χρονικά διαστήματα.

- Μελετώντας την συμπεριφορά του αλγορίθμου LFLRU σε συνδυασμό με τις δύο τεχνικές αποθήκευσης FCS και VCS παρατηρήσαμε έντονες διαφορές. Όταν χρησιμοποιείται η τεχνική FCS παρατηρείται μια σταθερή συμπεριφορά του ss_BHR και Delay Start με πολύ αξιόλογες τιμές οι οποίες δεν επηρεάζονται από την μεταβολή του μεγέθους του chunk και είναι συγκρίσιμες με τις τιμές της πολιτικής HPF. Η μεταβολή του χρονικού διαστήματος R επηρεάζει λίγο αρνητικά τις μετρικές μόνο όταν η cache είναι σχετικά μικρή. Η αδυναμία αυτή εξαλείφεται με την αύξηση του R. Αντίθετα όταν χρησιμοποιείται VCS ο αλγόριθμος δίνει καλά αποτελέσματα μόνο όταν η cache είναι σχετικά μεγάλη. Όταν η cache είναι μικρή ο αλγόριθμος αποτυγχάνει.
- Χρησιμοποιώντας τον αλγόριθμο LRU σε συνδυασμό με FCS παρατηρείται και πάλι, όπως στην περίπτωση της στατικής κατανομής πιθανοτήτων αναζήτησης, έντονη πτώση του ss_BHR και έντονη αύξηση του Delay Start καθώς το μέγεθος του chunk αυξάνεται. Αντίθετα όταν χρησιμοποιείται VCS οι μετρικές παρουσιάζουν πιο σταθερή συμπεριφορά με αρκετά ψηλό ss_BHR και πάρα πολύ χαμηλό Delay Start.

Η μεταβολή του R δεν επηρεάζει αισθητά τα αποτελέσματα των μετρικών σε κανένα σενάριο.

- Η τεχνική αντικατάστασης LFLRU παρουσιάζει πολύ καλύτερο ss_BHR από την τεχνική LRU σε όλες τις περιπτώσεις, αν χρησιμοποιείται η τεχνική αποθήκευσης FCS. Παράλληλα δίνει και καλύτερο Delay Start αν το μέγεθος του chunk είναι σχετικά μεγάλο. Ωστόσο, όταν χρησιμοποιείται η τεχνική VCS τότε ο αλγόριθμος LFLRU δίνει καλύτερο ss_BHR μόνο αν η χωρητικότητα της cache είναι σχετικά μεγάλη (RCS=10%) και πάντα χειρότερο Delay Start.
- Η αδυναμία που παρουσιάζει ο LFLRU όταν συνεργάζεται με την τεχνική VCS μπορεί να διορθωθεί χρησιμοποιώντας την τεχνική Zero References με αποτέλεσμα να επιτυγχάνεται καλύτερο ss_BHR από αυτό του LRU ακόμα και σε μικρά μεγέθη cache. Επιπλέον, βελτιώνεται το Delay Start προσεγγίζοντας την τιμή που πετυχαίνει ο LRU.

5.2 Επεκτάσεις

Σίγουρα οι επεκτάσεις που μπορεί να δεχθεί ένα τέτοιο σύστημα ενός Video Proxy Cache Server είναι πολλές και διαφορετικές ανάλογα με τον εκάστοτε στόχο. Για παράδειγμα, στην προσπάθεια να βελτιωθεί περαιτέρω το Delay Start θα μπορούσαν να μελετηθούν διαφοροποιημένες τεχνικές αξιοποίησης του χώρου που προσφέρει η cache. Μια τέτοια τεχνική παρουσιάζεται στο [8]. Σύμφωνα με αυτή, η cache μοιράζεται σε δύο μέρη, το ένα χρησιμοποιείται για να αποθηκεύονται μόνο τα video prefixes (σταθερού μεγέθους) ενώ στο άλλο αποθηκεύονται τα video suffixes. Διαφορετικές τεχνικές αντικατάστασης χρησιμοποιούνται από τον Cache Manager στα δύο μέρη της cache. Η τεχνική αντικατάστασης του δεύτερου μέρους συνδυάζεται με κατάλληλα επιλεγμένη τεχνική τμηματικής αποθήκευσης των video suffixes.

Τέλος, μια αναγκαία και φυσική επέκταση της παρούσας εργασίας περιλαμβάνει την ανάπτυξη τεχνικής που θα εκτιμά την τιμή της μεταβλητής Z που χρησιμοποιεί η τεχνική Zero References. Με άλλα λόγια το ζητούμενο είναι η ανάπτυξη μηχανισμού που θα διακρίνει σε σύντομο χρόνο το γεγονός ότι συντελέστηκε αλλαγή στις πιθανότητες αναζήτησης των videos, ώστε να μηδενίζει

άμεσα τους μετρητές αναζήτησης των videos που βρίσκονται αποθηκευμένα στην cache.

Βιβλιογραφία

[1] Scott A. Barnett, Gary J. Anido, and H. W. Beadle, "Caching policies in a distributed video on-demand system", in Proceedings of the Australian Telecommunication Networks and Applications Conference, Sydney, Australia, 1995.

[2] J. - P. Nussbaumer, B. V. Patel, F. Schaffa, and J. P. G. Sterbenz, "Networking requirements for interactive video on demand", IEEE Journal on Selected Areas in Communications, vol.13, 5, pp.779 - 87,1995.

[3] Christos Papadimitriou, Srinivas Ramanathan, P Venkat Rangan, and Srihari Sampathkumar, "Multimedia information caching for personalized video-on-demand", Computer Communications, vol.18, no. 3, Mar.1995.

[4] Bing Wang, Subhabrata Sen, Micah Adler, and Don Towsley, "Proxy-based distribution of streaming video over unicast/multicast connections", Technical Report UMASS TR-2001-05, University of Massachusetts,Amherst,2001.

[5] S.Ramesh, I.Rhee, and K.Guo, "Multicast with cache (mcache) : An adaptive zero-delay video-on-demand service", in Proceedings of the 2001 IEEE INFOCOM Conference, Anchorage,Alaska,Apr.2001.

[6] Jussi Kangasharju, Felix Hartanto, Martin Reisslein, and Keith W.Ross, "Distributing layered encoded video through caches", in Proceedings of the 2001 IEEE INFOCOM Conference, Anchorage, Alaska, Apr.2001.

[7] Reza Rejaie, Haobo Yu, Mark Handley, and Deborah Estrin, "Multimedia proxy caching mechanism for quality adaptive streaming applications in the internet", in Proceedings of the 2000 IEEE INFOCOM Conference, Tel Aviv, Israel, Mar.2000.

[8] Kun-Lung Wu, Philip S. Yu, and Joel L. Wolf, “Segment-based proxy caching of multimedia streams”, in Proceeding WWW 2001 Conference, Hong Kong, China.

[9] Elias Balafoutis, Antonis Panagakis, Nikolaos Laoutaris, and Ioannis Stavrakakis, “The impact of replacement granularity on video caching”, in Proceedings of IFIP Networking 2002 , Pisa, Italy, May 2002,pp.214-225.

[10] Elias Balafoutis, Antonis Panagakis, Nikolaos Laoutaris, and Ioannis Stavrakakis, “Study of the impact of replacement granularity and associated strategies on video caching”, accepted for journal publication.

[11] Barford Paul and Crovella Mark, “Generating representative web workloads for network and server performance evaluation”, in Measurement and Modeling of Computer Systems, 1998, pp. 151-160.

[12] Lee Breslau, Pei Cao, Li Fan, Graham Phillips, and Scott Shenker, “Web caching and zipf-like distributions: evidence and implications”, in Proceedings of the 1999 IEEE INFOCOM Conference, New York, 1999.

ΠΑΡΑΡΤΗΜΑ Α

Οι τιμές των μετρικών ss_BHR και Delay Start όταν εφαρμόζεται η πολιτική High Popularity First

RCS	**α**	**HPF_ss_BHR**	**HPF_Delay Start**
10%	0,8	0,525	0,470
10%	1,17	0,813	0,186
2%	0,8	0,303	0,695
2%	1,17	0,634	0,365
0.3%	0,8	0,128	0,871
0.3%	1,17	0,370	0,630

RCS = Relative Cache Size
α = παράμετρος της Zipf κατανομής πιθανοτήτων αναζήτησης

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή της εργασίας αυτής, κ. Μιχάλη Πατεράκη, για την αμέριστη βοήθεια που μου πρόσφερε και την άψογη συνεργασία που είχαμε καθ' όλη τη διάρκεια της διεκπεραίωσης της διπλωματικής εργασίας αυτής.

Με εκτίμηση

Χριστοφόρου Χριστόφορος