Άσκηση 3, Αντικατάσταση γραμμών σε κρυφές μνήμες

Β. Δημακόπουλος, Α. Ευθυμίου

Παραδοτέο: Δευτέρα 7 Μαΐου

1 Εισαγωγή

Ο κύριος στόχος των κρυφών μνημών είναι να κρατούν τις γραμμές οι οποίες είναι πιθανότερο ότι θα προσπελαστούν ξανά στο μέλλον. Ο αλγόριθμος αντικατάστασης παίζει κύριο ρόλο σε αυτό: διαλέγει πια γραμμή θα εκδιωχθεί από τη κρυφή μνήμη ώστε να ανοίξει χώρος για μια καινούρια γραμμή. Σε μια κρυφή μνήμη τύπου direct mapped δεν υπάρχουν επιλογές: κάθε γραμμή μπορεί να τοποθετηθεί σε μία μόνο θέση. Αλλά σε μια set-associative, κάθε γραμμή μπορεί να τοποθετηθεί σε οποιαδήποτε από της N θέσεις ενός set, όπου N είναι η associativity της κρυφής μνήμης. Ο πιο συνηθισμένος αλγόριθμος αντικατάστασης είναι ο LRU - least recently used. Δηλαδή η γραμμή που προσπελάστηκε παλιότερα (λιγότερο πρόσφατα) είναι αυτή που εκδιώχνεται.

Ο αλγόριθμος LRU δουλεύει καλά σε κρυφές μνήμες πρώτου επιπέδου (κοντά στον πυρήνα), αλλά έχει προβλήματα όταν χρησιμοποιείται σε άλλα επίπεδα, ειδικά όταν η κρυφή μνήμη είναι κοινόχρηστη μεταξύ πυρήνων. Για παράδειγμα έστω ένα σύστημα με 2 πυρήνες με ιδιωτικές κρυφές μνήμες πρώτου επιπέδου και μοιραζόμενη κρυφή μνήμη δεύτερου επιπέδου που είναι inclusive, δηλαδή ό,τι περιέχουν οι κρυφές μνήμες πρώτου επιπέδου πρέπει να υπάρχει και στη κρυφή μνήμη 2ου επιπέδου. Οι κρυφές μνήμες πρώτου επιπέδου έχουν associativity 2 και η κρυφή μνήμη δεύτερου επιπέδου έχει associativity 4. Ο ένας πυρήνας χρησιμοποιεί συνέχεια δύο γραμμές A, B που βρίσκονται στο ίδιο set και στα δύο επίπεδα κρυφής μνήμης. Αν ο δεύτερος πυρήνας τυχαίνει να φέρνει συνέχεια γραμμές που αντιστοιχούν στο ίδιο set με τις A, B στο 2ο επίπεδο, πολύ γρήγορα οι γραμμές A, B θα είναι παλιές για την κρυφή μνήμη επιπέδου 2. Έτσι θα αποφασίσει να τις αντικαταστήσει, για να φέρει τις καινούριες γραμμές του 2ο πυρήνα. Θα αναγκάσει όμως τη κρυφή μνήμη του πρώτου πυρήνα να εκδιώξει τις γραμμές A, B για να διατηρηθεί η ιδιότητα της inclusivity. Έτσι οι επόμενες προσπελάσεις του 1ου πυρήνα θα αστοχίσουν και θα πρέπει να προσκομιστούν από την κύρια μνήμη.

Το πρόβλημα είναι ότι η πληροφορία προσπέλασης φιλτράρεται από τις κρυφές μνήμες πρώτου επιπέδου και δεν περνάει στα επόμενα επίπεδα. Έτσι όταν η κρυφή μνήμη δεύτερου επιπέδου πρέπει να διαλέξει μια γραμμή για αντικατάσταση, μπορεί να διαλέξει μία γραμμή που, ενώ είναι παλιά για την ίδια, για το σύστημα συνολικά είναι πρόσφατα προσπελάσιμη.

2 Η άσκηση

Στην άσκηση αυτή θα διερευνήσετε μια άλλη μέθοδο αντικατάστασης στην κρυφή μνήμη τελευταίου επιπέδου (last level cache - llc) χρησιμοποιώντας τον προσομοιωτή sniper. Θα πρέπει να γράψετε κώδικα που να υλοποιεί ένα διαφορετικό αλγόριθμο αντικατάστασης, να προσθέσετε αυτή τη δυνατότητα στο sniper, να αλλάξετε τη σύνθεση του μηχανήματος που προσομοιώνετε ώστε το πρώτο και δεύτερο επίπεδο να χρησιμοποιεί LRU και το τρίτο (τελευταίο) την καινούρια μέθοδο. Χρησιμοποιώντας μετρο-προγράμματα που δίνονται από το sniper, θα συγκρίνετε τις επιδόσεις του καινούριου συστήματος με το παλιό που χρησιμοποιεί LRU για την κρυφή μνήμη τελευταίου επιπέδου.

Συγκεκριμένα θα πρέπει να υλοποιήσετε τον αλγόριθμο αντικατάστασης SRRIP (Static Re-Reference Interval Prediction) που παρουσιάζεται στη δημοσίευση:
A. Jaleel, K. B. Theobald, J. S. C. Steely, and J. Emer, "High performance cache replacement using re-reference interval prediction (RRIP)" *In Proceedings of the 37th annual international symposium on Computer architecture*, pages 60–71, Saint-Malo, France, June 2010. Αντίγραφο εδώ

Θα χρειαστεί να διαβάσετε το άρθρο τουλάχιστον μέχρι και το τμήμα 4.2. Θα πρέπει να υλοποιήσετε τη μέθοδο αντικατάστασης SRRIP με frequeny priority (FP) hit promotion policy και M=2 bits. Η ορολογία που χρησιμοποιεί για να περιγράψει τη νέα μέθοδο είναι λιγο παράξενη, αλλά η μέθοδος είναι απλή. Κάθε μία από τις N γραμμές ενός set της κρυφής μνήμης έχει μια τιμή (M bits) που ονομάζεται re-reference interval value (rrpv) που χρησιμοποιείται για τον αλγόριθμο αντικατάστασης.

Όσο μικρότερη είναι αυτή η τιμή, τόσο πιο σύντομα υποθέτουμε ότι θα ξανα-προσπελαστεί η γραμμή και όσο μεγαλύτερη είναι, τόσο πιο αργά θα ξαναπροσπελαστεί. Η δημοσίευση μετά τον αλγόριθμο SRRIP παρουσιάζει και άλλους, αλλά για την άσκηση δεν χρειάζεται να υλοποιηθούν.

Οι αλλαγές που θα χρειαστούν στον sniper περιορίζονται σε αρχεία του καταλόγου `common/core/memory_subsystem/cache`:

- `cache_set.h`, δήλωση καινούριας κλάσης για τον αλγόριθμο SRRIP χρησιμοποιώντας ως πρότυπο την αντίστοιχη του LRU.
- δημιουργία καινούριου αρχείου `cache_set_rrip.cc` χρησιμοποιώντας ως πρότυπο το `cache_set_lru.cc`, που υλοποιεί τον αλγόριθμο LRU.
- `cache_set.cc`, αλλαγή των createSet, parsePolicyType για προσθήκη του νέου αλγορίθμου.
- `cache_base.h`, προσθήκη του SRRIP στην enum ReplacementPolicy.

Προσοχή, όταν καλείται η μέθοδος getReplacementIndex που επιστρέφει σε πιο way του set (ποια από τις N θέσεις σύμφωνα με την παραπάνω σύντομη περιγραφή του αλγορίθμου) θα τοποθετηθεί η καινούρια γραμμή της κρυφής μνήμης, πρέπει να έχει ετοιμάσει και την κατάλληλη τιμή για το RRPV της *καινούριας* γραμμής. Όπως είναι γραμμένος ο προσομοιωτής δεν αλλάζει η τιμή αυτή κατά την τοποθέτηση της καινούριας γραμμής (δεν γίνεται καμία κλήση μεθόδου του cache set), οπότε πρέπει να την ετοιμάσετε κατά την αντικατάσταση της γραμμής που εκδιώχνεται.

Ξαναχτίστε το προσομοιωτή και φτιάξτε ένα καινούριο αρχείο σύνθεσης μηχανής (configuration) αντιγράφοντας το gainestown και αλλάζοντας τον αλγόριθμο αντικατάστασης της κρυφής μνήμης επιπέδου 3 (`perf_model/l3_cache/replacement_policy`) σε SRRIP.

Δοκιμάστε τους δύο αλγορίθμους τρέχοντας προσομοιώσεις δύο μηχανημάτων χρησιμοποιώντας συνθέσεις που διαφέρουν μόνο στον αλγόριθμο αντικατάστασης κρυφής μνήμης 3ου επιπέδου. Θα χρειαστεί να χρησιμοποιείσετε κάποιο σχετικά «βαρύ» πρόγραμμα όπως το splash2-fft που είναι διαθέσιμο από τα επιπλέον benchmarks που υπάρχουν για τον sniper. Ακολουθήστε τις οδηγίες στο σύνδεσμο και τρέξτε:
`./run_sniper -p splash2-fft -i small -n 4 -c gainestown` στον κατάλογο benchmarks του sniper. Οι διαφορές στο CPI μπορεί να είναι μικρές, αλλά μπορείτε να δείτε τις διαφορές στα ποσοστά αστοχίας (miss) της κρυφής μνήμης στο αρχείο sim.out.

Δοκιμάστε και άλλες εφαρμογές, άλλα μεγέθη κρυφών μνημών και καταγράψτε τις παρατηρήσεις σας. Τέλος παραδώστε μια σύντομη αναφορά με τα πειράματα και τα αποτελέσματά σας και τον κώδικα που έχετε αλλάξει στα αρχεία που αναφέρονται παραπάνω.