ΠΡΟΣ

1) **Όλα τα μέλη ΔΕΠ του Τμήματος Επιστήμης Υπολογιστών**

2) **Τους εκπροσώπους των Μεταπτυχιακών φοιτητών του Τμήματος Επιστήμης Υπολογιστών**

3) **Την Επταμελή Εξεταστική Επιτροπή**

4) **Όλα τα μέλη της Πανεπιστημιακής Κοινότητας**

Πρόσκληση σε Δημόσια Παρουσίαση της Διδακτορικής Διατριβής του

κ. Παπαγιάννη Αναστασίου

Doctoral Dissertation Defense

Mr. Papagiannis Anastasios

Την Πέμπτη, 22/10/2020 και ώρα 17:00 μ.μ. μέσω Τηλεδιάσκεψης και δημόσιας μετάδοσης (url): http://video.ucnet.uoc.gr/live/show/320, YouTube channel του Τμήματος: https://www.youtube.com/channel/UC7uE3QiMTQjkrpByB_Gnt6Q/live, στο Ηράκλειο, θα γίνει η δημόσια παρουσίαση και υποστήριξη της Διδακτορικής Διατριβής του υποψηφίου διδάκτορα του Τμήματος Επιστήμης Υπολογιστών κ. Παπαγιάννη Αναστασίου με θέμα:

" Πρόσβαση σε γρήγορες συσκευές αποθήκευσης μέσω απεικόνισης στη μνήμη"

"Memory-mapped I/O for Fast Storage"

ΠΕΡΙΛΗΨΗ

Οι εφαρμογές συνήθως προσπελαύνουν τις συσκευές αποθήκευσης χρησιμοποιώντας κλήσεις συστήματος (system calls) για ανάγνωση και εγγραφή. Επιπλέον, χρησιμοποιούν μια κρυφή μνήμη για να μειώσουν τις ακριβές προσπελάσεις στις συσκευές αποθήκευσης. Ωστόσο, συσκευές αποθήκευσης υψηλής ταχύτητας, παρέχουν πλέον πρόσβαση στα δεδομένα σε χαμηλό χρόνο. Κατά συνέπεια, το κόστος των αναζητήσεων στην κρυφή μνήμη (που γίνετε σε λογισμικό) αλλά και των κλήσεων συστήματος γίνεται σημαντικό υπό αυτές τις συνθήκες.

Σε αυτή τη διατριβή, προτείνουμε την διαχείριση της κρυφής μνήμης που χρησιμοποιείται για είσοδο/έξοδο (Ε/Ε) μέσω απεικόνισης των συσκευών στη μνήμη (memory mapped I/O), με στόχο την εξάλειψη του κόστους στις περιπτώσεις ευστοχίας (hits) στην κρυφή μνήμη. Με την απεικόνιση των συσκευών αποθήκευσης στη μνήμη (Linux mmap), ο χρήστης μπορεί να απεικονίσει ένα αρχείο στον χώρο των εικονικών διευθύνσεων μιας διεργασίας και να αποκτήσει πρόσβαση στα δεδομένα του χρησιμοποιώντας τις εντολές φόρτωσης και εγγραφής (load/store) του επεξεργαστή. Σε αυτήν την περίπτωση, το λειτουργικό σύστημα είναι υπεύθυνο για τη μεταφορά δεδομένων μεταξύ της κυρίας μνήμης και των συσκευών αποθήκευσης, τη δημιουργία/καταστροφή αντιστοιχίσεων εικονικής με φυσική μνήμη και τον χειρισμό της απόρριψης και εγγραφής σελίδων της κρυφής μνήμης στις συσκευές αποθήκευσης. Επομένως η διαχείριση των προσβάσεων στην κρυφή μνήμη που είναι εύστοχες (hits) γίνεται εξ ολοκλήρου από το υλικό μέσω του μηχανισμού για την μετάφραση εικονικών διευθύνσεων.

Αρχικά, σχεδιάζουμε και υλοποιούμε ένα σύστημα αποθήκευσης ζευγαριών κλειδιού-τιμής που χρησιμοποιεί την απεικόνιση στη μνήμη για την διαχείριση της κρυφής μνήμης ΕΕ και για να αλληλεπιδρά με τις συσκευές αποθήκευσης. Στην συνέχεια, παρουσιάζουμε τα πλεονεκτήματα του σε σύγκριση με τις αναζητήσεις στην κρυφή μνήμη που είναι υλοποιημένη σε λογισμικό. Δείχνουμε ότι το μονοπάτι στο λειτουργικό σύστημα Linux για την απεικόνιση στη μνήμη των συσκευές αποθήκευσης έχει πολλά προβλήματα στην περίπτωση εφαρμογών με μεγάλες ανάγκες σε πρόσβαση δεδομένων πάνω από συσκευές γρήγορης αποθήκευσης, όταν το σύνολο των δεδομένων τους δεν χωρά στη κύρια μνήμη. Σε αυτά περιλαμβάνονται: (1) η έλλειψη ελέγχου για την απόρριψη σελίδων, ειδικά κατά την περίπτωση των εγγράφων, (2) η μη επαρκής κλιμάκωση σε συνάρτηση με την αύξηση του αριθμού των νημάτων και (3) το υψηλό κόστος των σφαλμάτων σελίδας που συμβαίνουν κατά την διάρκεια των αστοχιών στην κρυφή μνήμη.

Κατόπιν, προτείνουμε τεχνικές για την αντιμετώπιση αυτών των μειονεκτημάτων. Προτείνουμε έναν μηχανισμό που χειρίζεται την απόρριψη και αντικατάσταση σελίδων μνήμης με βάση τις ανάγκες της εφαρμογής. Για να δείξουμε τη δυνατότητα εφαρμογής του, σχεδιάζουμε και υλοποιούμε ένα σύστημα αποθήκευσης ζευγαριών κλειδιού-τιμής που χρησιμοποιεί αυτόν τον μηχανισμό. Στη συνέχεια, καταργούμε όλα τα κεντρικά σημεία συνωστισμού κατά τον συγχρονισμό στο μονοπάτι της πραγματοποίησης Ε/Ε μέσω απεικόνισης στη μνήμη. Ο σχεδιασμός μας παρέχει κλιμακώσιμη απόδοση με τις συσκευές αποθήκευσης καθώς αυξάνεται ο αριθμός των πυρήνων και νημάτων στους εξυπηρετητές. Τέλος, διαχωρίζουμε την προστασία και τις κοινές λειτουργίες στο μονοπάτι κατά την πρόσβαση σε γρήγορες συσκευές αποθήκευσης μέσω απεικόνισης στη μνήμη. Αξιοποιούμε επεκτάσεις εικονικοποίησης (virtualization extensions) του επεξεργαστή για να μειώσουμε το κόστος των σφάλματων σελίδας (page faults) και να διατηρήσουμε την ισχυρή προστασία μεταξύ χρηστών που παρέχει το λειτουργικό σύστημα.

Αξιολογούμε τις προτεινόμενες επεκτάσεις χρησιμοποιώντας κυρίως συστήματα αποθήκευσης ζευγαριών κλειδιού-τιμής που αποτελούν σημαντικό κομμάτι για πολλά συστήματα επεξεργασίας και εξυπηρέτησης δεδομένων και δείχνουμε σημαντικά οφέλη όσον αφορά την κατανάλωση σε κύκλους του επεξεργαστή, την απόδοση και την προβλεψιμότητα.

Επιβλέπων: Καθηγητής, Άγγελος Μπίλας

ABSTRACT

Applications typically access storage devices using read/write system calls. Additionally, they use a storage cache to reduce expensive accesses to the devices. Fast storage devices provide high sequential throughput and low access latency. Consequently, the cost of cache lookups and system calls in the I/O path becomes significant at high I/O rates.

In this dissertation, we propose the use of memory-mapped I/O to manage storage caches and remove software overheads in the case of hits. With memory-mapped I/O (i.e. mmap), a user can map a file in the process virtual address space and access its data using processor load/store instructions. In this case, the operating system is responsible for moving data between DRAM and the storage devices, creating/destroying memory mappings, and handling page evictions/writebacks. Hits in memory-mapped I/O are handled entirely in hardware through the virtual memory mappings.

First, we design and implement a persistent key-value store that uses memory-mapped I/O to interact with storage devices, and we show the advantages of memory-mapped I/O for hits compared to explicit lookups in the storage cache. Then we show that the Linux memory-mapped I/O path suffers from several issues in the case of data-intensive applications over fast storage devices when the dataset does not fit in memory. These include: (1) the lack of user control for evictions of I/Os, especially in the case of writes, (2) scalability issues with increasing the number of threads, and (3) the high cost of page faults that happen in the common path for misses.

Next, we propose techniques to deal with these shortcomings. We propose a mechanism that handles evictions in memory-mapped I/O based on application needs. To show the applicability of this mechanism, we build an efficient memory-mapped I/O persistent key-value store that uses this mechanism. Subsequently, we remove all centralized contention points and provide scalable performance with increasing I/O concurrency and number of threads. Finally, we separate protection and common operations in the memory-mapped I/O path. We leverage CPU

virtualization extensions to reduce the overhead of page faults and maintain the protection semantics of the OS.

We evaluate the proposed extensions using mainly persistent key-value stores that are a central component for many analytics processing frameworks and data serving systems. We show significant benefits in terms of CPU consumption, performance (throughput and average latency), and predictability (tail latency).

Supervisor: Professor, Angelos Bilas