Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων

Υλοποίηση και Αξιολόγηση Δρομολογητή Σελίδων με Χρήση Τεχνικών Μηχανικής Μάθησης για Υβριδικά Συστήματα Μνήμης

Διπλωματικη Εργασια

του

ΑΝΔΡΕΑ Ι. ΜΑΓΚΟΥΤΑ

Επιβλέπων: Νεκτάριος Κοζύρης
Καθηγητής ΕΜΠ

Αθήνα, Απρίλιος 2025

Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων

Υλοποίηση και Αξιολόγηση Δρομολογητή Σελίδων με Χρήση Τεχνικών Μηχανικής Μάθησης για Υβριδικά Συστήματα Μνήμης

Διπλωματικη Εργασια

του

ΑΝΔΡΕΑ Ι. ΜΑΓΚΟΥΤΑ

Επιβλέπων : Νεκτάριος Κοζύρης
Καθηγητής ΕΜΠ

Συνεπιβλέπων : Κωνσταντίνος Νίκας
ΕΔΙΠ ΕΜΠ

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 8 Απριλίου 2025.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Νεκτάριος Κοζύρης	Γεώργιος Γκούμας	Διονύσιος Πνευματικάτος
Καθηγητής ΕΜΠ	Αναπληρωτής Καθηγητής ΕΜΠ	Καθηγητής ΕΜΠ

Αθήνα, Απρίλιος 2025

Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων

ΔΗΛΩΣΗ ΜΗ ΛΟΓΟΚΛΟΠΗΣ ΚΑΙ ΑΝΑΛΗΨΗΣ ΠΡΟΣΩΠΙΚΗΣ ΕΥΘΥΝΗΣ

Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ενυπογράφως ότι είμαι αποκλειστικός συγγραφέας της παρούσας Πτυχιακής Εργασίας, για την ολοκλήρωση της οποίας κάθε βοήθεια είναι πλήρως αναγνωρισμένη και αναφέρεται λεπτομερώς στην εργασία αυτή. Έχω αναφέρει πλήρως και με σαφείς αναφορές, όλες τις πηγές χρήσης δεδομένων, απόψεων, θέσεων και προτάσεων, ιδεών και λεκτικών αναφορών, είτε κατά κυριολεξία είτε βάσει επιστημονικής παράφρασης. Αναλαμβάνω την προσωπική και ατομική ευθύνη ότι σε περίπτωση αποτυχίας στην υλοποίηση των ανωτέρω δηλωθέντων στοιχείων, είμαι υπόλογος έναντι λογοκλοπής, γεγονός που σημαίνει αποτυχία στην Πτυχιακή μου Εργασία και κατά συνέπεια αποτυχία απόκτησης του Τίτλου Σπουδών, πέραν των λοιπών συνεπειών του νόμου περί πνευματικών δικαιωμάτων. Δηλώνω, συνεπώς, ότι αυτή η Πτυχιακή Εργασία προετοιμάστηκε και ολοκληρώθηκε από εμένα προσωπικά και αποκλειστικά και ότι, αναλαμβάνω πλήρως όλες τις συνέπειες του νόμου στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής άλλης πνευματικής ιδιοκτησίας.

(Υπογραφή)

.....
Ανδρέας Μαγκούτας
Διπλωματούχος Ηλεκτρολόγος
Μηχανικός και Μηχανικός
Υπολογιστών Ε.Μ.Π

8 Απριλίου 2025

Περίληψη

Η διαρκώς αυξανόμενη απήχηση των συστημάτων υβριδικής μνήμης έχει οδηγήσει στη σχεδίαση αρκετών μεθόδων για την αποτελεσματική διαχείριση των δεδομένων μεταξύ των διαφορετικών τύπων μνήμης. Μία από τις πλέον καινοτόμες προσεγγίσεις είναι αυτή του Kleio καθώς πρόκειται για έναν δρομολογητή σελίδων σε επίπεδο λογισμικού που στηρίζεται σε τεχνικές μηχανικής μάθησης για την κατανόηση και την πρόβλεψη της συμπεριφοράς των σελίδων. Αν και το Kleio υπόσχεται αξιοσημείωτη βελτίωση στην απόδοση, η αξιολόγηση του μέχρι στιγμής έχει περιοριστεί αποκλειστικά σε περιβάλλον προσομοίωσης. Το κύριο θέμα της παρούσας εργασίας είναι η ανάπτυξη και αξιολόγηση ενός δρομολογητή σελίδων που στηρίζεται στις αρχές του Kleio, σε ένα πραγματικό σύστημα ετερογενούς μνήμης με μια DRAM και μια NVM. Στόχος είναι η διερεύνηση των προκλήσεων και δυσκολιών που εμφανίζονται κατά τη μετάβαση από το περιβάλλον προσομοίωσης σε μια πραγματική εφαρμογή. Η ανάπτυξη του δρομολογητή γίνεται σε χώρο χρήστη και βασίζεται στη δειγματοληψία προσπελάσεων μνήμης μέσω μετρητών υλικού (PEBS). Τα πειραματικά αποτελέσματα δείχνουν πως η μέθοδος αυτή δεν είναι κατάλληλη για σελίδες του παραδοσιακού μεγέθους 4KB. Στην περίπτωση σελίδων μεγέθους 2MB, το Kleio εμφανίζει βελτίωση έναντι της συμβατικής πολιτικής δρομολόγησης που στηρίζεται στο ιστορικό προσπελάσεων. Παρ' όλα αυτά, η επίδοσή του αποκλίνει σημαντικά από την αναμενόμενη ιδανική συμπεριφορά, γεγονός που αποδίδεται κυρίως στις στρεβλώσεις που εισάγει η διαδικασία δειγματοληψίας.Το συμπέρασμα που προκύπτει είναι πως, με τις κατάλληλες προσαρμογές και βελτιστοποιήσεις, το Kleio μπορεί να αποτελέσει μια πρακτική και αποδοτική λύση για τη διαχείριση δεδομένων σε υβριδικά συστήματα μνήμης.

Λέξεις Κλειδιά

Σύστημα υβριδικής μνήμης, Δρομολογητής σελίδων, Μηχανική μάθηση, Kleio, PEBS

Abstract

The increasing adoption of hybrid memory systems has led to the development of various techniques aimed at efficiently managing data across different memory types. One particularly innovative approach is Kleio, a software-based page scheduler that leverages machine learning techniques to analyze and predict page behavior. Despite its promising potential, previous evaluations of Kleio have been restricted exclusively to simulation environments. The primary objective of this work is the development and evaluation of a page scheduler inspired by Kleio, deployed in a real system consisting of DRAM and NVM. Specifically, the study aims to explore the practical challenges and difficulties arising when transitioning from simulated environments to real-world implementations. The proposed scheduler operates in user space and relies on memory-access sampling using hardware performance counters (PEBS). Experimental results indicate that this methodology is not effective for standard-sized 4KB pages. In contrast, when applied to larger pages (2MB), the Kleio outperforms traditional history-based page management policies. However, its performance still deviates significantly from the ideal case, primarily due to variability induced by the sampling mechanism. Overall, the findings suggest that, given appropriate optimizations, this approach holds practical promise for effective data management in hybrid memory systems.

Keywords

Hybrid memory systems, Page scheduler, Machine learning, Kleio, PEBS

στους γονείς μου

Ευχαριστίες

Θα ήθελα καταρχήν να ευχαριστήσω τον καθηγητή κ. Κοζύρη Νεκτάριο καθώς και τον συνεπιβλέποντα κ. Νίκα Κωνσταντίνο για την επίβλεψη αυτής της διπλωματικής εργασίας και για την ευκαιρία που μου έδωσε να την εκπονήσω στο Εργαστήριο Υπολογιστικών Συστημάτων. Επίσης ευχαριστώ ιδιαίτερα τον κ. Ψωμαδάκη Στράτο για την καθοδήγησή του και την εξαιρετική συνεργασία που είχαμε. Τέλος θα ήθελα να ευχαριστήσω τους γονείς μου για την καθοδήγηση και την ηθική συμπαράσταση που μου προσέφεραν όλα αυτά τα χρόνια.

Αθήνα, Απρίλιος 2025

Ανδρέας Μαγκούτας

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Κεφάλαιο 1

Εισαγωγή

Η συνεχής μεγέθυνση του όγκου δεδομένων που καλούνται να διαχειριστούν οι εφαρμογές των σύγχρονων datacenters σε συνδυασμό με την αύξηση των πυρήνων σε κάθε επεξεργαστικό chip, που επιτρέπει την ταυτόχρονη εκτέλεση πολλαπλών νημάτων, οδηγούν σε σημαντική επιβάρυνση στο σύστημα της κύριας μνήμης. Δεδομένου ότι η κλασική διαδικασία προσωρινής μεταφοράς των δεδομένων σε δίσκο (swapping) επιβαρύνει σε μεγάλο βαθμό τη συνολική επίδοση, είναι ξεκάθαρο πως έχει δημιουργηθεί η ανάγκη για ταχεία μνήμη πολύ μεγάλης χωρητικότητας [3]. Η διαρκής εξέλιξη και αναπροσαρμογή των DRAM, που αποτελούν την πλέον διαδεδομένη μορφή μνήμης, δεν προσφέρει μια κλιμακώσιμη λύση ικανή να καλύψει τις νέες απαιτήσεις. Συγκεκριμένα, σε μια μνήμη DRAM, η αύξηση της συνολικής χωρητικότητας απαιτεί τη δραστική μείωση του μεγέθους κάθε κελιού, γεγονός που συνεπάγεται εκθετική αύξηση του κόστους, σημαντική άνοδο της κατανάλωσης ενέργειας λόγω υψηλών τιμών ρεύματος διαρροής και αδράνειας και περιορισμένη αντοχή σε σφάλματα [4, 5, 6].

Μια προσέγγιση που έχει συμβάλει καθοριστικά στην αντιμετώπιση του εν λόγω προβλήματος είναι η δημιουργία υβριδικών (ή ετερογενών) συστημάτων μνήμης (hybrid memory systems). Πρόκειται για τον συνδυασμό πολλαπλών τεχνολογιών μνήμης που διαφέρουν μεταξύ τους ως προς χαρακτηριστικά όπως η επίδοση, το κόστος, η κατανάλωση ενέργειας και η αξιοπιστία. Ο στόχος ενός τέτοιου συστήματος είναι η αξιοποίηση των πλεονεκτημάτων κάθε τεχνολογίας για τη βελτιστοποίηση της συνολικής απόδοσης. Το πιο συνηθισμένο μοντέλο υβριδικής μνήμης αποτελείται από μια γρήγορη μνήμη μικρής χωρητικότητας, συνήθως DRAM, και μια πιο αργή μη-πτητική μνήμη (Non Volatile Memory), η οποία προσφέρει μεγαλύτερη χωρητικότητα αλλά συνοδεύεται από υψηλότερους χρόνους προσπέλασης. [7].

Εκτός από την επιλογή κατάλληλης αρχιτεκτονικής, μια σημαντική πρόκληση στη σχεδίαση ενός υβριδικού συστήματος είναι η αποτελεσματική διαχείριση των δεδομένων ανάμεσα στις διαφορετικές μνήμες. Αυτή η λειτουργία ανατίθεται στον δρομολογητή σελίδων (page scheduler), ένα βασικό στοιχείο που μπορεί να υλοποιηθεί σε επίπεδο υλικού, λογισμικού ή ως ένας συνδυασμός των δύο και αποσκοπεί στη δυναμική ανακατανομή των δεδομένων μεταξύ των μνημών, διασφαλίζοντας τη βέλτιστη απόδοση του συστήματος. Εν ολίγοις, ο ρόλος ενός δρομολογητή είναι να μεταφέρει τις σελίδες μνήμης που αναμένεται να χρησιμοποιηθούν σύντομα από την εφαρμογή στη γρήγορη μνήμη, βελτιώνοντας την απόδοση του συστήματος. Μια πρωτοποριακή προσέγγιση σε αυτόν τον τομέα είναι το Kleio, ένας δρομολογητής που χρησιμοποιεί τεχνικές μηχανικής μάθησης για να προβλέπει τη συμπεριφορά

των σελίδων και να τις τοποθετεί στρατηγικά στις κατάλληλες μνήμες. Η λειτουργία του βασίζεται στην εκπαίδευση νευρωνικών δικτύων τύπου LSTM και οι μετρήσεις έχουν δείξει ότι σε πολλές περιπτώσεις η απόδοσή του πλησιάζει αυτή ενός ιδανικού δρομολογητή που διαθέτει προγνωστική ικανότητα για τη συμπεριφορά των σελίδων.

1.1 Αντικείμενο της διπλωματικής

Παρόλο που το Kleio φαίνεται να αποτελεί μια υποσχόμενη λύση για το πρόβλημα διαχείρισης δεδομένων σε ετερογενή συστήματα μνήμης, η αξιολόγησή του μέχρι στιγμής περιορίζεται σε περιβάλλον προσομοίωσης. Το γεγονός αυτό αφήνει ανοιχτά ερωτήματα σχετικά με την αποτελεσματικότητα της εφαρμογής μεθόδων μηχανικής μάθησης σε πραγματικά συστήματα. Αντικείμενο της παρούσας διπλωματικής εργασίας είναι η ανάπτυξη ενός δρομολογητή σελίδων που αξιοποιεί τις ιδέες του Kleio σε ένα πραγματικό σύστημα ετερογενούς μνήμης. Στόχος είναι η αξιολόγηση της εφαρμοσιμότητας και της αποτελεσματικότητας των τεχνικών μηχανικής μάθησης σε πραγματικές συνθήκες, καθώς και η διερεύνηση των προκλήσεων που προκύπτουν κατά τη μετάβαση από το στάδιο της προσομοίωσης σε ένα πραγματικό σύστημα.

1.2 Δομή της εργασίας

Η εργασία είναι διαρθρωμένη σε πέντε κεφάλαια:

- Στο Κεφάλαιο 2, παρουσιάζεται το θεωρητικό υπόβαθρο που αφορά τους δρομολογητές σελίδων. Συγκεκριμένα, αναλύονται τα χαρακτηριστικά των μη πτητικών μνημών, οι τοπολογίες υβριδικών συστημάτων μνήμης και οι βασικές σχεδιαστικές επιλογές που επηρεάζουν τη λειτουργία ενός δρομολογητή σελίδων. Επιπλέον, περιγράφεται αναλυτικά το Kleio, καλύπτονται οι βασικές αρχές της μηχανικής μάθησης που είναι απαραίτητες για την εφαρμογή του και εξετάζονται σχετικές ερευνητικές εργασίες που συνδέονται με την παρούσα μελέτη.
- Στο Κεφάλαιο 3 περιγράφεται το περιβάλλον που χρησιμοποιήθηκε για την ανάπτυξη και την αξιολόγηση μαζί με τα εργαλεία που αξιοποιήθηκαν και εξηγείται αναλυτικά η υλοποίηση του δρομολογητή σελίδων.
- Στο Κεφάλαιο 4 παρουσιάζεται η πειραματική αξιολόγηση του δρομολογητή σελίδων, με ανάλυση των προκλήσεων που αναδείχθηκαν κατά τη διαδικασία. Εξετάζονται οι τρόποι προσαρμογής σε αυτές τις προκλήσεις, με στόχο τη βελτίωση της αποτελεσματικότητας και της απόδοσης του συστήματος.
- Στο Κεφάλαιο 5 συνοψίζονται τα κύρια συμπεράσματα της εργασίας, ενώ παράλληλα προτείνονται κατευθύνσεις για μελλοντική έρευνα και περαιτέρω ανάπτυξη.

Μέρος I

Θεωρητικό Μέρος

Κεφάλαιο 2

Θεωρητικό υπόβαθρο

Στο κεφάλαιο αυτό παρουσιάζονται οι θεωρητικές βάσεις για την υλοποίηση ενός δρομολογητή σελίδων. Αρχικά, αναλύεται ο τρόπος λειτουργίας των μη πτητικών μνημών, ενώ στη συνέχεια διερευνώνται οι προκλήσεις και οι εναλλακτικές επιλογές που προκύπτουν από τη σχεδίαση των δρομολογητών, βάσει σύγχρονων ερευνητικών μελετών. Τέλος, γίνεται παρουσίαση της περίπτωσης του Kleio, ενός δρομολογητή που ενσωματώνει τεχνικές μηχανικής μάθησης για τη βελτιστοποίηση της διαχείρισης των σελίδων.

2.1 Μη πτητικές μνήμες

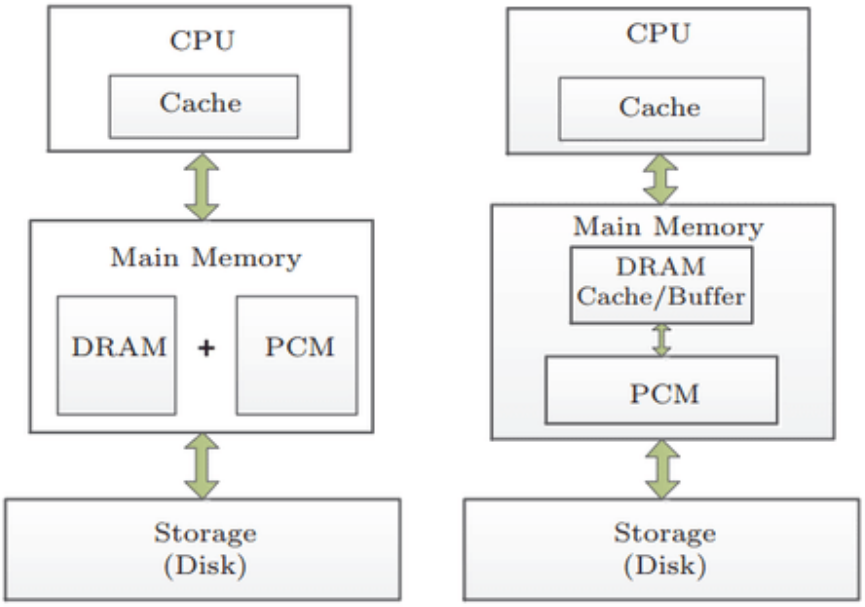
Η πλειονότητα των μνημών που χρησιμοποιούνται ως εναλλακτική ή συμπληρωματική επιλογή για τη DRAM εμπίπτουν στην κατηγορία των μη-πτητικών μνημών (Non-Volatile Memory, NVM). Η ονομασία τους προέρχεται από την ιδιότητά τους να διατηρούν τα δεδομένα ακόμη και σε περίπτωση διακοπής της ηλεκτρικής τροφοδοσίας. Πρόκειται για αρκετά ανθεκτικές μνήμες οι οποίες διαθέτουν μεγάλη συνολική χωρητικότητα λόγω αυξημένης πυκνότητας κελιών, ενώ παράλληλα παρουσιάζουν σχεδόν μηδενική κατανάλωση ενέργειας όσο βρίσκονται σε αδρανή κατάσταση. Από την άλλη, οι μη πτητικές μνήμες έχουν συνήθως μεγαλύτερους χρόνους προσπέλασης σε σύγκριση με τις DRAM και παρουσιάζουν ανισορροπία μεταξύ του χρόνου ανάγνωσης και εγγραφής, με τον δεύτερο να αποτελεί συχνά τον κύριο παράγοντα επιβάρυνσης. [1] Οι μνήμες NVM διακρίνονται σε διάφορες κατηγορίες, ανάλογα με τη βασική τους τεχνολογία. Η Spin-Torque Transfer RAM (STT-RAM) αξιοποιεί τη ροπή μεταφοράς spin για την αλλαγή καταστάσεων, ενώ η Resistive RAM (ReRAM) βασίζεται στη μεταβολή της αντίστασης για την αποθήκευση δεδομένων. Η Phase-Change Memory (PCM) χρησιμοποιεί υλικά αλλαγής φάσης που εναλλάσσονται μεταξύ άμορφης και κρυσταλλικής κατάστασης, και η Flash Memory αποθηκεύει δεδομένα μέσω ηλεκτρικής φόρτισης. Τέλος, η FeRAM επιτυγχάνει μη πτητικότητα χρησιμοποιώντας φερροηλεκτρικό στρώμα αντί για διηλεκτρικό, διαφοροποιούμενη από τις άλλες τεχνολογίες. [8]

Μια μη πτητική μνήμη μπορεί να ενσωματωθεί σε ένα υπολογιστικό σύστημα με δύο βασικούς τρόπους:

- **Οριζόντια ενσωμάτωση**: Η NVM λειτουργεί ως επέκταση της DRAM, αυξάνοντας τη συνολική χωρητικότητα του συστήματος και δημιουργώντας έναν διευρυμένο, ενιαίο χώρο φυσικών διευθύνσεων. Χάρη στη μη πτητικότητα, μπορεί να αξιοποιηθεί και

ως αποθηκευτικό μέσο από τις εφαρμογές, προσφέροντας μεγαλύτερη ευελιξία και επιπλέον δυνατότητες αποθήκευσης.

- **Κάθετη ενσωμάτωση**: Η NVM αναλαμβάνει τον ρόλο της κύριας μνήμης, αντικαθιστώντας τη DRAM. Στην περίπτωση αυτή, η DRAM λειτουργεί ως κρυφή μνήμη (cache) για τη NVM, μειώνοντας έτσι τον χρόνο προσπέλασης.



Σχήμα 2.1: *Υβριδικό σύστημα μνήμης με DRAM και PCM NVM. Στα αριστερά παρουσιάζεται η οριζόντια ενσωμάτωση, ενώ στα δεξιά η κάθετη.[1].*

2.2 Δρομολόγηση σελίδων

Σε τοπολογία κάθετης ενσωμάτωσης, οι εφαρμογές δεν έχουν επίγνωση της υβριδικής φύσης του συστήματος και λειτουργούν χωρίς να απαιτούνται τροποποιήσεις και βελτιστοποιήσεις, καθώς η ενσωμάτωση των μνημών πραγματοποιείται σε επίπεδο λειτουργικού συστήματος, δημιουργώντας μια ενιαία ιεραρχία μνήμης που υποστηρίζεται αυτόματα.

Αντίθετα, κατά τη σχεδίαση ενός υβριδικού συστήματος μνήμης σε οριζόντια ενσωμάτωση, μια από τις πιο σημαντικές προκλήσεις είναι η ανάπτυξη ενός κατάλληλου μηχανισμού για την τοποθέτηση και μεταφορά δεδομένων μεταξύ των μνημών, γνωστού ως δρομολογητής σελίδων (page scheduler). Η ονομασία του πηγάζει από τη σελιδοποίηση, μια τεχνική διαχείρισης μνήμης που εφαρμόζεται σχεδόν σε όλα τα σύγχρονα λειτουργικά συστήματα, σύμφωνα με την οποία η φυσική μνήμη χωρίζεται σε μικρές, σταθερού μεγέθους μονάδες που ονομάζονται σελίδες και η μεταφορά δεδομένων μεταξύ μνήμης και δίσκου γίνεται σε επίπεδο σελίδας. Στόχος ενός page scheduler είναι η συνεχής διατήρηση των πιο συχνά χρησιμοποιούμενων σελίδων (hot pages) μιας εφαρμογής στην γρήγορη μνήμη DRAM.

Η υλοποίηση ενός δρομολογητή σελίδων είναι μια σύνθετη διαδικασία που απαιτεί προσεκτική ανάλυση και λήψη κρίσιμων αποφάσεων για τη διασφάλιση της βέλτιστης απόδοσης του συστήματος. Ανάμεσα στις σημαντικότερες προκλήσεις περιλαμβάνονται η σχεδίαση κατάλληλης πολιτικής για τη διάκριση των συχνά χρησιμοποιούμενων σελίδων, ο καθορισμός

του επιπέδου υλοποίησης (πυρήνας, χώρος χρήστη, υλικό), η χρήση ενός αποδοτικού μηχανισμού συλλογής δεδομένων για τη συλλογή των απαραίτητων δεδομένων και η βελτιστοποίηση της διαδικασίας μεταφοράς σελίδων. Η σύντομη ανάλυση των παραπάνω ζητημάτων αποτελεί το βασικό αντικείμενο της ενότητας αυτής.

2.2.1 Πολιτική δρομολόγησης

Το βασικότερο ίσως χαρακτηριστικό ενός page scheduler είναι η πολιτική που χρησιμοποιεί για τον εντοπισμό των συχνά χρησιμοποιούμενων σελίδων. Πρόκειται ουσιαστικά για τα κριτήρια βάσει των οποίων το σύστημα διακρίνει τις 'hot pages' και αποφασίζει πότε είναι απαραίτητες οι μετακινήσεις σελίδων για τη βελτίωση της απόδοσης. Σε θεωρητικό επίπεδο, μια πολιτική είναι εν γένει ανεξάρτητη από τον μηχανισμό, δηλαδή από τη διαδικασία με την οποία πραγματοποιούνται οι δεσμεύσεις και οι μεταφορές των σελίδων.

Η πλειονότητα των σύγχρονων δρομολογητών σελίδας εμπίπτουν στην κατηγορία των δρομολογητών με βάση το ιστορικό (History page schedulers), οι οποίοι στηρίζονται σε ιστορικά δεδομένα για τη λήψη αποφάσεων σχετικά με την δέσμευση και τη μετακίνηση σελίδων. Το συγκεκριμένο είδος πολιτικής έχει τις ρίζες του στην υπόθεση της χρονικής τοπικότητας των δεδομένων, σύμφωνα με την οποία τα προγράμματα συχνά τείνουν να προσπελαύνουν επανειλημμένα τα ίδια δεδομένα σε σύντομα χρονικά διαστήματα. [9]. Σε έναν History page scheduler, ο χρόνος διαιρείται σε σταθερά διαστήματα που ονομάζονται περίοδοι δρομολόγησης (scheduling periods). Σε κάθε περίοδο, πραγματοποιούνται οι απαραίτητες ανακατατάξεις σελίδων προκειμένου οι σελίδες με τις περισσότερες προσβάσεις κατά την τρέχουσα περίοδο να παραμείνουν στη γρήγορη μνήμη (έως ότου αυτή φτάσει στο όριο της χωρητικότητας της) με την ελπίδα ότι θα παραμείνουν συχνά χρησιμοποιούμενες και στην επόμενη περίοδο [10].

Ωστόσο, έχουν ήδη αναπτυχθεί αρκετές παραλλαγές στην υλοποίηση της παραπάνω πολιτικής. Το HeMem [11], χρησιμοποιεί τρεις ουρές FIFO: cold, hot και free, για τη διαχείριση των σελίδων και διαθέτει ξεχωριστούς μετρητές για αναγνώσεις (loads) και εγγραφές (stores). Μια σελίδα χαρακτηρίζεται ως συχνά προσπελάσιμη όταν οι αναγνώσεις της υπερβαίνουν τις 8 ή οι εγγραφές της ξεπερνούν τις 4, δίνοντας έτσι προτεραιότητα στις εγγραφές, καθώς το χρονικό κόστος είναι σημαντικά υψηλότερο στις μη πτητικές μνήμες. Παράλληλα, εφαρμόζεται ένας μηχανισμός 'ψύξης', ο οποίος φροντίζει να μειώνει τις τιμές των μετρητών ανά τακτά χρονικά διαστήματα βάσει ενός ρολογιού.

Το Thermostat [12] έχει αντιστρέψει την οπτική γωνία του προβλήματος και επιχειρεί να εντοπίζει και να υποβαθμίζει τις λιγότερο συχνά προσπελάσιμες σελίδες. Ο χρήστης ορίζει ένα μέγιστο όριο αποδεκτής επιβράδυνσης και το σύστημα τοποθετεί τις σελίδες με τους χαμηλότερους ρυθμούς προσβάσεων στη βραδύτερη μνήμη ώστε να διατηρείται η συνολική απόδοση εντός του επιθυμητού ορίου. Παράλληλα διαθέτει και μηχανισμό εντοπισμού εσφαλμένων προβλέψεων που επιτρέπει σε συχνά προσπελάσιμες σελίδες που τοποθετήθηκαν εσφαλμένα στην αργή μνήμη να μετακινηθούν πίσω στη γρήγορη.

Το HeteroVisor [13] είναι ένα σύστημα διαχείρισης υβριδικών υπολογιστικών συστημάτων σε εικονικοποιημένα περιβάλλοντα που λαμβάνει υπόψη τόσο τις διαφορές στις μνήμες όσο και την ετερογένεια μεταξύ των επεξεργαστών. Αναφορικά με την πολιτική διαχείρισης των

σελίδων, το σύστημα παρακολουθεί περιοδικά τις προσβάσεις και κρίνει ότι μια σελίδα είναι σημαντική όταν οι προσβάσεις της υπερβαίνουν τις 22. Χρησιμοποιεί λίστα ως βασική δομή δεδομένων για την παρακολούθηση των σελίδων και σε κάθε περίοδο εξετάζει μόνο ένα υποσύνολο της λίστας αυτής ώστε να διασφαλιστεί η αποδοτικότητα της διαδικασίας. Επιπλέον, σελίδες που δεν έχουν προσπελαστεί για μεγάλο χρονικό διάστημα (3 sec) θεωρούνται ανενεργές, μέχρι να χρησιμοποιηθούν ξανά.

Ενδιαφέρουσα είναι και η προσέγγιση του Adaptive Page Migration (AMP) [14], το οποίο δεν δημιουργεί κάποιον νέο ευριστικό κανόνα για την διάκριση των σημαντικών σελίδων αλλά αναλύει τη συμπεριφορά των προσβάσεων μνήμης εισάγοντας δύο νέα πεδία (age, history) σε κάθε εγγραφή του πίνακα σελίδων και περιοδικά επιλέγει την καλύτερη πολιτική μεταξύ τριών διαθέσιμων: Least Recently Used (LRU), Least Frequently Used (LFU) και τυχαίας (Random). Αν υπάρχουν ενδείξεις χρονικής τοπικότητας, επιλέγεται είτε η πολιτική LRU είτε LFU, αλλιώς προτιμάται η τυχαία.

Τέλος, τόσο το Nimble [15] όσο και το Transparent Page Placement (TPP) [16] δεν επιβάλλουν κάποιο μηχανισμό παρακολούθησης σελίδων αλλα χρησιμοποιούν τις ήδη υπάρχουσες LRU λίστες που διατηρεί ο πυρήνας. Στην περίπτωση του Nimble, προκειμένου να αποφασιστεί αν μια σελίδα είναι συχνά χρησιμοποιούμενη, ελέγχονται δύο bit πρόσβασης: το hardware access bit (που ενημερώνεται από τον hardware page table walker) και το software access bit (που ενημερώνεται από τον ήδη υπάρχοντα Linux paging αλγόριθμο). Η βασική διαφοροποίηση έγκειται στο γεγονός ότι το Nimble αντί να αποδεσμεύει τις αδρανείς σελίδες ή να τις μεταφέρει σε δίσκο,τις μετακινεί στη βραδύτερη μνήμη. Η πολιτική του TPP χρησιμοποιεί και αυτή τις LRU λίστες του πυρήνα αλλά μόνο για τη μεταφορά αδρανών σελίδων από τη γρήγορη στην αργή μνήμη. Η μεταφορά σελίδων στην αντίθετη κατεύθυνση βασίζεται στη χρήση του εργαλείου AutoNUMA το οποίο θα περιγραφεί εν συντομία παρακάτω. Η πρωτοτυπία του συγκεκριμένου δρομολογητή είναι ότι παρέχει τη δυνατότητα να εστιάσει σε συγκεκριμένο είδος σελίδων (anonymous pages, file backed memory maps) ανάλογα με τη συμπεριφορά της κάθε εφαρμογής.

2.2.2 Συλλογή δεδομένων

Κάθε δρομολογητής οφείλει να παρακολουθεί προσεκτικά τη δραστηριότητα των σελίδων κατά την εκτέλεση των εφαρμογών, προκειμένου να συλλέγει τα δεδομένα που χρειάζονται για τον εντοπισμό του συνόλου των hot pages σε κάθε περίοδο. Υπάρχουν διάφοροι τρόποι υλοποίησης αυτής της διαδικασίας, εκ των οποίων οι ευρύτερα χρησιμοποιούμενοι είναι η δειγματοληψία με βάση μετρητές επίδοσης σε επίπεδο υλικού (hardware performance counters), οι τεχνικές δηλητηρίασης σελίδας (Page Poisoning), η αξιοποίηση του εργαλείου AutoNUMA και η χρήση των ήδη υπαρχόντων δομών του πυρήνα όπως οι LRU λίστες. Κάθε τεχνική συνοδεύεται από τα δικά της πλεονεκτήματα και περιορισμούς που αναλύονται παρακάτω.

Δειγματοληψία με μετρητές επίδοσης

Στην τεχνική της δειγματοληψίας, η πληροφορία σχετικά με τις προσπελάσεις της μνήμης πηγάζει απευθείας από το υλικό μέσω της μονάδας παρακολούθησης της επίδοσης (Perfor-

mance Monitoring Unit - PMU). Πρόκειται για μια ειδική μονάδα που περιλαμβάνεται σε όλους σχεδόν τους σύγχρονους επεξεργαστές και έχει σκοπό να καταγράφει διάφορα συμβάντα όπως αστοχίες κρυφής μνήμης (cache misses), αποτυχίες μετάφρασης εικονικών διευθύνσεων (TLB misses), καθυστερήσεις (stalls) κλπ. Μπορεί να χρησιμοποιηθεί είτε για απλή μέτρηση των γεγονότων (counting mode) είτε για δειγματοληπτική καταγραφή συμβάντων (precise sampling mode):

- Σε counting mode, η χρήση της PMU περιορίζεται στην καταμέτρηση του συνολικού αριθμού των γεγονότων που έχουν συμβεί κατά τη διάρκεια μιας περιόδου. Είναι ιδανική για τη συλλογή συνολικών στατιστικών αλλά δεν παρέχει επιμέρους λεπτομέρειες (π.χ ακριβή χρονική στιγμή του γεγονότος, διευθύνσεις μνήμης κ.λ.π), γεγονός που καθιστά δύσκολη τη χρήση του κατά τη δρομολόγηση σελίδων.

- Σε sampling mode, οι μετρητές της PMU δεν περιορίζονται στην καταμέτρηση του συνολικού αριθμού των συμβάντων, αλλά δίνουν τη δυνατότητα καταγραφής λεπτομερών στιγμιότυπων για ένα υποσύνολο αυτών. Κάθε τέτοιο στιγμιότυπο μπορεί να περιλαμβάνει κρίσιμες πληροφορίες όπως τη διεύθυνση της εντολής που εκτελούνταν, τις τιμές των καταχωρητών της ΚΜΕ, τη διεύθυνση μνήμης που προσπελάστηκε, καθώς και άλλα δεδομένα που αποτυπώνουν την κατάσταση του συστήματος εκείνη τη χρονική στιγμή.

Η διαδικασία της δειγματοληψίας υλοποιείται ως εξής: ορισμένοι ειδικοί καταχωρητές μετρούν τα ζητούμενα γεγονότα, ενώ άλλοι καταχωρητές αποθηκεύουν το πιο πρόσφατο δείγμα. Όταν η τιμή ενός μετρητή υπερβεί μια προκαθορισμένη τιμή, γνωστή ως sample period ή reset value, ένας καταχωρητής υπερχειλίζει, ενεργοποιώντας μια διακοπή (interrupt) και το λογισμικό (συνήθως το λειτουργικό σύστημα) αναλαμβάνει να συλλέξει ένα δείγμα [17]. Η μέθοδος αυτή, υποφέρει από ορισμένα μεγάλα προβλήματα. Πρώτον, τα δείγματα δεν είναι απολύτως ακριβή καθώς ακόμα και η παραμικρή καθυστέρηση μεταξύ της υπερχείλισης και της καταγραφής του δείγματος μπορεί να οδηγήσει σε εσφαλμένη πληροφορία, φαινόμενο που συνήθως ονομάζεται ολίσθηση (skid). Δεύτερον, οι μικροεντολές εκτελούνται παράλληλα και εκτός σειράς, ενώ ο δείκτης εντολών (instruction pointer) δείχνει την εντολή που θα συνεχιστεί η εκτέλεση και όχι την εντολή που προκάλεσε το συμβάν. Τέλος, αν η συχνότητα δειγματοληψίας αυξηθεί υπερβολικά, προκαλούνται αλλεπάλληλα interrupts, γεγονός που με τη σειρά του επιφέρει σημαντική επιβάρυνση στο σύστημα. Η λύση στα παραπάνω προβλήματα είναι η ενίσχυση του υλικού με σκοπό να παρέχει ακριβέστερη δειγματοληψία γεγονότων. Στους επεξεργαστές της Intel, αυτή η τεχνολογία είναι γνωστή ως Precise Event-Based Sampling (PEBS) και διακρίνεται για τη λεπτομέρεια που προσφέρει κατά τη συλλογή δεδομένων απόδοσης [18]. Η κύρια διαφορά του PEBS από τη συμβατική δειγματοληψία είναι ότι, αντί να ενεργοποιείται ένα interrupt σε κάθε υπερχείλιση των μετρητών, ένας μικροκώδικας αναλαμβάνει να καταγράψει τα δείγματα απευθείας σε μια κυκλική δομή αποθήκευσης που ονομάζεται κυκλικός πίνακας (ring buffer). Διακοπή δημιουργείται μόνο στην περίπτωση που ο ring buffer έχει γεμίσει πλήρως. [19, 20].

Θα μπορούσε κανείς να υποθέσει ότι η χρήση των PEBS έχει ελάχιστη ή μηδενική επίδραση στη συνολική απόδοση, ειδικά αν το μέγεθος του ring buffer είναι αρκετά μεγάλο. Ωστόσο, πρέπει να σημειωθεί ότι ο μικροκώδικας καταγράφει τα δείγματα χρησιμοποιώντας

την κρυφή μνήμη του επεξεργαστή, προκαλώντας έτσι σημαντική 'ρύπανση' της cache (cache pollution).

Αξίζει επίσης να σημειωθεί πως η υπερβολικά λεπτομερής δειγματοληψία ενδέχεται να προκαλέσει φαινόμενα throttling [21]. Πρόκειται για έναν μηχανισμό των σύγχρονων επεξεργαστών που μειώνει αυτόματα τη συχνότητα λειτουργίας όταν η θερμοκρασία υπερβεί ένα προκαθορισμένα όριο, με σκοπό την αποφυγή υπερθέρμανσης και την προστασία του υλικού. Παράλληλα, αποτελεί ένδειξη ότι το θερμικό φορτίο του συστήματος ξεπερνά τα ασφαλή όρια και χρήζει άμεσης αντιμετώπισης.

Page Poisoning

Κάθε εγγραφή στον πίνακα σελίδων συνοδεύεται από ένα bit που ονομάζεται bit πρόσβασης (access bit), το οποίο υποδεικνύει αν η σελίδα έχει προσπελαστεί πρόσφατα. Η τεχνική του Page Poisoning βασίζεται στην περιοδική εκκαθάριση του access bit και στην εκκαθάριση της προσωρινής μνήμης μετάφρασης διευθύνσεων (TLB). Έτσι, προκαλούνται σκόπιμα σφάλματα σελίδας (page faults), τα οποία, κατά τη διαχείρισή τους, ενεργοποιούν ξανά το access bit. Με αυτόν τον τρόπο, ο δρομολογητής σελίδων μπορεί να μετράει τα access bits κάθε σελίδας για να λάβει μια εικόνα σχετικά με τη συχνότητα προσπελάσεων. Το βασικό πλεονέκτημα της προσέγγισης αυτής είναι πως δεν απαιτεί χρήση εξειδικευμένου υλικού. Μπορεί να αποτελέσει μια καλή λύση όταν ο στόχος είναι ο εντοπισμός των λιγότερο συχνά χρησιμοποιούμενων σελίδων (cold pages) [22, 12]. Ωστόσο, η συχνή παρακολούθηση των bit πρόσβασης του πίνακα σελίδων οδηγεί σε απαγορευτική επιβάρυνση του συστήματος. Στο πλαίσιο του page scheduling, χρησιμοποιείται συνήθως σε συνδυασμό με τη χρήση μεγάλου μεγέθους σελίδων (huge pages).

AutoNUMA

Το AutoNUMA [23] (γνωστό και ως NUMA Balancing) είναι ένας μηχανισμός στον πυρήνα του Linux που στοχεύει στη βέλτιστη κατανομή των σελίδων σε συστήματα με NUMA αρχιτεκτονική. Για την παρακολούθηση των προσπελάσεων μνήμης, εφαρμόζεται η τεχνική του Page Poisoning, δηλαδή η περιοδική απενεργοποίηση των access bits στις εγγραφές των πινάκων σελίδων (PTEs), ώστε να προκληθεί page fault κατά την επόμενη προσπέλαση της σελίδας. Σε κάθε page fault καταγράφεται ο NUMA κόμβος από τον οποίο πραγματοποιήθηκε η πρόσβαση, επιτρέποντας στον πυρήνα να εντοπίζει μοτίβα απομακρυσμένων προσπελάσεων. Με βάση αυτή την πληροφορία, το AutoNUMA μπορεί είτε να μεταφέρει μια σελίδα στον τοπικό κόμβο του νήματος είτε να επανατοποθετήσει το ίδιο το νήμα σε άλλο κόμβο όπου βρίσκονται αρκετά από τα δεδομένα του, στοχεύοντας έτσι στη μείωση της καθυστέρησης προσπέλασης και στη βελτίωση της συνολικής απόδοσης του συστήματος.

Linux LRU lists

Ο πυρήνας του Linux χρησιμοποιεί δύο ειδικές δομές δεδομένων για να εντοπίζει σελίδες που μπορούν να απελευθερωθούν ή να μεταφερθούν στον δίσκο [24]: τη λίστα των ενεργών σελίδων, όπου τοποθετούνται οι σελίδες που χρησιμοποιήθηκαν πιο πρόσφατα και τη λίστα των ανενεργών σελίδων που συγκεντρώνει τις σελίδες που παραμένουν αδρανείς για κάποιο

χρονικό διάστημα. Κάθε φορά που μια σελίδα προσπελαύνεται, μετακινείται προς την αρχή της ενεργής λίστας, ενώ οι σελίδες που δεν χρησιμοποιούνται σταδιακά μεταφέρονται στην ανενεργή λίστα. Όταν το σύστημα χρειάζεται να απελευθερώσει μνήμη, επιλέγει να απομακρύνει (στον αποθηκευτικό χώρο) σελίδες που βρίσκονται στο τέλος της λίστας ανενεργών σελίδων, καθώς αυτές κρίνονται ως λιγότερο πιθανές να απαιτηθούν σύντομα. Με τον τρόπο αυτό, επιτυγχάνεται βέλτιστη αξιοποίηση του διαθέσιμου χώρου μνήμης. Ορισμένοι δρομολογητές [15, 16] αξιοποιούν αυτές τις λίστες για να εκτιμήσουν τη συχνότητα πρόσβασης των σελίδων. Αντί να τις μεταφέρουν σε δευτερεύοντα αποθηκευτικό χώρο, συνήθως επιλέγουν να τις ανακατανείμουν μεταξύ των διαφορετικών μνημών του υβριδικού συστήματος.

2.2.3 Επίπεδο υλοποίησης

Ακόμα μια σημαντική πρόκληση που εμφανίζεται κατά τη σχεδίαση ενός δρομολογητή σελίδων είναι η επιλογή του κατάλληλου επιπέδου υλοποίησης. Αν και συνήθως απαιτείται κάποια υποστήριξη από πλευράς υλικού, το κύριο μέρος της λειτουργικότητας ενός page scheduler υλοποιείται συνήθως σε επίπεδο λογισμικού. Το ερώτημα που προκύπτει όμως σε αυτές τις περιπτώσεις είναι αν ο δρομολογητής θα πρέπει να ενσωματωθεί στον πυρήνα ή να εκτελείται σε χώρο χρήστη. Όταν η πολιτική εντοπισμού των 'σημαντικών' σελίδων είναι σχετικά απλή, η υλοποίηση εντός του πυρήνα αποτελεί μια εξαιρετικά αποδοτική επιλογή, καθώς αξιοποιεί τις υπάρχουσες δομές δεδομένων του λειτουργικού συστήματος και αποφεύγει το κόστος των αλλεπάλληλων context switches. Ωστόσο, για δρομολογητές με πιο σύνθετες πολιτικές, όπως εκείνους που βασίζονται στη δειγματοληψία με PEBS, η υλοποίηση εντός του πυρήνα δεν ενδείκνυται λόγω των πολλαπλών περιορισμών. Για παράδειγμα, η χρήση των μετρητών επίδοσης, αν και υποστηρίζεται, δεν συνιστάται για διεργασίες που εκτελούνται εντός του πυρήνα, οι αριθμητικές πράξεις κινητής υποδιαστολής είναι εξαρχής απενεργοποιημένες και αρκετές βιβλιοθήκες έχουν σχεδιαστεί αποκλειστικά για userspace λειτουργία. Σε αυτές τις περιπτώσεις, ο χώρος χρήστη προσφέρει μεγαλύτερη ευελιξία και διευκολύνει τη χρήση ήδη υπαρχόντων εργαλείων και τεχνικών αλλά συνοδεύεται από την ανάγκη για δημιουργία και συντήρηση καινούριων δομών για την παρακολούθηση της συμπεριφοράς κάθε σελίδας [16].

2.2.4 Μηχανισμός μετακίνησης σελίδων

Μέχρι στιγμής έχει γίνει λόγος μόνο για τις πολιτικές με τις οποίες καθορίζεται αν μια σελίδα είναι σημαντική ή όχι. Ωστόσο, η δημιουργία ενός ολοκληρωμένου page scheduler θα ήταν αδύνατη χωρίς έναν μηχανισμό που να αναλαμβάνει την μετακίνηση των σελίδων από τη μια μνήμη στην άλλη. Σε επίπεδο χρήστη, η διεπαφή που παρέχει το Linux για την ανακατανομή των σελίδων είναι η κλήση συστήματος `move_pages()`. Η συγκεκριμένη συνάρτηση δέχεται ως ορίσματα μεταξύ άλλων έναν πίνακα από διευθύνσεις μνήμης και το αναγνωριστικό ενός NUMA κόμβου και επιχειρεί να μετακινήσει τις σελίδες των διευθύνσεων εκτελώντας τέσσερα συγκεκριμένα βήματα:

- Δεσμεύει χώρο στον κόμβο-προορισμό
- Ακυρώνει τις υπάρχουσες εγγραφές στον πίνακα σελίδων και στον TLB

- Μεταφέρει τα δεδομένα στη μνήμη προορισμό
- Απελευθερώνει τον παλιό χώρο των σελίδων

Οι περισσότερες σχεδιάσεις δρομολογητών βασίζονται στη χρήση της εν λόγω κλήσης συστήματος. Δυστυχώς όμως, η `move_pages()` διαθέτει πολλούς περιορισμούς που πρέπει να ληφθούν σοβαρά υπόψη προκειμένου να βελτιστοποιηθεί η επίδοση ενός δρομολογητή. Οι κυριότεροι εξ' αυτών είναι η αδυναμία της να πραγματοποιήσει αμφίδρομη μεταφορά δεδομένων (ανταλλαγή σελίδων), η έλλειψη παραλληλίας και η αδυναμία ομαδοποίησης μαζικών μετακινήσεων. Το Nimble [15] επιχειρεί να ξεπεράσει τα προβλήματα αυτά δημιουργώντας τη δική του εκδοχή της `move_pages()` εντός του πυρήνα.

2.3 Case study: Kleio

Η ενότητα αυτή είναι αφιερωμένη στην αναλυτική παρουσίαση του Kleio, [2] ενός καινοτόμου δρομολογητή σελίδων για συστήματα υβριδικής μνήμης, ο οποίος αποτέλεσε τη βασική πηγή έμπνευσης για τη συγκεκριμένη διπλωματική εργασία.

2.3.1 Επισκόπηση

Η πρωτοπορία του Kleio έγκειται στην εφαρμογή τεχνικών μηχανικής μάθησης για την αποδοτικότερη τοποθέτηση και αναδιάταξη των σελίδων μεταξύ των μνημών. Συγκεκριμένα, το Kleio στηρίζεται στην παρατήρηση ότι οι δρομολογητές με βάση το ιστορικό αδυνατούν να εκτιμήσουν σωστά τη συμπεριφορά των σελίδων όταν καταστρατηγείται η αρχή της χρονικής τοπικότητας. Δεν είναι λίγες οι φορές που μια σελίδα χρησιμοποιείται ξανά μετά από ένα μεγάλο διάστημα αδράνειας, γεγονός που δεν μπορεί να γίνει αντιληπτό από έναν History page scheduler. Για την αντιμετώπιση αυτού του φαινομένου, το Kleio χρησιμοποιεί μοντέλα μηχανικής μάθησης για να διαχειρίζεται ένα σύνολο σελίδων που θεωρούνται κρίσιμες και ονομάζονται σημαντικές σελίδες (important pages). Τα μοντέλα αυτά, χρησιμοποιώντας το ιστορικό προσβάσεων μιας σελίδας από προηγούμενες περιόδους δρομολόγησης, είναι σε θέση να προβλέψουν με ακρίβεια τον αριθμό των προσπελάσεων που αναμένονται στη σελίδα κατά την επόμενη περίοδο. Έτσι, σε κάθε περίοδο, ο δρομολογητής προσπαθεί να προβλέψει τον αριθμό προσβάσεων κάθε σελίδας στην επόμενη περίοδο, είτε με τη χρήση ενός μοντέλου (για τις σημαντικές σελίδες) είτε με βάση το ιστορικό προσβάσεων. Οι σελίδες με τον μεγαλύτερο αριθμό προβλεπόμενων προσπελάσεων μετακινούνται στη γρήγορη μνήμη μέχρι εξαντλήσεως της χωρητικότητας της ενώ οι υπόλοιπες καταλήγουν στην αργή.

2.3.2 Επιλογή σημαντικών σελίδων

Είναι λογικό πως η δημιουργία ενός μοντέλου μηχανικής μάθησης για κάθε σελίδα μιας εφαρμογής δεν αποτελεί αποδοτική και κλιμακώσιμη λύση, καθώς θα οδηγούσε σε υπερβολικούς χρόνους εκπαίδευσης και αλόγιστη σπατάλη πόρων. Επιπλέον, η χρήση μεγάλου αριθμού μοντέλων δεν συνεπάγεται απαραίτητα εμφανή βελτίωση στη συνολική επίδοση, αφού για ένα μεγάλο πλήθος σελίδων, ο History page scheduler εξακολουθεί να παρέχει ικανοποιητικά ακριβείς προβλέψεις. Για τον λόγο αυτό, στο Kleio, η δημιουργία μοντέλων

μηχανικής μάθησης περιορίζεται αποκλειστικά σε ένα υποσύνολο σελίδων που ο History scheduler αποτυγχάνει να διαχειριστεί αποτελεσματικά. Αυτό επιτυγχάνεται μέσω του Επιλογέα Σελίδων (Page Selector), ενός βασικού συστατικού του Kleio, που ταξινομεί τις σελίδες μιας εφαρμογής ανάλογα με τη σημασία τους. Η λειτουργία του βασίζεται στον υπολογισμό μιας μετρικής που ονομάζεται **όφελος (benefit)** και ορίζεται ως το γινόμενο των λανθασμένων τοποθετήσεων (misplacements) και του συνολικού αριθμού προσπελάσεων κάθε σελίδας ($benefit = misplacements \times total\ access\ counts$). Ως λανθασμένη τοποθέτηση ορίζεται η περίπτωση κατά την οποία, στην αρχή μιας περιόδου δρομολόγησης, η σελίδα θα έπρεπε να βρίσκεται στη γρήγορη μνήμη αλλά δεν είναι, εξαιτίας εσφαλμένης πρόβλεψης. Μέσω της μετρικής του οφέλους, ο Page Selector αποτιμά τη συνολική σημασία κάθε σελίδας, συνδυάζοντας τόσο τη συχνότητα προσπέλασής της από την εφαρμογή όσο και την αδυναμία του History scheduler να την προβλέψει σωστά. Τέλος, οι σελίδες ταξινομούνται σε φθίνουσα σειρά οφέλους και επιλέγεται ένα υποσύνολο αυτών για το οποίο εκπαιδεύονται μοντέλα μηχανικής μάθησης.

2.3.3 Μοντέλα μηχανικής μάθησης στο Kleio

Έχει ήδη αναφερθεί ότι το Kleio βασίζεται στη δημιουργία μοντέλων μηχανικής μάθησης ανά σελίδα. Τα μοντέλα αυτά, λαμβάνοντας ως είσοδο τον αριθμό προσπελάσεων μιας σελίδας κατά τις προηγούμενες περιόδους δρομολόγησης, προβλέπουν τον μελλοντικό αριθμό προσπελάσεων της (per page predictions). Η επιλογή αυτή αποτελεί βασική σχεδιαστική απόφαση, καθώς μια εναλλακτική προσέγγιση θα ήταν η πρόβλεψη των πιο σημαντικών σελίδων σε κάθε περίοδο (across pages predictions), με βάση το συνολικό αποτύπωμα μνήμης (memory access traces). Ωστόσο, η λύση αυτή απορρίφθηκε, αφενός λόγω του υπερβολικού κόστους εκπαίδευσης που προκύπτει από το μεγάλο μέγεθος των memory access traces, και αφετέρου λόγω της μειωμένης ακρίβειας, καθώς η πρόβλεψη γίνεται σημαντικά πιο δύσκολη όταν ο αριθμός των πιθανών στόχων (δηλαδή των διαφορετικών σελίδων) είναι πολύ μεγάλος.

Για την υλοποίηση της προσέγγισης αυτής, είναι κρίσιμο να επιλεγεί ο κατάλληλος τύπος μοντέλου μηχανικής μάθησης. Ανάμεσα στις διάφορες επιλογές που υπάρχουν, οι δημιουργοί του Kleio κατέληξαν στο συμπέρασμα πως τα **Αναδρομικά Νευρωνικά Δίκτυα** (Recurrent Neural Networks, RNN) φαίνεται να ταιριάζουν αρκετά καλά στις απαιτήσεις του προβλήματος. [25].

Αναδρομικά Νευρωνικά δίκτυα

Τα RNNs αποτελούν ένα είδος νευρωνικών δικτύων ικανά να επεξεργάζονται σειριακά δεδομένα (sequential data) όπως χρονοσειρές, κείμενα, ηχητικά δεδομένα, βιολογικές αλληλουχίες κλπ. Τα δεδομένα μιας τέτοιας ακολουθίας πρέπει να εμφανίζουν μεταξύ τους εξαρτήσεις. Για παράδειγμα, η σημασία μιας λέξης σε μια πρόταση μπορεί να καθορίζεται από τις προηγούμενες. Το στοιχείο που διακρίνει τα αναδρομικά δίκτυα από τα κλασικά Feedforward Neural Networks είναι ο μηχανισμός ανάδρασης, που επιτρέπει τη διατήρηση πληροφορίας από προηγούμενα χρονικά βήματα. Η επεξεργασία των στοιχείων γίνεται διαδοχικά, δηλαδή κάθε στοιχείο x_t εισάγεται όταν ολοκληρωθεί η επεξεργασία όλων των προηγούμενων εισόδων $x_1, x_2, \ldots, x_{t-1}$. Το δίκτυο διατηρεί μια εσωτερική κρυφή κατάστα-

ση (hidden state), η οποία συνοψίζει την πληροφορία από τα προηγούμενα δεδομένα και προσαρμόζεται δυναμικά κάθε φορά που εισάγεται ένα νέο στοιχείο.

Η εκπαίδευση ενός RNN γίνεται με μια παραλλαγή της μεθόδου οπισθοδρόμησης που ονομάζεται οπισθοδρόμηση στον χρόνο (backpropagation through time), σύμφωνα με την οποία το δίκτυο ξεδιπλώνεται σε κάθε χρονικό βήμα, δημιουργώντας μια ακολουθία αντιγράφων του, όπου οι παράμετροι (weights, biases) επαναλαμβάνονται. Το RNN μετατρέπεται έτσι σε ένα feedforward νευρωνικό δίκτυο στο οποίο μπορεί να εφαρμοστεί ο κλασικός αλγόριθμος οπισθοδρόμησης: Κατά τη διάρκεια της εκπαίδευσης, το δίκτυο επεξεργάζεται όλες τις ακολουθίες εισόδου, παράγοντας μια πρόβλεψη για καθεμία από αυτές. Η διαφορά ανάμεσα στις προβλέψεις του και στις πραγματικές τιμές αξιολογείται μέσω της συνάρτησης απώλειας (loss function). Ο ρυθμός μεταβολής αυτού του σφάλματος μεταφέρεται πίσω στο δίκτυο με βάση τον κανόνα της αλυσίδας, ώστε να ενημερωθούν οι παράμετροι του δικτύου [26, 27]

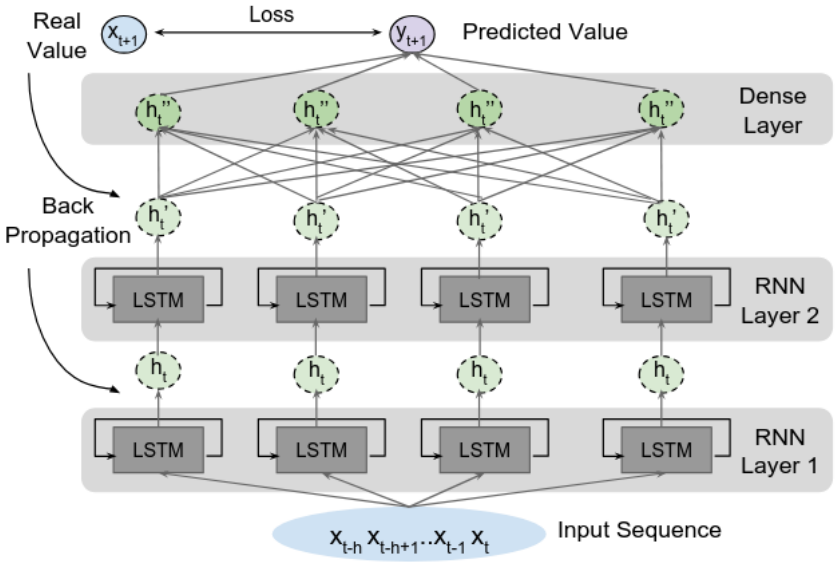
Long Short-Term Memory

Ίσως το πιο δημοφιλές είδος RNN είναι το Long Short-Term Memory (LSTM). Η δομική μονάδα ενός LSTM δικτύου αποτελείται από ένα κελί μνήμης (cell) που αντιπροσωπεύει την εσωτερική κατάσταση σε κάθε χρονική στιγμή και τρεις πύλες: εισόδου (input), εξόδου (output) και λήθης (forget). Το κελί έχει τη δυνατότητα να διατηρεί πληροφορίες για αυθαίρετα μεγάλα χρονικά διαστήματα ενώ οι πύλες ρυθμίζουν τη ροή των πληροφοριών από και προς το κελί. Η πύλη forget αποφασίζει ποιες πληροφορίες από την προηγούμενη κατάσταση θα διαγραφούν, η πύλη input επιτρέπει σε νέες πληροφορίες να αποθηκευτούν στην τρέχουσα κατάσταση και τέλος η πύλη output καθορίζει ποια στοιχεία της τρέχουσας κατάστασης της μνήμης θα χρησιμοποιηθούν για την έξοδο.

Αρχιτεκτονική μοντέλων και διαδικασία εκπαίδευσης στο Kleio

Στην περίπτωση του Kleio, κάθε μοντέλο αποτελείται από δύο διαδοχικά επίπεδα LSTM με 128 νευρώνες στο καθένα και στη συνέχεια ένα πυκνό επίπεδο το οποίο δίνει την τελική πρόβλεψη για τον αριθμό προσπελάσεων της σελίδας στην επόμενη περίοδο. Ως είσοδος δίνεται η ακολουθία όλων των προσπελάσεων μιας σελίδας σε κάθε περίοδο, η οποία, αφού κανονικοποιηθεί ώστε όλες οι τιμές να βρίσκονται στο διάστημα [0,1], διασπάται σε υπακολουθίες σταθερού μήκους (history length) ίσου με 16, με χρήση κυλιόμενου παραθύρου (rolling window fashion). Για τη διαδικασία της εκπαίδευσης το 70% του συνολικού συνόλου δεδομένων χρησιμοποιείται ως σύνολο εκπαίδευσης (training dataset), ενώ το υπόλοιπο 30% χρησιμοποιείται ως σύνολο επαλήθευσης (validation dataset). Ο ρυθμός εκμάθησης έχει οριστεί ίσος με 0.001 και ως συνάρτηση κόστους έχει χρησιμοποιηθεί το μέσο τετραγωνικό σφάλμα (mean squared error). Η διαδικασία εκπαίδευσης υποστηρίζεται επίσης από μηχανισμό early stopping, ώστε να αποφεύγεται η υπερεκπαίδευση (overfitting) και να εξοικονομούνται υπολογιστικοί πόροι. Συγκεκριμένα, η εκπαίδευση τερματίζεται πρόωρα εάν δεν παρατηρηθεί βελτίωση στο σφάλμα του validation dataset για 20 διαδοχικές εποχές. [28]. Τέλος, αξίζει να σημειωθεί πως η εκπαίδευση των μοντέλων γίνεται προς το παρόν εκτός χρόνου εκτέλεσης (offline), παρότι οι συντάκτες του Kleio προτείνουν την εκπαίδευση κατά

την εκτέλεση (online) ως μια ενδιαφέρουσα μελλοντική επέκταση.



Σχήμα 2.2: *Αρχιτεκτονική του νευρωνικού δικτύου στο Kleio [2].*

2.3.4 Αξιολόγηση επίδοσης

Ο βασικός στόχος του Kleio είναι να προσεγγίσει τη συμπεριφορά ενός ιδανικού δρομολογητή σελίδων (Oracle scheduler), ο οποίος γνωρίζει με ακρίβεια τη μελλοντική πρόσβαση κάθε σελίδας. Η αξιολόγηση του Kleio πραγματοποιείται σε περιβάλλον προσομοίωσης, όπου η αναλογία γρήγορης προς αργή μνήμη διαμορφώνεται στις τιμές 1/8, 1/16 και 1/32, ενώ η εκπαίδευση των PNN μοντέλων περιορίζεται σε μόλις 100 σημαντικές σελίδες. Υπό αυτές τις συνθήκες, το Kleio καταφέρνει να μειώσει **κατά 80%** το χάσμα απόδοσης μεταξύ ενός απλού History page scheduler και του ιδανικού Oracle, ενώ σε ορισμένες περιπτώσεις η βελτίωση αγγίζει έως και το **95%**. Όσον αφορά την ακρίβεια των προβλέψεων, το Kleio ενδέχεται να τοποθετήσει λανθασμένα κάποιες από τις επιλεγμένες σελίδες σε ορισμένες χρονικές περιόδους. Ωστόσο, τα λάθη αυτά είναι σπάνια και δεν αφορούν σελίδες με υψηλό αριθμό προσβάσεων, με αποτέλεσμα να μην επηρεάζεται ουσιαστικά το ποσοστό επιτυχιών στη DRAM. Συνολικά, το Kleio μειώνει κατά μέσο όρο το 85% των λανθασμένων τοποθετήσεων (misplacements) αυτών των σελίδων στη διάρκεια της εφαρμογής.

Μέρος II

Πρακτικό Μέρος

Κεφάλαιο 3

Περιγραφή υλοποίησης

Το κεφάλαιο αυτό περιλαμβάνει μια λεπτομερή περιγραφή της υλοποίησης του δρομολογητή. Αρχικά γίνεται μια σύντομη αναφορά στη λειτουργία του δρομολογητή και στις βασικές παραδοχές που υιοθετήθηκαν. Στη συνέχεια περιγράφονται τα τεχνικά χαρακτηριστικά του περιβάλλοντος καθώς και τα κύρια εργαλεία που χρησιμοποιήθηκαν. Τέλος, αναλύονται διεξοδικά οι κύριες συνιστώσες του λογισμικού.

3.1 Επισκόπηση

Η ενότητα αυτή ξεκινά με μια συνοπτική περιγραφή της λειτουργίας του δρομολογητή, η οποία περιλαμβάνει δύο βασικές φάσεις: τη φάση εκπαίδευσης (offline) και τη φάση πρόβλεψης (online). Κάθε εφαρμογή υποβάλλεται πρώτα στην φάση εκπαίδευσης, όπου εκτελείται μια φορά χρησιμοποιώντας αποκλειστικά δρομολόγηση με βάση το ιστορικό (History page scheduler) ενώ παράλληλα καταγράφονται οι προσπελάσεις κάθε σελίδας ανά περίοδο. Με την ολοκλήρωση της εκτέλεσης, τα συλλεγμένα δεδομένα μεταφέρονται στον Page Selector, ο οποίος επιλέγει το σύνολο των σημαντικών σελίδων. Στη συνέχεια, για κάθε σημαντική σελίδα εκπαιδεύεται και αποθηκεύεται ένα μοντέλο LSTM, με βάση τα καταγεγραμμένα δεδομένα προσβάσεων. Όλες οι μελλοντικές εκτελέσεις της συγκεκριμένης εφαρμογής αξιοποιούν τη φάση πρόβλεψης του δρομολογητή, κατά την οποία τα εκπαιδευμένα μοντέλα ενεργοποιούνται σε κάθε περίοδο για να προβλέψουν τις επόμενες προσβάσεις των σημαντικών σελίδων, ενώ για τις υπόλοιπες η πρόβλεψη ταυτίζεται με τον πραγματικό αριθμό προσβάσεων κατά την πιο πρόσφατη περίοδο.

3.2 Περιβάλλον υλοποίησης και εργαλεία

3.2.1 Αρχιτεκτονική και τοπολογία

Το σύστημα είναι εξοπλισμένο με δύο επεξεργαστές Intel Xeon Gold 5218T σε αρχιτεκτονική dual-socket. Κάθε επεξεργαστής διαθέτει 16 πυρήνες, με υποστήριξη 2 νημάτων ανά πυρήνα, προσφέροντας συνολικά 64 λογικούς επεξεργαστές.

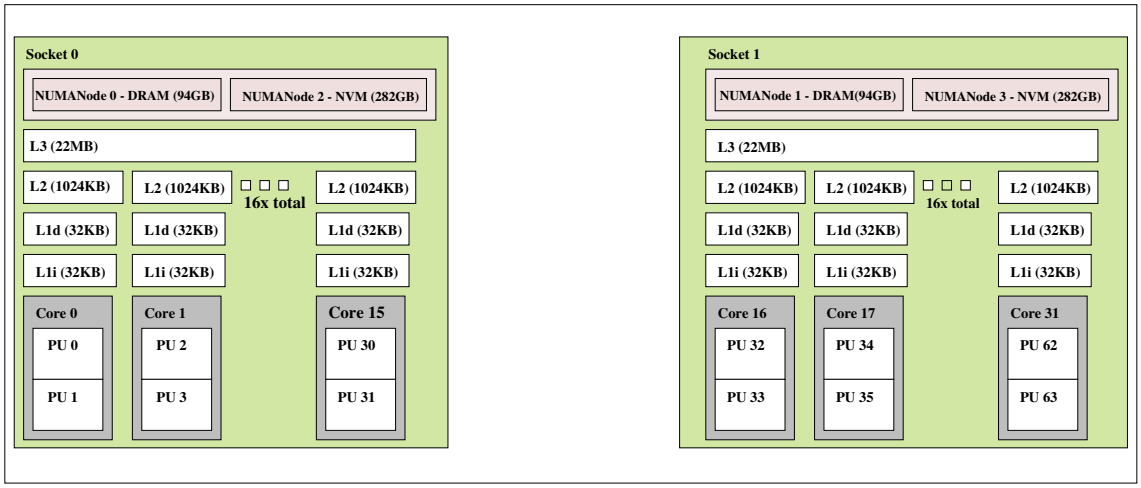
Το υβριδικό σύστημα μνήμης αποτελείται 6x32 GiB DDR DIMMs και 6x128 GB NVDIMMs, άρα σε κάθε socket αντιστοιχούν περίπου 96GiB γρήγορης μνήμης και 384GB NVM. Η μη πτητική μνήμη είναι Intel Optane DC, η οποία βασίζεται στην τεχνολογία 3D XPoint, επιτρέποντας προσβάσεις σε επίπεδο byte με καθυστερήσεις παρόμοιες με αυτές της DRAM

(346 ns). Διαθέτει ασύμμετρο εύρος ζώνης, με μέγιστη ταχύτητα ανάγνωσης 6.6 GB/s και εγγραφής 2.3 GB/s. Με δυνατότητα αποθήκευσης έως και 3 TB μνήμης ανά επεξεργαστή, αποτελεί ιδανική λύση για συστήματα με αυξημένες απαιτήσεις μνήμης και αποθήκευσης. Υποστηρίζει δύο τρόπους λειτουργίας: Memory Mode και App Direct Mode, οι οποίοι αντιστοιχούν στην κάθετη και οριζόντια ενσωμάτωση που αναφέρθηκε στο προηγούμενο κεφάλαιο. Τα NVDIMMs είναι χωρισμένα σε δύο περιοχές (regions) όπου κάθε περιοχή περιλαμβάνει 3 NVDIMMs. Για κάθε περιοχή έχει δημιουργηθεί ένας ονοματοχώρος (namespace) μεγέθους 282 GB. Ένας ονοματοχώρος είναι στην ουσία ένα σύνολο συνεχόμενων διευθύνσεων της NVM που είναι διαθέσιμο στο λειτουργικό σύστημα ως μια ειδική συσκευή κάτω από το */dev*. Κάθε ονοματοχώρος μπορεί να οριστεί σε έναν εκ των εξής τεσσάρων διαφορετικούς τρόπους λειτουργίας: sector, fsdax, devdax και raw mode:

- **sector mode**: Η μνήμη παρουσιάζεται ως μια συσκευή block (block device), ιδανική για παλαιότερες εφαρμογές που ενδιαφέρονται να χρησιμοποιούν την NVM ως αποθηκευτικό χώρο.
- **fsdax mode**: επιτρέπει στις εφαρμογές να έχουν άμεση πρόσβαση στη μνήμη μέσω ενός file system που υποστηρίζει (Direct Access mode - DAX). Όταν μια εφαρμογή απεικονίζει ένα αρχείο στη μνήμη (με την κλήση συστήματος `mmap()`), το σύστημα παρακάμπτει την τυπική αλυσίδα λογισμικού που χρησιμοποιεί το λειτουργικό σύστημα για να διαχειρίζεται δεδομένα σε αποθηκευτικές συσκευές (block device drivers, io schedulers κ.λ.π)
- **devdax mode**: επιτρέπει άμεση πρόσβαση στη μνήμη χωρίς τη χρήση συστήματος αρχείων, αντιμετωπίζοντας την NVM ως συσκευή χαρακτήρων (character device). Είναι η κατάλληλη επιλογή για το σύστημά μας όπου η ταχύτητα πρόσβασης παίζει καθοριστικό ρόλο.
- **raw mode**: Αυτός ο τρόπος λειτουργίας παρουσιάζει σοβαρούς περιορισμούς και δεν συνιστάται.

Ακόμα, οι δύο ονοματοχώροι πρέπει να ρυθμιστούν σε τρόπο λειτουργίας system-ram ώστε οι μη πτητικές μνήμες να φαίνονται ως ξεχωριστοί NUMA κόμβοι. Όλες οι ρυθμίσεις που αφορούν τον χειρισμό των μη πτητικών μνημών γίνονται με το εργαλείο *ndctl*. Συνοψίζοντας, το σύστημα αποτελείται από 2 NUMA nodes, κάθε ένα εκ των οποίων περιλαμβάνει 32 λογικούς επεξεργαστές και από 2 NUMA nodes χωρίς επεξεργαστές (cpu-less NUMA nodes), τα οποία αντιστοιχούν στις NVM [29].Το λειτουργικό σύστημα του συστήματος είναι Linux με πυρήνα έκδοσης 5.15.0-122-generic.

Η τοπολογία του συστήματος απεικονίζεται στην παρακάτω εικόνα (3.1).



Σχήμα 3.1: *Τοπολογία του συστήματος.*

3.2.2 Το εργαλείο perf

Όπως έχει ήδη αναφερθεί, η συλλογή δεδομένων σχετικά με τις προσπελάσεις σελίδων σε κάθε περίοδο γίνεται με χρήση δειγματοληψίας PEBS. Αυτό επιτυγχάνεται μέσω του perf [30], ένα εργαλείο που παρέχει το Linux ικανό να αλληλεπιδρά με τους performance counters του υλικού. Ο χρήστης μπορεί να επιλέξει τα events που τον ενδιαφέρουν και να προσαρμόσει τις παραμέτρους της δειγματοληψίας σύμφωνα με τις ανάγκες του προβλήματος. Στην πλειονότητα των περιπτώσεων, η χρήση του perf γίνεται μέσω command line διεπαφής. Ωστόσο, επειδή η υλοποίηση του δρομολογητή απαιτεί πιο λεπτομερή έλεγχο της δειγματοληψίας, χρησιμοποιήθηκε απευθείας το perf_events API που επιτρέπει την διαχείριση του PEBS μέσα από κώδικα C++ και το οποίο αποτελεί τον πυρήνα του cli εργαλείου. Ανάμεσα στα διάφορα events που προσφέρονται από τον επεξεργαστή, επιλέχθηκαν τα *mem_load_l3_miss_retired.local_dram* και *mem_load_retired.local_pmm* τα οποία καταγράφουν τις αναγνώσεις μνήμης που οδήγησαν σε αστοχία και των τριών επιπέδων κρυφής μνήμης και εξυπηρετήθηκαν από την dram ή την nvm αντίστοιχα. Στο σημείο αυτό πρέπει να σημειωθεί ότι η υλοποίηση δεν λαμβάνει υπόψη της τις εγγραφές. Αυτή η παραδοχή έγινε διότι δεν υπάρχει κάποιο event για εγγραφές που οδηγούν σε αστοχία του τελευταίου επιπέδου κρυφής μνήμης, το οποίο να παρέχει πληροφορία για το είδος μνήμης (DRAM, NVM) όπου ανήκει η διεύθυνση προορισμού. Αυτό συμβαίνει πιθανότατα επειδή στους σύγχρονους επεξεργαστές, οι εγγραφές θεωρούνται ολοκληρωμένες μόλις ολοκληρωθεί η μετάφραση διεύθυνσης στον TLB. Ως εκ τούτου, έμφαση στη συγκεκριμένη μελέτη δίνεται αποκλειστικά στις αναγνώσεις.

Οι σημαντικότερες παράμετροι που πρέπει να ρυθμιστούν προκειμένου η δειγματοληψία να είναι αποδοτική είναι η περίοδος δειγματοληψίας (sample period) και το μέγεθος του ring buffer. Η περίοδος δειγματοληψίας μετράται σε δείγματα (samples) και υποδηλώνει τον αριθμό των συμβάντων έως ότου προκύψει υπερχείλιση του μετρητή και εκτελεστεί ο μικροκώδικας του PEBS. Η διαδικασία προσδιορισμού της βέλτιστης τιμής για την περίοδο δειγματοληψίας θα παρουσιαστεί αργότερα σε αυτή την ενότητα. Σχετικά με το μέγεθος του ring buffer, ο μοναδικός περιορισμός είναι ότι πρέπει να είναι της μορφής $(2^n + 1)$*pa-

gesize (όπου pagesize το μέγεθος σελίδας που χρησιμοποιεί το λειτουργικό σύστημα). Σε κάθε δείγμα που συλλέγεται, καταγράφεται αποκλειστικά η διεύθυνση μνήμης στην οποία επιχειρείται η προσπέλαση (PERF_SAMPLE_ADDR), ενώ δείγματα που αφορούν εντολές του πυρήνα παραλείπονται. Αν και υπάρχει η δυνατότητα ομαδοποίησης των events, επιλέχθηκε να παραμείνουν ανεξάρτητα, δηλαδή οι καταγραφές από κάθε event να αποθηκεύονται σε ξεχωριστούς ring buffers για να διευκολυνθεί η διαδικασία συλλογής και να μην δημιουργηθούν προβλήματα συγχρονισμού.

3.2.3 Ρυθμίσεις συστήματος

Για τη σωστή λειτουργία της δειγματοληψίας, τη διαχείριση των μη πτητικών μνημών και την εφαρμογή των μοντέλων, απαιτούνται ορισμένες πρόσθετες ρυθμίσεις στο σύστημα. Μια εκ των σημαντικότερων αφορά την απενεργοποίηση του μηχανισμού τυχαίας διάταξης του χώρου διευθύνσεων (Address Space Layout Randomization, ASLR) [31]. Πρόκειται για μια ευρέως χρησιμοποιούμενη τεχνική ασφαλείας, κατά την οποία τα τμήματα της μνήμης μιας μια διεργασία (text, heap, stack, libraries) τοποθετούνται σε τυχαίες εικονικές διευθύνσεις κάθε φορά που εκτελείται το πρόγραμμα. Με αυτόν τον τρόπο αποτρέπονται επιθέσεις που βασίζονται στη γνώση της ακριβούς θέσης των δεδομένων στη μνήμη. Παρόλο που η απενεργοποίηση του ASLR δεν θεωρείται καλή πρακτική, είναι απαραίτητη στη συγκεκριμένη περίπτωση, καθώς οι εικονικές διευθύνσεις πρέπει να παραμένουν σταθερές τόσο κατά την offline όσο και κατά την online φάση. Επιπλέον, το Linux θέτει περιορισμούς σχετικά με την χρήση PEBS σε μη προνομιούχους χρήστες (χωρίς την ικανότητα CAP_SYS_ADMIN). Συνεπώς είναι αναγκαίο να ρυθμιστεί η παράμετρος perf_event_paranoid [32] στην τιμή -1, ώστε να επιτρέπεται η χρήση όλων των events χωρίς περιορισμούς. Ακόμη, ιδιαίτερη σημασία έχει η απενεργοποίηση του μηχανισμού AutoNUMA, καθώς οι μετακινήσεις σελίδων πρέπει να ελέγχονται αποκλειστικά από τον δρομολογητή και όχι από άλλους μηχανισμούς του πυρήνα. Τέλος, προκειμένου να μην υπάρχουν ανεπιθύμητες αλλοιώσεις κατά τις μετρήσεις, όλοι οι μηχανισμοί προανάκλησης (prefetchers) απενεργοποιήθηκαν.

3.3 Περιγραφή υλοποίησης

Στην ενότητα αυτή περιγράφεται αναλυτικά ο τρόπος λειτουργίας του δρομολογητή και οι κύριες συνιστώσες του.

3.3.1 Σχεδιαστικές επιλογές και παραδοχές

Όπως αναφέρθηκε στην προηγούμενη ενότητα, δεν είναι λίγες οι αποφάσεις που πρέπει να ληφθούν κατά τη σχεδίαση ενός δρομολογητή σελίδων, οι σημαντικότερες εκ των οποίων αφορούν το επίπεδο υλοποίησης, τον μηχανισμό μετακίνησης σελίδων και τη συλλογή δεδομένων σχετικά με τις προσβάσεις μνήμης. Επιπλέον, ορισμένες δυσκολίες που προκύπτουν κατά τη μετάβαση σε ένα πραγματικό σύστημα καθιστούν αναγκαία την υιοθέτηση σημαντικών παραδοχών.

Πολιτικές δρομολόγησης

Η υλοποίηση υποστηρίζει τις εξής τέσσερις πολιτικές δρομολόγησης:

- **Kleio**: Ακολουθεί πλήρως τη λογική του Kleio, όπως περιγράφεται στο Κεφάλαιο 2. Συνοπτικά, σε κάθε περίοδο προβλέπεται ο αριθμός μελλοντικών προσπελάσεων για κάθε σελίδα, είτε με χρήση μοντέλου LSTM (αν η σελίδα θεωρείται σημαντική), είτε βάσει των προσβάσεων της προηγούμενης περιόδου.

- **History**: Αντιστοιχεί σε έναν κλασικό History page scheduler, όπου σε κάθε περίοδο η προβλεπόμενη τιμή προσπελάσεων για κάθε σελίδα ισούται με τον αριθμό προσβάσεων που καταγράφηκαν στην αμέσως προηγούμενη περίοδο.

- **History Record**: Παρόμοια με την πολιτική History, αλλά με επιπλέον δυνατότητα καταγραφής όλων των δειγμάτων σε αρχείο. Κάθε γραμμή περιλαμβάνει μια εικονική διεύθυνση και τον τύπο της μνήμης (DRAM ή NVM), ενώ ειδικές εγγραφές σηματοδοτούν το τέλος κάθε περιόδου.

- **Oracle**: Σε κάθε περίοδο δρομολόγησης υπολογίζεται εκ των υστέρων η επίδοση που θα πετύχαινε ένας ιδανικός δρομολογητής (Oracle scheduler), αν γνώριζε εξαρχής τις πραγματικές προσβάσεις που θα πραγματοποιούνταν. Για τον λόγο αυτό, στα πειραματικά αποτελέσματα που ακολουθούν, η πολιτική Oracle χρησιμοποιείται ως μέτρο σύγκρισης, χωρίς όμως να συνιστά μια αυτόνομη πολιτική δρομολόγησης.

Συλλογή δεδομένων

Η ακριβής **δειγματοληψία** (με χρήση του Intel PEBS) κρίθηκε ως η πλέον κατάλληλη μέθοδος για τη συλλογή πληροφοριών σχετικά με τις προσβάσεις μνήμης. Οι υπάρχουσες λίστες του πυρήνα αφενός δεν μπορούν να αξιοποιηθούν στον χώρο χρήστη και αφετέρου δεν παρέχουν την απαιτούμενη ακρίβεια και πληρότητα για την εκπαίδευση των μοντέλων. Από την άλλη, οι μέθοδοι δηλητηρίασης (Page Poisoning) θα επέφεραν σημαντική επιβάρυνση, διότι η εξασφάλιση της απαιτούμενης ακρίβειας στην καταμέτρηση των προσπελάσεων θα απαιτούσε τη συχνή εκκαθάριση των access bits στις εγγραφές του πίνακα σελίδων. Αυτό θα οδηγούσε σε αυξημένο αριθμό page faults, επηρεάζοντας δραματικά την απόδοση του συστήματος.

Επίπεδο υλοποίησης και μηχανισμός μετακίνησης σελίδων

Ο δρομολογητής εκτελείται ως διεργασία σε **χώρο χρήστη**. Οι βασικότεροι λόγοι για αυτή την επιλογή είναι η απλότητα στην υλοποίηση και η δυνατότητα αξιοποίησης έτοιμων εργαλείων και βιβλιοθηκών. Επιπλέον, η χρήση τεχνικών μηχανικής μάθησης δεν θεωρείται ακόμη κατάλληλη για εφαρμογή εντός του πυρήνα λόγω των πολλαπλών περιορισμών του. Η παραμικρή λανθασμένη πρόβλεψη για μια θέση μνήμης μπορεί να έχει σοβαρές επιπτώσεις, αφού ο πυρήνας δεν παρέχει προστατευτικούς μηχανισμούς σε τέτοιες περιπτώσεις.

Η μεταφορά των σελίδων πραγματοποιείται μέσω της κλήσης συστήματος `move_pages()`. Δεν επιχειρήθηκε η χρήση ή ανάπτυξη κάποιας βελτιωμένης εκδοχής της, καθώς η έμφαση

δίνεται πρωτίστως στην επαλήθευση της ορθής λειτουργίας του Kleio. Οποιαδήποτε βελτιστοποίηση στον μηχανισμό μετακίνησης σελίδων μπορεί να ενσωματωθεί εύκολα ως μελλοντική επέκταση, χωρίς να απαιτούνται τροποποιήσεις στην υπόλοιπη υλοποίηση.

Μηχανισμός παύσης και επανεκκίνησης εφαρμογών

Η βασικότερη σχεδιαστική διαφοροποίηση της παρούσας υλοποίησης σε σχέση με άλλους δρομολογητές έγκειται στο γεγονός ότι η εκτέλεση της εφαρμογής διακόπτεται και επανεκκινείται κατόπιν εντολής του δρομολογητή, με χρήση των σημάτων SIGSTOP και SIGCONT αντίστοιχα. Η συγκεκριμένη προσέγγιση ισοδυναμεί με την παραδοχή ότι η διαδικασία της δρομολόγησης εκτελείται ακαριαία (σε μηδενικό χρόνο) και επιλέχθηκε για να εξασφαλιστεί καλύτερη ακρίβεια κατά τη μέτρηση της επίδοσης. Ειδικότερα, ο χρόνος που απαιτείται για τον υπολογισμό των προβλέψεων μέσω των μοντέλων σε κάθε περίοδο είναι αρκετά μεγάλος. Για παράδειγμα, ο καθορισμός των προβλέψεων για 200 σελίδες μπορεί να διαρκέσει έως και 10 δευτερόλεπτα. Εάν η εφαρμογή συνεχίσει να εκτελείται κατά τη διάρκεια αυτού του υπολογισμού, η δρομολόγηση θα βασίζεται σε παρωχημένα δεδομένα. Στην παρούσα υλοποίηση, για λόγους απλότητας, οι προβλέψεις πραγματοποιούνται σειριακά και χωρίς την αξιοποίηση εξειδικευμένου υλικού, γεγονός που συνεπάγεται αυξημένους χρόνους υπολογισμού. Σε μια πλήρως λειτουργική και πρακτικά εφαρμόσιμη υλοποίηση, ο δρομολογητής δεν θα πρέπει να παρεμβαίνει στην εκτέλεση της εφαρμογής, καθώς κάτι τέτοιο παραβιάζει τη φυσιολογική ροή της επεξεργασίας και εισάγει ανεπιθύμητη επιβάρυνση. Συνεπώς, η διαδικασία της δρομολόγησης θα πρέπει να εκτελείται στο παρασκήνιο και να ολοκληρώνεται με ελάχιστη καθυστέρηση, χωρίς να επηρεάζει τη λειτουργία της εφαρμογής. Προς αυτή την κατεύθυνση, η ενσωμάτωση επιταχυντών όπως οι GPUs ή οι TPUs, καθώς και η αξιοποίηση παραλληλίας σε επίπεδο σελίδας, θα μπορούσαν να μειώσουν δραστικά τον χρόνο εκτέλεσης των προβλέψεων. Οι GPUs, λόγω της υψηλής τους ικανότητας για ταυτόχρονη εκτέλεση χιλιάδων νημάτων, είναι ιδιαίτερα κατάλληλες για την παράλληλη επεξεργασία προβλέψεων για μεγάλο αριθμό σελίδων. Ακόμα καλύτερη επιλογή θα αποτελούσαν οι TPUs καθώς πρόκειται για εξειδικευμένους επιταχυντές σχεδιασμένους ειδικά για εργασίες μηχανικής μάθησης.

Κβαντοποίηση των προσπελάσεων μνήμης

Η λειτουργία αυτή υιοθετήθηκε από την υπάρχουσα υλοποίηση του Kleio, στην οποία οι προσπελάσεις μνήμης κάθε σελίδας ανά εποχή είναι κβαντισμένες. Αυτό σημαίνει πως το εύρος των προσπελάσεων μιας σελίδας (0 - μέγιστος αριθμός προσπελάσεων) χωρίζεται σε διαστήματα (bins) σταθερού μήκους (bin size) και κάθε αριθμός προσπελάσεων κατατάσσεται στο αντίστοιχο διάστημα. Η τεχνική αυτή είναι χρήσιμη καθώς μειώνει τις απαιτήσεις του δρομολογητή για ακριβείς προβλέψεις. Στόχος δεν είναι να προβλεφθεί με ο ακριβής αριθμός προσπελάσεων στην επόμενη περίοδο, αλλά να εξασφαλιστεί ότι οι προσπελάσεις θα αντιστοιχίζονται στο ίδιο bin. Για παράδειγμα, αν το εύρος τιμών προσπελάσεων είναι 0–100 και το bin size έχει οριστεί σε 20, τότε μια σελίδα με 35 προσπελάσεις θα καταταχθεί στο δεύτερο διάστημα (20–39), και οποιαδήποτε πρόβλεψη εντός του ίδιου διαστήματος θεωρείται επιτυχής. Αν και παρέχεται η δυνατότητα κβαντοποίησης, στην πράξη δεν αξιοποιείται ουσιαστικά στο πλαίσιο της παρούσας εργασίας (ή ισοδύναμα το bin size είναι 1).

Αναλογία μεγέθους μεταξύ γρήγορης και αργής μνήμης

Όπως θα φανεί και στη συνέχεια, τα μετροπρογράμματα που επιλέχθηκαν για την αξιολόγηση της επίδοσης δεν εξαντλούν τη διαθέσιμη χωρητικότητα της μνήμης. Προκειμένου να δημιουργηθούν πιο ρεαλιστικές συνθήκες, το μέγεθος της διαθέσιμης γρήγορης μνήμης περιορίζεται εικονικά από τον δρομολογητή. Συγκεκριμένα, για κάθε benchmark ορίζεται μια αναλογία (ratio) που καθορίζει το ποσοστό των συνολικών σελίδων της εφαρμογής που μπορούν να διατηρηθούν στη γρήγορη μνήμη. Κάτι τέτοιο προϋποθέτει την καταγραφή του συνολικού αριθμού σελίδων που χρησιμοποιεί κάθε μετροπρόγραμμα. Στην πλειονότητα των περιπτώσεων, η αναλογία αυτή είναι ίση με 1/8, δηλαδή η DRAM καλύπτει μόνο το 12.5% των συνολικών απαιτήσεων μνήμης της εφαρμογής.

Ανεξαρτησία μεγέθους σελίδας

Η υλοποίηση του δρομολογητή είναι συμβατή με οποιοδήποτε μέγεθος σελίδας. Ωστόσο, όπως θα φανεί σε επόμενο κεφάλαιο, η επιλογή του μεγέθους σελίδας αποτελεί κρίσιμη απόφαση που επηρεάζει άμεσα τη λειτουργικότητα του δρομολογητή.

Υποστήριξη πολυνηματικών εφαρμογών

Η σχεδίαση του δρομολογητή στην παρούσα εργασία εστιάζει κυρίως σε μονονηματικές (single-threaded) εφαρμογές. Παρόλο που η υλοποίηση υποστηρίζει και πολυνηματικές (multi-threaded) εφαρμογές, αυτό προϋποθέτει ότι όλα τα νήματα εκτελούνται στον ίδιο πυρήνα. Ωστόσο, η επέκταση για πλήρη υποστήριξη πολυνηματικών εφαρμογών —χωρίς τον περιορισμό του κοινού πυρήνα— είναι σχετικά απλή, καθώς απαιτεί μόνο τη ρύθμιση της δειγματοληψίας σε κάθε επεξεργαστή που χρησιμοποιείται από την εφαρμογή.

Ρύθμιση παραμέτρων ανά εφαρμογή

Οι περισσότερες παράμετροι της υλοποίησης πρέπει να προσαρμόζονται στις ανάγκες κάθε εφαρμογής χωρίς να απαιτούνται αλλαγές στον πυρήνα της υλοποίησης. Για τον λόγο αυτό, αποθηκεύονται σε ξεχωριστό αρχείο ρυθμίσεων (configurations file). Αναλυτικά, οι παράμετροι που μπορούν να ρυθμιστούν σε επίπεδο εφαρμογής περιλαμβάνουν:

- Περίοδος δειγματοληψίας
- Περίοδος δρομολόγησης
- Παράμετροι που σχετίζονται με τη δομή και την εκπαίδευση των μοντέλων, όπως ο αριθμός επιπέδων (layers), το μήκος της εισόδου (history length) και ο ρυθμός εκμάθησης (learning rate).
- Μέγιστος αριθμός σημαντικών σελίδων (ή ισοδύναμα μέγιστος αριθμός μοντέλων που θα δημιουργηθούν)
- Αναλογία χωρητικότητας μνημών
- Πολιτική δρομολόγησης

- Μέγεθος bin size

3.4 Λειτουργία δρομολογητή σε πολιτική Kleio

Η ενότητα αυτή είναι αφιερωμένη στην αναλυτική παρουσίαση του τρόπου λειτουργίας του Kleio με βάση τις παραδοχές και τις σχεδιαστικές επιλογές που προαναφέρθηκαν. Η υλοποίηση χωρίζεται σε δύο βασικά στάδια:

- Τη φάση της **εκπαίδευσης** (offline training), κατά την οποία εντοπίζονται οι σημαντικές σελίδες και εκπαιδεύεται ένα μοντέλο LSTM για καθεμία από αυτές.
- Τη φάση της **πρόβλεψης** (online inference), κατά την οποία τα εκπαιδευμένα μοντέλα χρησιμοποιούνται σε πραγματικό χρόνο, κατά την εκτέλεση της εφαρμογής, για την πρόβλεψη μελλοντικών προσβάσεων μνήμης.

3.4.1 Φάση εκπαίδευσης

Η περιγραφή ξεκινά με την παρουσίαση της φάσης εκπαίδευσης, σκοπός της οποίας είναι η επιλογή των σημαντικών σελίδων και η εκπαίδευση των μοντέλων. Το τμήμα του κώδικα που επιτελεί τη συγκεκριμένη λειτουργία είναι γραμμένο εξολοκλήρου σε Python. Αρχικά η εκάστοτε εφαρμογή εκτελείται χρησιμοποιώντας πολιτική History Record και προκύπτει έτσι ένα αρχείο καταγραφών που περιλαμβάνει όλα τα ζεύγη εικονικών διευθύνσεων είδος μνήμης (DRAM, NVM), οργανωμένα ανά περίοδο δρομολόγησης.

Το αρχείο αυτό δίνεται ως είσοδος στον Page Selector, ο οποίος σε πρώτη φάση αντιστοιχίζει διευθύνσεις μνήμης σε σελίδες και δημιουργεί δύο πίνακες για κάθε σελίδα: έναν που περιέχει τον αριθμό των προσπελάσεων ανά περίοδο και έναν που καταγράφει το είδος μνήμης όπου βρίσκεται η σελίδα ανά περίοδο. Σύμφωνα με τον κώδικα του Kleio, η λίστα με τις προσπελάσεις μνήμης πρέπει να είναι κβαντισμένη δηλαδή οι προσβάσεις στρογγυλοποιούνται σε συγκεκριμένες διακριτές τιμές που προκύπτουν από την παράμετρο bin size. Οι δύο αυτοί πίνακες χρησιμοποιούνται στη συνέχεια για τον υπολογισμό των αστοχιών τοποθέτησης (misplacement) κάθε σελίδας. Οι συγγραφείς του Kleio ορίζουν ως αστοχία τοποθέτησης το φαινόμενο κατά το οποίο στην αρχή μιας περιόδου δρομολόγησης μια σελίδα έπρεπε να βρίσκεται στη γρήγορη μνήμη αλλά αυτή απουσιάζει, λόγω λανθασμένης πρόβλεψης. Ο υπολογισμός των αστοχιών για μια μεμονωμένη σελίδα γίνεται διατρέχοντας τις δύο λίστες και ελέγχοντας κάθε φορά την πρόβλεψη (που στην περίπτωση του History scheduler ταυτίζεται με τον αριθμό των προσπελάσεων στην προηγούμενη περίοδο), τον τρέχοντα αριθμό προσβάσεων και την τοποθεσία. Αφού υπολογιστούν όλα τα misplacements, ο Page Selector ταξινομεί τις σελίδες σε φθίνουσα σειρά με βάση τη μετρική benefit. Υπενθυμίζουμε πως το benefit ορίζεται ως το γινόμενο του αθροίσματος προσβάσεων με τον αριθμό των misplacement.

Από την ταξινομημένη λίστα σελίδων, επιλέγονται ορισμένες εξ αυτών με βάση τη δυναμική παράμετρο Important Pages, οι οποίες συνθέτουν το σύνολο των σημαντικών σελίδων. Για κάθε μια εξάυτών δημιουργείται ένα νευρωνικό δίκτυο σύμφωνα με τις προδιαγραφές του Kleio. Συγκεκριμένα, το δίκτυο αποτελείται από δύο επίπεδα LSTM με 128 νευρώνες

έκαστο και ένα πυκνό (dense) δίκτυο. Η υλοποίηση έγινε με χρήση του Keras API [33] και του tensorflow [34] ως backend. Όσον αφορά τα δεδομένα εισόδου των μοντέλων, η ακολουθία προσπελάσεων μνήμης ανά περίοδο χωρίζεται σε διαστήματα μήκους 16 (προεπιλεγμένη τιμή της παραμέτρου history length) και κανονικοποιείται με χρήση ενός MinMax scaler. Το 70% του συνόλου δεδομένων χρησιμοποιείται για την εκπαίδευση (training dataset) ενώ το υπόλοιπο λειτουργεί ως σύνολο επαλήθευσης (validation dataset). Η εκπαίδευση διαρκεί το πολύ 120 εποχές, ωστόσο έχει εφαρμοστεί μηχανισμός early stopping που επιτρέπει τη λήξη της εαν η απώλεια στο σύνολο επαλήθευσης δεν μειωθεί μέσα σε ένα διάστημα 20 συνεχόμενων εποχών. Τέλος, ο ρυθμός εκμάθησης είναι ίσος με 0.001 ενώ για τη βελτιστοποίηση επιλέχθηκε ο Adam optimizer. Μόλις ολοκληρωθεί η εκπαίδευση, τα μοντέλα, οι scalers και οι διευθύνσεις των σημαντικών σελίδων αποθηκεύονται για μελλοντική χρήση.

3.4.2 Φάση πρόβλεψης

Η περιγραφή της υλοποίησης ολοκληρώνεται με την επεξήγηση της λειτουργίας του δρομολογητή κατά τη φάση πρόβλεψης, η οποία προϋποθέτει ότι η διαδικασία εντοπισμού των σημαντικών σελίδων έχει ολοκληρωθεί και τα εκπαιδευμένα μοντέλα για κάθε σημαντική σελίδα έχουν αποθηκευτεί. Το μεγαλύτερο μέρος του κώδικα έχει γραφτεί σε C++, ενώ η πρόβλεψη των LSTM πραγματοποιείται μέσω Python 3, χρησιμοποιώντας κατάλληλους C++ wrappers.

Αρχικοποίηση

Το πρώτο βήμα της συγκεκριμένης φάσης είναι η αρχικοποίηση του δρομολογητή. Στο στάδιο αυτό, η εκτέλεση του δρομολογητή μεταφέρεται σε κάποιον επεξεργαστή του NUMA κόμβου 1, προκειμένου να εξασφαλιστεί η πλήρης απομόνωσή του από την εφαρμογή, η οποία θα εκτελεστεί σε επεξεργαστή του NUMA κόμβου 0. Στη συνέχεια, με βάση τα αποτελέσματα της φάσης εκπαίδευσης, ανακτάται η λίστα των σημαντικών σελίδων, αρχικοποιείται ο διερμηνέας της Python και φορτώνονται τα μοντέλα σε μια δομή λεξικού (dictionary), ώστε να αποφευχθούν οι επαναλαμβανόμενες μελλοντικές αναγνώσεις αρχείων. Επιπλέον, διαβάζονται όλες οι δυναμικές παράμετροι από το configuration file που αναφέρθηκε νωρίτερα και τέλος καθορίζονται οι απαραίτητες ρυθμίσεις για τη δειγματοληψία, χωρίς όμως αυτή να εκκινεί ακόμα.

Στη συνέχεια, δημιουργείται μια θυγατρική διεργασία, η οποία, αφού μεταφερθεί σε κάποιον επεξεργαστή του NUMA κόμβου 0, αδρανοποιείται χρησιμοποιώντας το σήμα SIGSTOP. Μόλις ο γονέας ολοκληρώσει τις απαραίτητες αρχικοποιήσεις, ενεργοποιεί τη δειγματοληψία και επανεκκινεί τη διεργασία-παιδί (στέλνοντας το σήμα SIGCONT). Κατόπιν, η θυγατρική διεργασία εκτελεί την εφαρμογή με την κλήση συστήματος `execv()` ενώ η γονική αναλαμβάνει χρέη δρομολογητή.

Ο δρομολογητής διατηρεί μια ειδική δομή (struct Page) για κάθε σελίδα, η οποία περιλαμβάνει δύο βασικά στοιχεία: έναν πίνακα ιστορικού και μία ακέραια τιμή που αντιπροσωπεύει την πρόβλεψη του αριθμού προσπελάσεων κατά την επόμενη περίοδο. Για να αποφευχθεί η διατήρηση περιττών δεδομένων, ο πίνακας ιστορικού για τις σημαντικές σελίδες έχει μέγεθος ίσο με το history length των LSTM μοντέλων, αποθηκεύοντας πληροφορίες

για τις τελευταίες history length προσπελάσεις, ενώ για τις υπόλοιπες σελίδες διατηρείται μόνο μία τιμή, ο αριθμός των προσπελάσεων κατά την προηγούμενη περίοδο.

Όλες οι σελίδες οργανώνονται σε μια κοινή δομή δεδομένων τύπου λεξικού (συγκεκριμένα χρησιμοποιήθηκε η map της STL) που ονομάζεται page_list. Πρόκειται για μια δομή κλειδιού-τιμής (key-value), όπου το κλειδί κάθε εγγραφής είναι η εικονική διεύθυνση της σελίδας και η τιμή είναι ένα αντικείμενο τύπου struct Page. Αυτή η προσέγγιση διευκολύνει τη γρήγορη πρόσβαση σε οποιαδήποτε σελίδα.

Δρομολόγηση

Η δρομολόγηση, σε κάθε χρονική περίοδο, πραγματοποιείται μέσω μιας διαδικασίας τριών βημάτων.

1. **Δειγματοληψία**: Ο δρομολογητής παραμένει ανενεργός για ένα χρονικό διάστημα που ορίζεται από τη δυναμική παράμετρο scheduling period, επιτρέποντας στην εφαρμογή να συνεχίσει κανονικά την εκτέλεσή της κατά το διάστημα αυτό. Αμέσως μετά, διακόπτει την εκτέλεση της εφαρμογής αποστέλλοντας το σήμα SIGSTOP, σταματά προσωρινά τη διαδικασία δειγματοληψίας και συλλέγει όλα τα δείγματα.

2. **Υπολογισμός προβλέψεων**: Αρχικά πραγματοποιείται η προεπεξεργασία των δειγμάτων κατά την οποία κάθε εικονική διεύθυνση αντιστοιχίζεται στην κατάλληλη σελίδα και ενημερώνονται οι εγγραφές στην page_list. Το πιο κρίσιμο βήμα του δρομολογητή είναι ο υπολογισμός των προβλέψεων για τις προσπελάσεις της επόμενης περιόδου. Οι προβλέψεις για τις σημαντικές σελίδες υπολογίζονται μέσω του αντίστοιχου LSTM μοντέλου (inference) χρησιμοποιώντας τον Python interpreter που είχε δημιουργηθεί κατά τη φάση της αρχικοποίησης. Ως είσοδος σε κάθε μοντέλο παρέχεται ο πίνακας ιστορικού της αντίστοιχης σελίδας ενώ η έξοδος αποτελεί την προβλεπόμενη τιμή προσπελάσεων της σελίδας κατά την επόμενη περίοδο δρομολόγησης. Για τις υπόλοιπες σελίδες, οι προβλέψεις για τις προσπελάσεις μνήμης στην επόμενη περίοδο ταυτίζονται με τον καταγεγραμμένο αριθμό προσβάσεων κατά την τρέχουσα περίοδο, όπως ακριβώς ορίζει η πολιτική History. Η παραπάνω περιγραφή αφορά αποκλειστικά την πολιτική Kleio. Ωστόσο, η υλοποίηση υποστηρίζει επίσης τις πολιτικές History και History Record, στις οποίες δεν γίνεται διάκριση σημαντικών σελίδων, δεν χρησιμοποιούνται μοντέλα, και όλες οι προβλέψεις βασίζονται αποκλειστικά στο ιστορικό των τελευταίων προσπελάσεων.

3. **Απόφαση - Μετακίνηση σελίδων**: Κατόπιν, ανεξαρτήτως πολιτικής, οι σελίδες ταξινομούνται σε φθίνουσα σειρά προβλεπόμενων προσπελάσεων. Σύμφωνα με αυτή την ταξινόμηση, οι σελίδες με τις υψηλότερες προβλέψεις μεταφέρονται στη DRAM (NUMA κόμβος 0) μέχρι να εξαντληθεί η διαθέσιμη χωρητικότητά της, ενώ οι υπόλοιπες μεταφέρονται στην NVM (NUMA κόμβος 2). Η μετακίνηση των σελίδων γίνεται με τη συνάρτηση `move_pages()`. Τέλος, ο δρομολογητής επανεκκινεί την εφαρμογή και τη δειγματοληψία.

Η διαδικασία αυτή επαναλαμβάνεται έως ότου ολοκληρωθεί η εκτέλεση της εφαρμογής.

Κεφάλαιο 4

Πειραματική αξιολόγηση

Το κεφάλαιο αυτό είναι αφιερωμένο στην αξιολόγηση της απόδοσης του δρομολογητή. Αρχικά, επαληθεύται η λειτουργικότητα του Kleio κάτω από ιδανικές συνθήκες με τη βοήθεια ενός τεχνητού microbenchmark το οποίο εμφανίζει κατάλληλο μοτίβο προσπελάσεων μνήμης. Στη συνέχεια, εξετάζεται λεπτομερώς η απόκριση του δρομολογητή σε επιλεγμένα μετροπρογράμματα από τις συλλογές PARSEC και SPEC, παρέχοντας έτσι μια ολοκληρωμένη και ρεαλιστική αποτίμηση της συνολικής του απόδοσης.

4.1 Περιβάλλον πειραματικής αξιολόγησης

Το σύστημα ακολουθεί τη διάταξη που περιγράφηκε στο Κεφάλαιο 3, δηλαδή αποτελείται από τέσσερις NUMA κόμβους. Οι κόμβοι 0 και 1 διαθέτουν από 96GB DRAM, ενώ οι κόμβοι 2 και 3 περιλαμβάνουν από 282GB Intel Optane DC Persistent Memory (NVM). Στα πειράματα που ακολουθούν χρησιμοποιούνται μόνο οι κόμβοι 0 (DRAM) και 2 (NVM), καθώς οι απαιτήσεις μνήμης των μετροπρογραμμάτων δεν ήταν ιδιαίτερα μεγάλες. Η αρχική αξιολόγηση πραγματοποιήθηκε σε ένα τεχνητά κατασκευασμένο μετροπρόγραμμα, το οποίο περιγράφεται αναλυτικά στη συνέχεια. Ακολούθως, δοκιμάστηκαν επιλεγμένα benchmarks από τις συλλογές PARSEC και SPEC. Η επιλογή των παραμέτρων δειγματοληψίας γίνεται σε επίπεδο εφαρμογής, μέσω διαδικασίας που επίσης παρουσιάζεται παρακάτω.

Με εξαίρεση το τεχνητό μετροπρόγραμμα, όπου η αναλογία γρήγορης προς αργή μνήμη ορίστηκε σε 0.25, σε όλες τις υπόλοιπες περιπτώσεις χρησιμοποιείται αναλογία 0.125. Εκτός αν αναφέρεται διαφορετικά, η περίοδος δρομολόγησης έχει οριστεί σε 1 δευτερόλεπτο. Η αξιολόγηση της απόδοσης βασίζεται κυρίως στη μετρική **DRAM hit rate**, δηλαδή το ποσοστό των προσβάσεων μνήμης που εξυπηρετούνται επιτυχώς από την γρήγορη μνήμη. Προκειμένου να εκτιμηθεί η αποτελεσματικότητα του Kleio, σε κάθε περίοδο δρομολόγησης υπολογίζεται αναδρομικά και το DRAM hit rate που θα είχε προκύψει σε περίπτωση εφαρμογής της πολιτικής History ή της ιδανικής Oracle. Στο σημείο αυτό, αξίζει να αναφερθεί πως όλοι οι μηχανισμοί προανάκτησης (prefetchers) έχουν απενεργοποιηθεί ώστε να μην επηρεάζονται οι μετρήσεις.

Τέλος, τα μοντέλα που χρησιμοποιούνται ακολουθούν πιστά τη δομή του Kleio όπως περιγράφεται στο Κεφάλαιο 2, αποτελούνται δηλαδή από δύο επίπεδα LSTM και ένα πυκνό (Dense) επίπεδο εξόδου. Η υλοποίησή τους έγινε με τη χρήση της βιβλιοθήκης Keras (έκδοση 3.6.0), με το TensorFlow (έκδοση 2.17.0) να χρησιμοποιείται ως πλατφόρμα εκτέλεσης των

RNN μοντέλων. Οι τιμές των βασικών παραμέτρων παρουσιάζονται στον Πίνακα4.1, ενώ για όλες τις υπόλοιπες χρησιμοποιούνται οι προκαθορισμένες τιμές του Keras. Για την εκπαίδευση εφαρμόζεται ο βελτιστοποιητής Adam, ενώ χρησιμοποιείται και τεχνική Early stopping, δηλαδή η διαδικασία διακόπτεται αυτόματα όταν δεν παρατηρείται βελτίωση στην απώλεια του συνόλου επαλήθευσης (validation dataset loss).

Παράμετρος	**Τιμή**
Αριθμός LSTM νευρώνων ανά επίπεδο	128
Μήκος εισόδου (history length)	16
Recurrent dropout	0.0
Ρυθμός εκμάθησης (learning rate)	0.0089

Πίνακας 4.1: *Βέλτιστος συνδυασμός παραμέτρων που εντοπίστηκε με χρήση του πακέτου Optuna*

4.1.1 Περιγραφή του microbenchmark

Προκειμένου να επιβεβαιωθεί η χρησιμότητα των μοντέλων στη διαδικασία δρομολόγησης και να μελετηθεί η επίδραση των σχεδιαστικών επιλογών και παραδοχών, υλοποιήθηκε ένα ειδικό μετροπρόγραμμα σε C++ που παράγει ένα ξεκάθαρο μοτίβο στις προσπελάσεις μνήμης που αστοχούν στην τελευταία κρυφή μνήμη.

Αλγόριθμος του microbenchmark

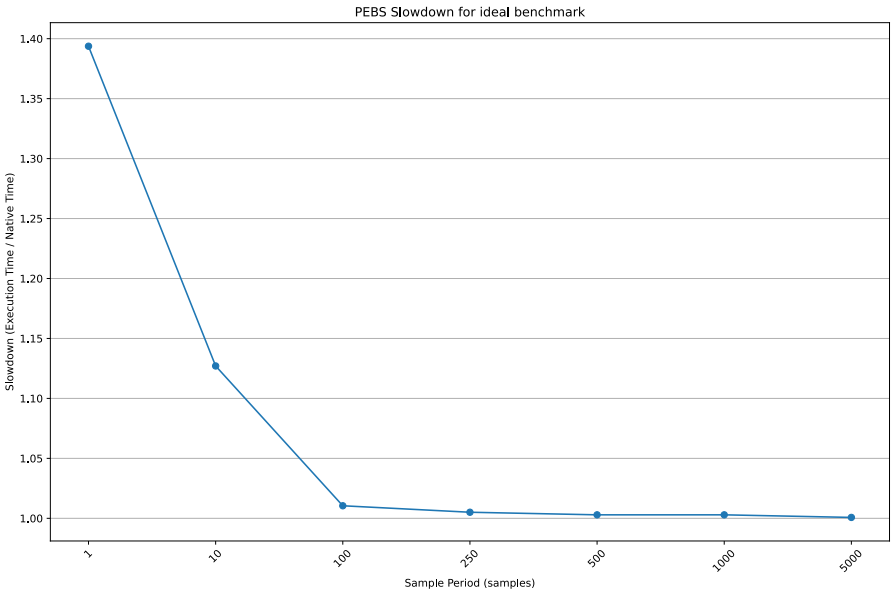
Το μετροπρόγραμμα που κατασκευάστηκε λειτουργεί επαναληπτικά πάνω σε έναν πίνακα μεγέθους 400MB, ο οποίος χωρίζεται νοητά σε 4 ομάδες (μεγέθους 100MB η καθεμία). Με αφετηρία την πρώτη ομάδα, σε κάθε επανάληψη επιλέγεται κυκλικά η επόμενη ομάδα στην οποία προκαλούνται αλλεπάλληλες αστοχίες ανάγνωσης στην κρυφή μνήμη τελευταίου επιπέδου. Ο μηχανισμός που χρησιμοποιήθηκε για την παραγωγή των LLC load misses βασίζεται σε έναν συνδυασμό της αλυσιδωτής προσπέλασης μέσω δεικτών (pointer chasing) και ενός μοτίβου σταθερού βήματος (strided pattern). Συγκεκριμένα, κάθε τιμή που ανακτάται από μια εντολή ανάγνωσης χρησιμοποιείται για να προσδιοριστεί η διεύθυνση μνήμης της επόμενης ανάγνωσης (Διεύθυνση επόμενης ανάγνωσης = Διεύθυνση προηγούμενης ανάγνωσης + Τιμή προηγούμενης ανάγνωσης × Μέγεθος cache line). Ωστόσο, κάθε θέση του πίνακα αρχικοποιείται στην τιμή 7, με αποτέλεσμα να δημιουργείται έτσι ένα σταθερό βήμα μήκους 7 × Μέγεθος cache line. Ταυτόχρονα, υλοποιήθηκε και κατάλληλος μηχανισμός που εξασφαλίζει ότι η διεύθυνση ανάγνωσης παραμένει στα όρια της εκάστοτε ομάδας. Πραγματοποιούνται συνολικά 250 εξωτερικές επαναλήψεις του αλγορίθμου και σε κάθε επανάληψη, παράγονται περίπου 10^6 αναγνώσεις με το προαναφερθέν μοτίβο για τη δημιουργία των LLC misses. Για την προκαθορισμένη τιμή περιόδου δρομολόγησης (1 sec), η εκτέλεση διαρκεί περίπου 250 περιόδους.

Όπως έχει ήδη αναφερθεί, για την αξιολόγηση του μετροπρογράμματος η αναλογία DRAM προς NVM ορίζεται ίση με 0.25, έτσι ώστε σε κάθε περίοδο δρομολόγησης η γρήγορη μνήμη να μπορεί να φιλοξενήσει μόνο το 1/4 του συνολικού working set (δηλαδή μία ομάδα του πίνακα).

Ρύθμιση των παραμέτρων δειγματοληψίας

Το επόμενο βήμα αφορά τον καθορισμό των βασικών παραμέτρων της δειγματοληψίας στο τεχνητό μετροπρόγραμμα, δηλαδή της περιόδου δειγματοληψίας και του μεγέθους του ring buffer. Ο στόχος είναι η συλλογή όσο το δυνατόν λεπτομερέστερων πληροφοριών για τις προσπελάσεις μνήμης, ελαχιστοποιώντας παράλληλα την επιβάρυνση του συστήματος. Για τον σκοπό αυτό, πραγματοποιήθηκαν δοκιμές με διάφορες τιμές του sample period στο εύρος 1 - 5000, καταγράφοντας σε κάθε περίπτωση την επιβράδυνση, το μέσο μέγεθος των δειγμάτων που συλλέγονται ανά περίοδο δρομολόγησης και τον αριθμό των περιστατικών throttling. Η επιβράδυνση ορίζεται ως το πηλίκο του χρόνου εκτέλεσης με χρήση PEBS προς τον αρχικό χρόνο εκτέλεσης. Αξίζει να σημειωθεί ότι στα πειράματα αυτά δεν περιλαμβάνεται η διαδικασία δρομολόγησης, γεγονός που σημαίνει ότι όλες οι σελίδες βρίσκονται στη γρήγορη μνήμη. Συνεπώς, η μέτρηση αφορά αποκλειστικά την επιβάρυνση που προσθέτει η δειγματοληψία PEBS σε ένα μόνο event.

Από το διάγραμμα 4.1, το οποίο απεικονίζει την επιβράδυνση συναρτήσει της περιόδου δειγματοληψίας, προκύπτει ότι η τιμή 500 αποτελεί μια καλή επιλογή του Sample period που δεν επιβαρύνει καθόλου την εκτέλεση του μετροπρογράμματος. Αξίζει να σημειωθεί πως φαινόμενα CPU throttling παρατηρούνται μόνο για μικρές περιόδους δειγματοληψίας (1 - 10).



Σχήμα 4.1: *Επιβράδυνση συναρτήσει περιόδου δειγματοληψίας για το microbenchmark*

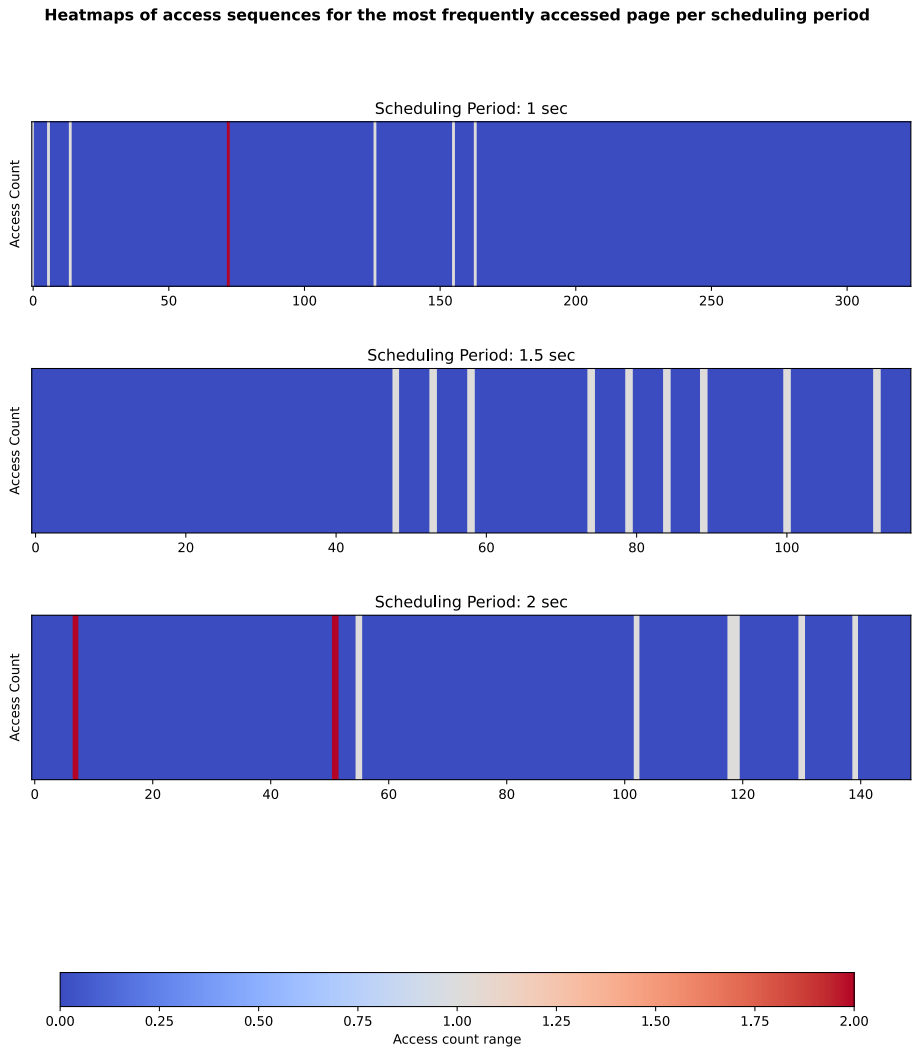
Τέλος, σε κάθε περίοδο δεν απαιτούνται πάνω από 2MB για την αποθήκευση των δειγμάτων ακόμα και για χαμηλές τιμές περιόδου δειγματοληψίας. Ωστόσο, υπάρχει ένας περιορισμός του perf_events σύμφωνα με τον οποίο το το μέγεθος του ring buffer πρέπει να είναι της μορφής $(2^N + 1) \times$ pagesize, όπου N είναι ένας θετικός ακέραιος αριθμός. Για τον λόγο αυτό, το τελικό μέγεθος του ring buffer ορίστηκε ίσο με 8MB.

4.1.2 Επίδραση της Δειγματοληπτικής Καταγραφής Προσβάσεων στη Συμπεριφορά του Δρομολογητή

Η ενότητα αυτή εξετάζει τον ρόλο των ακολουθιών πρόσβασης που παράγονται μέσω δειγματοληψίας στην εκπαίδευση των μοντέλων και στη συμπεριφορά του δρομολογητή. Στο πλαίσιο αυτό, ο δρομολογητής αξιολογήθηκε στο τεχνητά κατασκευασμένο μετροπρόγραμμα που παρουσιάστηκε στην προηγούμενη ενότητα. Αρχικά, χρησιμοποιήθηκαν **σελίδες μεγέθους 4KB** και δοκιμάστηκαν τρεις διαφορετικές τιμές περιόδου δρομολόγησης: 1, 1.5 και 2 δευτερόλεπτα. Η διαμόρφωση των μοντέλων ακολούθησε την παραμετροποίηση του πίνακα 4.1. Η τιμή του bin size καθορίστηκε ίση με 1, ώστε ακόμα και η παραμικρή απόκλιση μιας πρόβλεψης του History page scheduler από την πραγματική τιμή να θεωρείται αποτυχία (misprediction). Τέλος, μοντέλα LSTM εκπαιδεύτηκαν αποκλειστικά για τις 100 σελίδες με την υψηλότερη τιμή στη μετρική benefit.

Τα πρώτα αποτελέσματα έδειξαν ότι τα μοντέλα δεν κατάφεραν να αποτυπώσουν σωστά το μοτίβο προσβάσεων κάθε σελίδας, καθώς όλες οι προβλεπόμενες τιμές μελλοντικών προσπελάσεων ήταν μηδενικές. Μετά από λεπτομερή αξιολόγηση, διαπιστώθηκε πως η ρίζα του προβλήματος εντοπίζεται στη χρήση της δειγματοληψίας, όπως αυτή έχει ρυθμιστεί για τις ανάγκες της υλοποίησης, πάνω σε σελίδες μεγέθους 4KB. Όπως φαίνεται και στην εικόνα 4.2, ακόμα και για τις πιο συχνά προσπελάσιμες σελίδες που εντοπίστηκαν κατά τη δειγματοληψία, τα δείγματα είναι εξαιρετικά αραιά. Δεν υπάρχει σελίδα για την οποία να καταγράφονται περισσότερες από 2 προσβάσεις σε διάστημα μιας περιόδου δρομολόγησης ενώ παρατηρούνται πολλές περιπτώσεις όπου για πάνω από 16 συνεχόμενες εποχές (τιμή ίση με το history length των μοντέλων) δεν καταγράφεται καμία προσπέλαση. Συνεπώς, είναι απολύτως λογικό οι προβλέψεις των μοντέλων να είναι πάντοτε μηδενικές.

Η συμπεριφορά αυτή δεν είναι παράλογη αν αναλογιστεί κανείς πως το μετροπρόγραμμα διαχειρίζεται συνολικά 25.600 σελίδες, ενώ η δειγματοληψία συλλέγει ένα δείγμα ανά 500 LLC misses. Υπό αυτές τις συνθήκες, είναι αναμενόμενο ότι οι περισσότερες σελίδες θα εμφανίζουν περιορισμένο αριθμό καταγεγραμμένων προσπελάσεων, καθώς μια υψηλή τιμή θα προϋπέθετε ιδιαίτερα έντονη πίεση στο σύστημα μνήμης από την πλευρά του benchmark.



Σχήμα 4.2: *Ακουλουθία προσβάσεων της συχνότερα προσπελάσιμης σελίδας για τις διάφορες τιμές περιόδου δρομολόγησης*

Πριν εξεταστεί όμως η αύξηση του μεγέθους των σελίδων, επιχειρήθηκε η διερεύνηση ορισμένων εναλλακτικών λύσεων στο πλαίσιο των σελίδων μεγέθους 4KB. Δύο ενδιαφέρουσες επιλογές οι οποίες όμως δεν ευδοκίμησαν ήταν οι εξής:

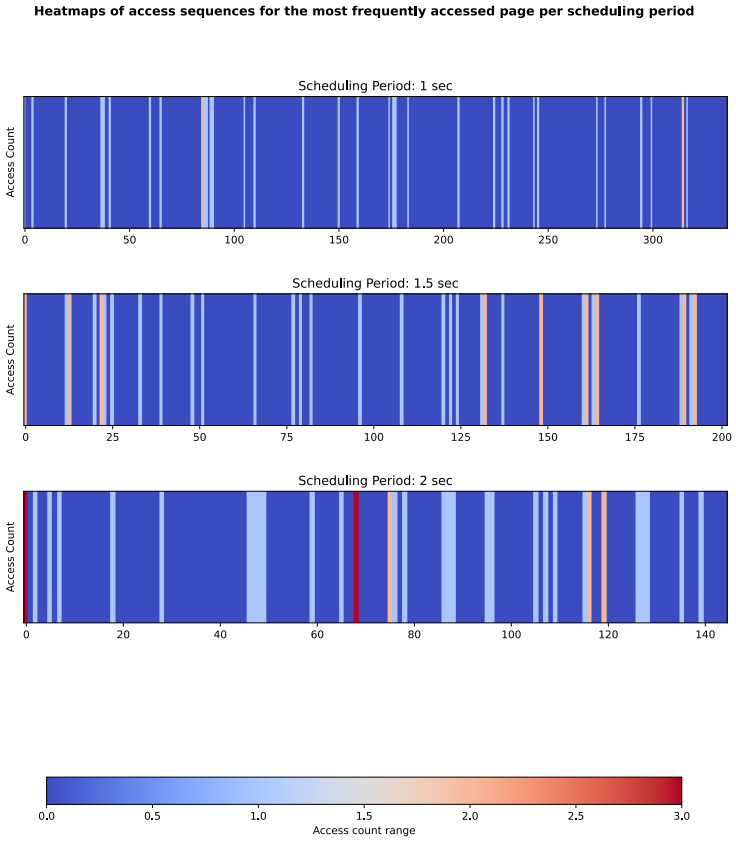
1. **Η αλλαγή της της περιόδου δειγματοληψίας**: Ως πρώτο βήμα, η περίοδος δειγματοληψίας μειώθηκε από 500 σε 100, προκειμένου να αυξηθεί η πυκνότητα των δειγμάτων και να δημιουργηθεί μια πιο λεπτομερής αποτύπωση της συμπεριφοράς του συστήματος. Παρόλο που ο μέγιστος αριθμός προσβάσεων ανά εποχή αυξήθηκε ελάχιστα (από 2 σε 4), το φαινόμενο των συνεχόμενων μηδενικών προσβάσεων παρέμεινε εξίσου έντονο και συνεπώς η επίδοση του δρομολογητή δεν παρουσίασε κάποια βελτίωση.

2. **Η αύξηση του RNN history length των μοντέλων**: Σκοπός αυτής της προσέγγισης είναι η αξιοποίηση δεδομένων από ένα ευρύτερο εύρος προηγούμενων προσβάσεων

κατά την πρόβλεψη των μελλοντικών προσπελάσεων. Ωστόσο, η προσέγγιση αυτή απορρίφθηκε διότι για μια ακολουθία μήκους N οι δυνατές συνεχόμενες υπακολουθίες μήκους RNN history length ανέρχονται σε $N - \text{RNN history length}$ και έτσι η αύξηση του history length θα επέφερε αισθητή μείωση του συνολικού συνόλου δεδομένων εκπαίδευσης για κάθε μοντέλο, υποβαθμίζοντας με αυτόν τον τρόπο την ακρίβεια των προβλέψεων.

Τροποποίηση μετροπρογράμματος

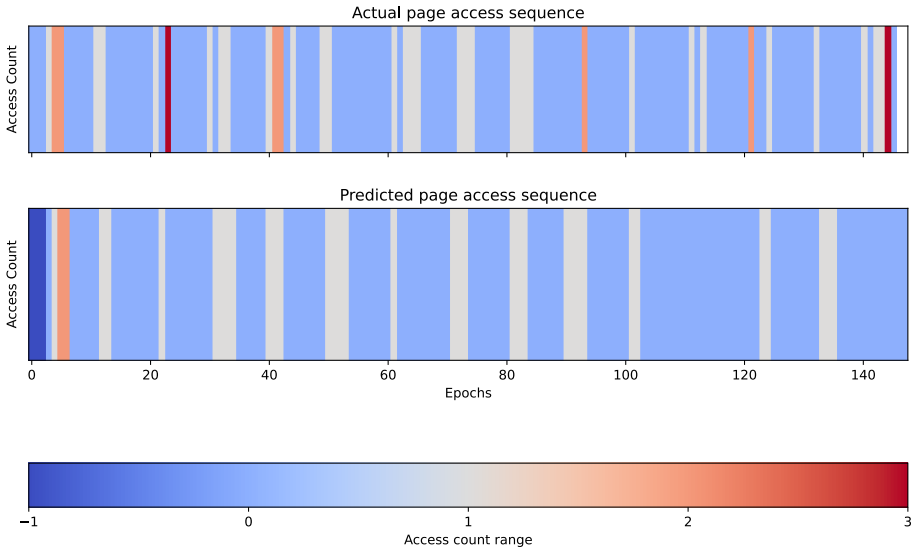
Στη συνέχεια, εξετάστηκε το ενδεχόμενο η αποτελεσματικότητα του δρομολογητή να περιορίζεται σε εφαρμογές με ιδιαίτερα έντονη δραστηριότητα μνήμης. Για τον σκοπό αυτό, δημιουργήθηκε μια τροποποιημένη εκδοχή του microbenchmark, στην οποία το μέγεθος του πίνακα μειώθηκε κατά 4 φορές (από 400MB σε 100MB, με ανάλογη μείωση του μεγέθους κάθε ομάδας) ενώ ο αριθμός των εσωτερικών επαναλήψεων που πραγματοποιούνται για την παραγωγή αστοχιών κρυφής μνήμης σε κάθε ομάδα αυξήθηκε κατά 7 περίπου φορές (7×10^6). Τέλος, ο αριθμός των εξωτερικών επαναλήψεων μειώθηκε κατά 8 φορές για να διατηρηθεί σχεδόν σταθερός ο συνολικός χρόνος εκτέλεσης του μετροπρογράμματος.



Σχήμα 4.3: *Ακολουθία προσβάσεων της συχνότερα προσπελάσιμης σελίδας για τις διάφορες τιμές περιόδου δρομολόγησης στην τροποποιημένη έκδοση του microbenchmark*

Από την εικόνα 4.3 γίνεται φανερό ότι η εντατικότερη παραγωγή αστοχιών κρυφής μνήμης δεν αποτυπώθηκε αισθητά στα αποτελέσματα της δειγματοληψίας, καθώς ο αριθ-

μός των καταγεγραμμένων προσβάσεων μνήμης εξακολουθεί να κυμαίνεται μεταξύ 0 και 3. Παρ' όλα αυτά, το φαινόμενο των συνεχόμενων μηδενικών προσβάσεων για διάστημα άνω των 16 εποχών έχει περιοριστεί. Αν και αρκετές προβλέψεις εξακολουθούν να είναι εσφαλμένα μηδενικές, υπάρχουν αρκετές περιπτώσεις στις οποίες πλησιάζουν αρκετά τις πραγματικές τιμές, όπως αποδεικνύεται στο παράδειγμα της εικόνας 4.4.



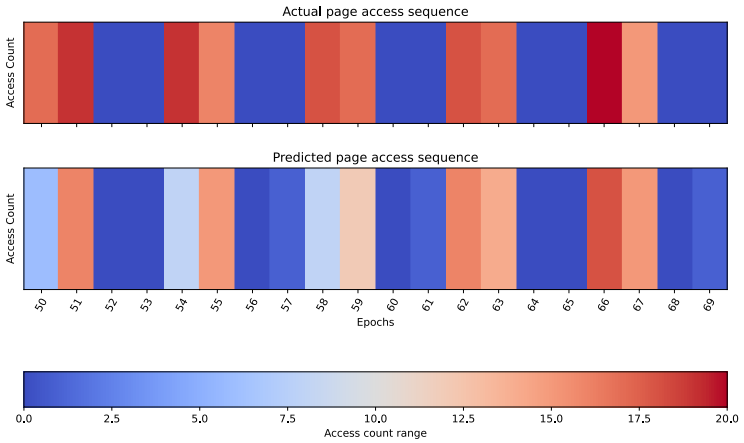
Σχήμα 4.4: *Σύγκριση μεταξύ πραγματικής και προβλεπόμενης ακολουθίας προσπελάσεων για μια τυχαία σελίδα σε περίοδο δρομολόγησης 2 sec*

Η σαφώς βελτιωμένη εικόνα που παρουσιάζουν τα μοντέλα στην ανανεωμένη έκδοση του microbenchmark δεν αρκεί για να αναδείξει σημαντική υπεροχή του Kleio έναντι του κλασικού History scheduler, αφού αμφότερες οι δύο πολιτικές επιτυγχάνουν ένα μέσο ποσοστό ευστοχίας στη γρήγορη μνήμη ίσο με 22%. Η συμπεριφορά αυτή οφείλεται κατά βάση στο γεγονός ότι η έξυπνη διαχείριση αφορά μόλις το 0.39% του συνόλου των σελίδων (100 από τις 25600). Μια προφανής προσέγγιση για την αντιμετώπιση αυτού του ζητήματος είναι η εκπαίδευση περισσότερων μοντέλων. Ωστόσο, αυτή η λύση δεν είναι κλιμακώσιμη, καθώς θα οδηγούσε σε υπερβολική κατανάλωση υπολογιστικών πόρων. Αντίθετα, στην πρόσφατη εργασία τους Coeus [35], οι δημιουργοί του Kleio προτείνουν μια εναλλακτική στρατηγική, αξιοποιώντας την παρατήρηση ότι, με κατάλληλη ρύθμιση της περιόδου δρομολόγησης, ορισμένες σελίδες εμφανίζουν ίδια ακριβώς μοτίβα προσβάσεων. Αυτό επιτρέπει την ομαδοποίηση τους κάτω από το ίδιο RNN, βελτιώνοντας έτσι την απόδοση της μοντελοποίησης. Όμως η συγκεκριμένη μέθοδος δεν συνδυάζεται εύκολα με τη χρήση της δειγματοληψίας, αφού δεν συλλέγονται όλες οι προσβάσεις μνήμης γεγονός που δημιουργεί υψηλή μεταβλητότητα στις καταγεγραμμένες ακολουθίες προσβάσεων. Ακόμα και σε δύο διαδοχικές εκτελέσεις, το ιστορικό των καταγεγραμμένων προσβάσεων μιας σελίδας μπορεί να διαφέρει σημαντικά, καθιστώντας δύσκολη την αξιόπιστη εκπαίδευση των μοντέλων. Η επίδραση αυτής της

μεταβλητότητας θα αναλυθεί λεπτομερώς σε επόμενο κεφάλαιο.

Το συμπέρασμα που προκύπτει μέχρι στιγμής είναι πως η χρήση σελίδων μεγέθους 4KB κρίνεται ακατάλληλη. Συνεπώς η λογική κατεύθυνση είναι η αύξηση του μεγέθους σελίδας σε 2MB. Η αλλαγή αυτή έγινε με χρήση του μηχανισμού Transparent Huge Pages (THP) που προσφέρει το Linux. Πρόκειται για μια τεχνική που επιτρέπει τη δυναμική χρήση μεγάλων σελίδων μνήμης (2MB Huge Pages) χωρίς τροποποίηση του κώδικα των εφαρμογών. Η χρήση των Huge Pages εν γένει στοχεύει στην ελαχιστοποίηση των αστοχιών κρυφής μνήμης μετάφρασης διευθύνσεων (TLB misses) καθώς μία εγγραφή καλύπτει μεγαλύτερο τμήμα μνήμης (2MB αντί για 4KB). Από την άλλη, ενδέχεται να οδηγήσει σε αυξημένο κατακερματισμό μνήμης και επιπλέον υπολογιστικό κόστος λόγω των λειτουργιών συγχώνευσης και αποσυγχώνευσης που εκτελεί ο πυρήνας. Στην προκειμένη περίπτωση όμως, σκοπός είναι η ομαδοποίηση των προσπελάσεων μνήμης χωρίς να απαιτούνται τροποποιήσεις στον μηχανισμό δειγματοληψίας. Οι μετρητές PEBS συνεχίζουν να καταγράφουν μεμονωμένες διευθύνσεις αναγνώσεων που αστοχούν σε όλα τα επίπεδα κρυφής μνήμης. Ωστόσο, με την αύξηση του μεγέθους της σελίδας, αυξάνεται σημαντικά η πιθανότητα δύο διαφορετικές διευθύνσεις να αντιστοιχούν στην ίδια σελίδα. Κάτι τέτοιο αρκεί για να αντιμετωπιστεί το πρόβλημα των αραιών πινάκων ιστορικού προσπελάσεων, όπου οι καταγεγραμμένες προσβάσεις ήταν σχεδόν πάντα μηδενικές, και διευκολύνει την εκμάθηση των μοντέλων.

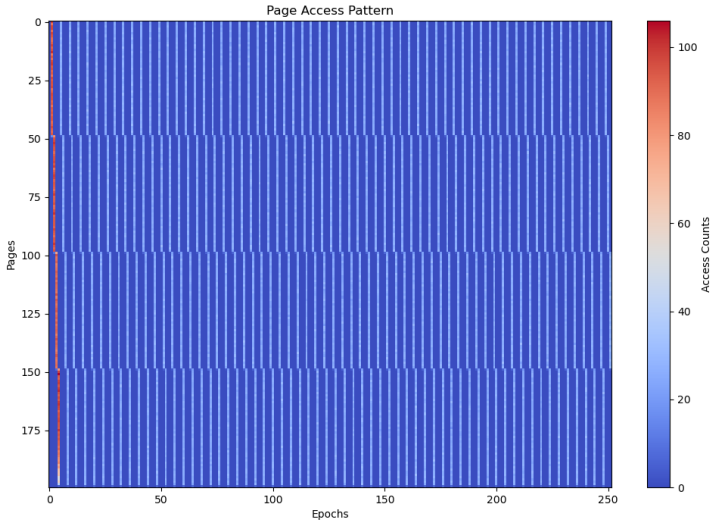
Όπως φαίνεται και στην εικόνα 4.5, η χρήση σελίδων μεγέθους 2MB, σε συνδυασμό με περίοδο δειγματοληψίας ίση με 500, οδηγεί σε σημαντική βελτίωση των καταγεγραμμένων ακολουθιών προσβάσεων, γεγονός που ενισχύει την προβλεπτική ικανότητα των μοντέλων. Οι συνεχόμενες περίοδοι χωρίς καμία καταγραφή πλέον εκλείπουν, ενώ ο μέγιστος αριθμός προσπελάσεων φτάνει τις 40.



Σχήμα 4.5: *Σύγκριση μεταξύ πραγματικής και προβλεπόμενης ακολουθίας προσπελάσεων για μια τυχαία σελίδα μεγέθους 2MB*

4.1.3 Παραμετροποίηση των μοντέλων μηχανικής μάθησης

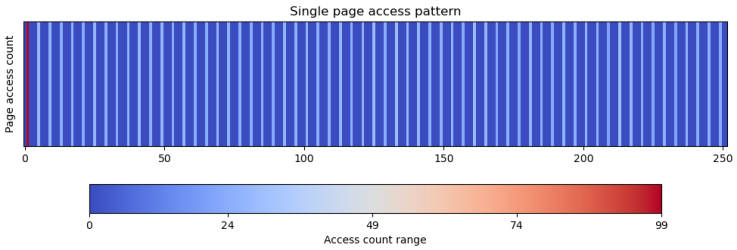
Το επόμενο στάδιο αφορά την αξιολόγηση της ικανότητας των μοντέλων να αποτυπώνουν και να προβλέπουν με ακρίβεια τη συμπεριφορά των σελίδων μνήμης, χρησιμοποιώντας πλέον σελίδες μεγέθους 2MB. Στο πλαίσιο αυτό, υλοποιήθηκε μία ακόμη παραλλαγή του microbenchmark με στόχο τη δημιουργία ιδανικών συνθηκών για την αξιολόγηση της αποτελεσματικότητας των μοντέλων πρόβλεψης. Σε αντίθεση με την αρχική εκδοχή, όπου κάθε εξωτερική επανάληψη του microbenchmark περιλάμβανε έναν σταθερό αριθμό εσωτερικών επαναλήψεων για την πρόκληση των LLC misses στην εκάστοτε ομάδα, η νέα έκδοση μεταθέτει τον έλεγχο των συνολικών επαναλήψεων στον δρομολογητή. Συγκεκριμένα, κάθε φορά που ο δρομολογητής στέλνει σήμα εκκίνησης, το μετροπρόγραμμα επιλέγει κυκλικά την επόμενη ομάδα και παράγει διαρκώς σε αυτήν LLC misses έως ότου ο δρομολογητής δώσει εντολή παύσης. Με αυτόν τον τρόπο, επιτυγχάνεται πλήρης ευθυγράμμιση μεταξύ της περιόδου δρομολόγησης και της επανάληψης του microbenchmark, διασφαλίζοντας ότι σε κάθε περίοδο δρομολόγησης προσπελαύνεται αποκλειστικά μία ομάδα. Οι παρακάτω εικόνες (4.6, 4.7) επιβεβαιώνουν τον σχηματισμό του αναμενόμενου μοτίβου προσπελάσεων. Ο στόχος είναι ξεκάθαρος: Τα LSTM μοντέλα θα πρέπει να είναι σε θέση να μάθουν το μοτίβο προσπελάσεων ενώ η πολιτική του History scheduler είναι καταδικασμένη να αποτυγχάνει σε κάθε περίοδο.



Σχήμα 4.6: *Ιστορικό προσπελάσεων για όλες τις σελίδες του microbenchmark ανά περίοδο δρομολόγησης.*

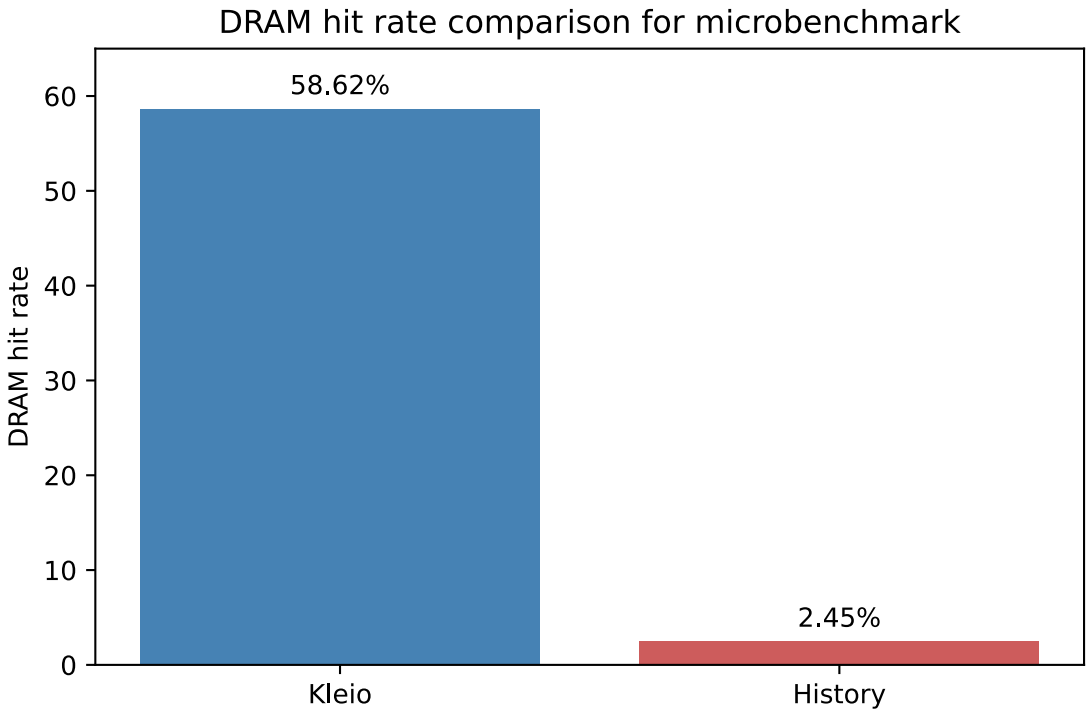
Ρύθμιση παραμέτρων

Σε μια πρώτη αξιολόγηση, οι τιμές των παραμέτρων κάθε μοντέλου διατηρούνται ίδιες με τις αντίστοιχες που επέλεξαν οι δημιουργοί του Kleio (πίνακας 4.1). Συγκεκριμένα, κάθε ένα εκ των δύο επιπέδων LSTM διαθέτει 128 νευρώνες, το μήκος εισόδου (history length) ισούται με 16 ενώ ο ρυθμός εκμάθησης παραμένει 0.001. Με αυτήν την παραμετροποίηση, ο History



Σχήμα 4.7: *Ιστορικό προσπελάσεων μιας μεμονωμένης σελίδας του microbenchmark με μοτίβο προσπελάσεων ανά 4 περιόδους*

page scheduler, όπως αναμενόταν, επιτυγχάνει ποσοστό ευστοχίας στη γρήγορη μνήμη μόλις 2.45%. Αντίθετα, το Kleio σημειώνει σημαντικά υψηλότερο ποσοστό ευστοχίας, φτάνοντας το 58.62% (σχήμα 4.8). Παρόλο που η επίδοση του Kleio είναι 23.7 φορές μεγαλύτερη από εκείνη του History scheduler, δεν κρίνεται ιδιαιτέρως ικανοποιητική, καθώς το μοτίβο προσβάσεων ήταν αρκετά απλό και συνεπώς υπάρχουν μεγάλα περιθώρια βελτίωσης.



Σχήμα 4.8: *Σύγκριση των DRAM hit rate μεταξύ Kleio και History για την αρχική παραμετροποίηση των μοντέλων*

Με μια πιο προσεκτική ματιά στην εξέλιξη της διαδικασίας εκμάθησης, διαπιστώθηκε ότι η ελλιπής εκμάθηση του μοτίβου αποδίδεται κυρίως στις παραμέτρους του μοντέλου. Στο πλαίσιο αυτό, δοκιμάστηκαν πειραματικά διάφοροι συνδυασμοί για τις εξής παραμέτρους: αριθμός νευρώνων ανά επίπεδο, μήκος εισόδου, ρυθμός εκμάθησης και recurrent dropout. Το recurrent dropout είναι μια τεχνική που χρησιμοποιείται για τη βελτίωση της δυνατότητας γενίκευσης ενός μοντέλου και στηρίζεται στην ιδέα ότι σε κάθε εποχή εκμάθησης ένα σύνολο

νευρώνων απενεργοποιούνται, δηλαδή δεν συμμετέχουν στους υπολογισμούς της προώθησης (forward pass) και οπισθοδρόμησης (backpropagation). Για την πιο συστηματική αναζήτηση καλύτερων παραμέτρων χρησιμοποιήθηκε το πακέτο optuna [36] της python. Η αξιολόγηση έγινε με βάση την απώλεια στο σύνολο επαλήθευσης (validation dataset loss). Ο πίνακας 4.2 συνοψίζει τον βέλτιστο συνδυασμό που εντοπίστηκε.

Παράμετρος	**Τιμή**
Αριθμός LSTM νευρώνων ανά επίπεδο	64
Μήκος εισόδου (history length)	8
Recurrent dropout	0.24
Ρυθμός εκμάθησης (learning rate)	0.0089

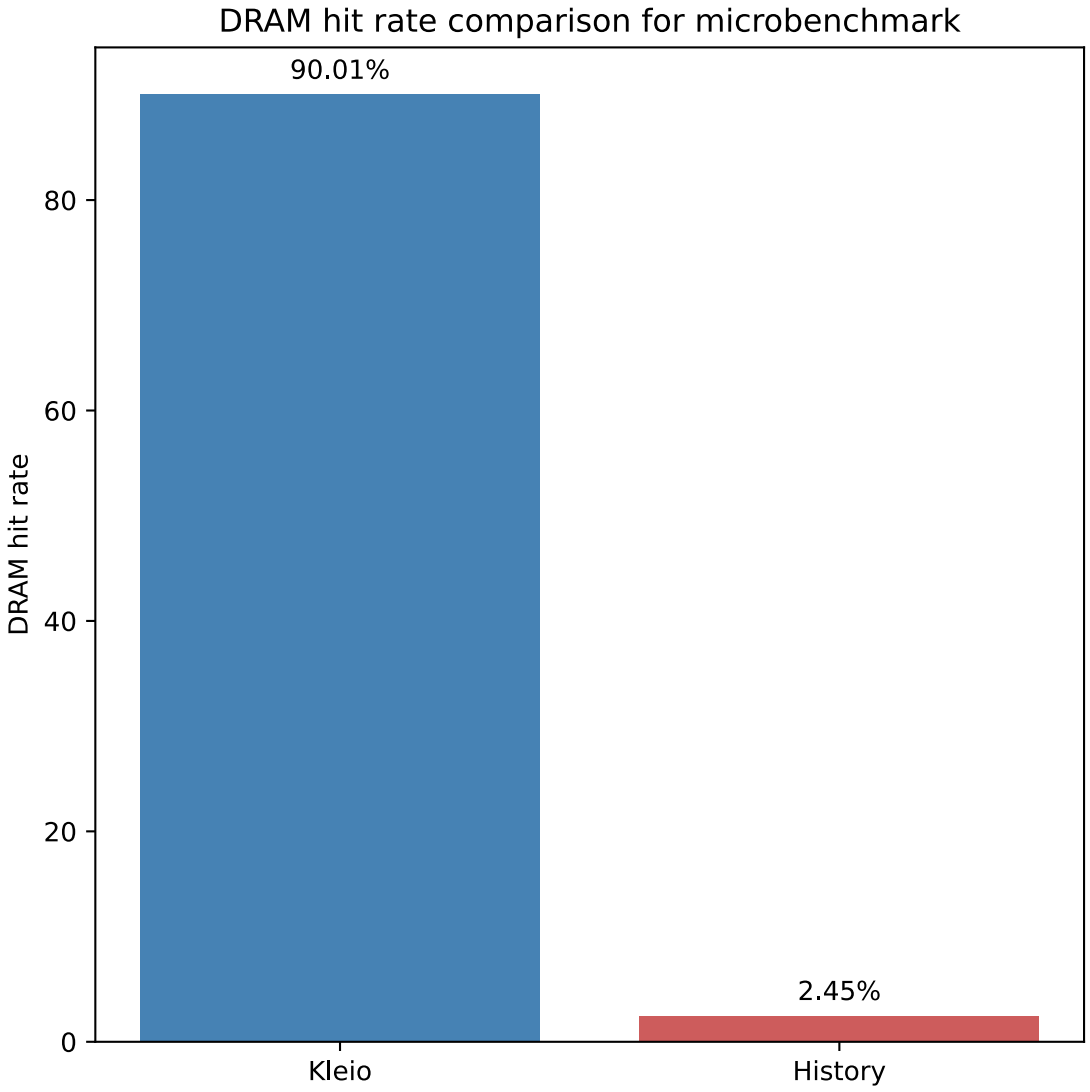
Πίνακας 4.2: *Βέλτιστος συνδυασμός παραμέτρων που εντοπίστηκε με χρήση του πακέτου Optuna*

Για τη συγκεκριμένη παραμετροποίηση των μοντέλων, το ποσοστό ευστοχίας στη DRAM αγγίζει το 90% (σχήμα 4.9). Το ποσοστό αυτό δεν μπορεί να φτάσει το 100% (που εν προκειμένω ταυτίζεται με το ποσοστό που θα επιτύγχανε ο Oracle scheduler), καθώς στις πρώτες history length περιόδους το Kleio δεν χρησιμοποιείται, αφού απαιτείται ιστορικό δεδομένων αντίστοιχου μήκους για να ξεκινήσει τις προβλέψεις. Το συμπέρασμα που προκύπτει είναι ότι η σωστή ρύθμιση των παραμέτρων ενός μοντέλου, με βάσει τις ιδιομορφίες της εκάστοτε εφαρμογής, αποτελεί καθοριστικό παράγοντα για τη βέλτιστη αξιοποίηση των δυνατοτήτων της μηχανικής μάθησης και συμβάλλει στην εξασφάλιση υψηλής ακρίβειας στις προβλέψεις.

4.1.4 Μελέτη Επίδοσης υπό τη Βελτιστοποιημένη Ρύθμιση Μοντέλων

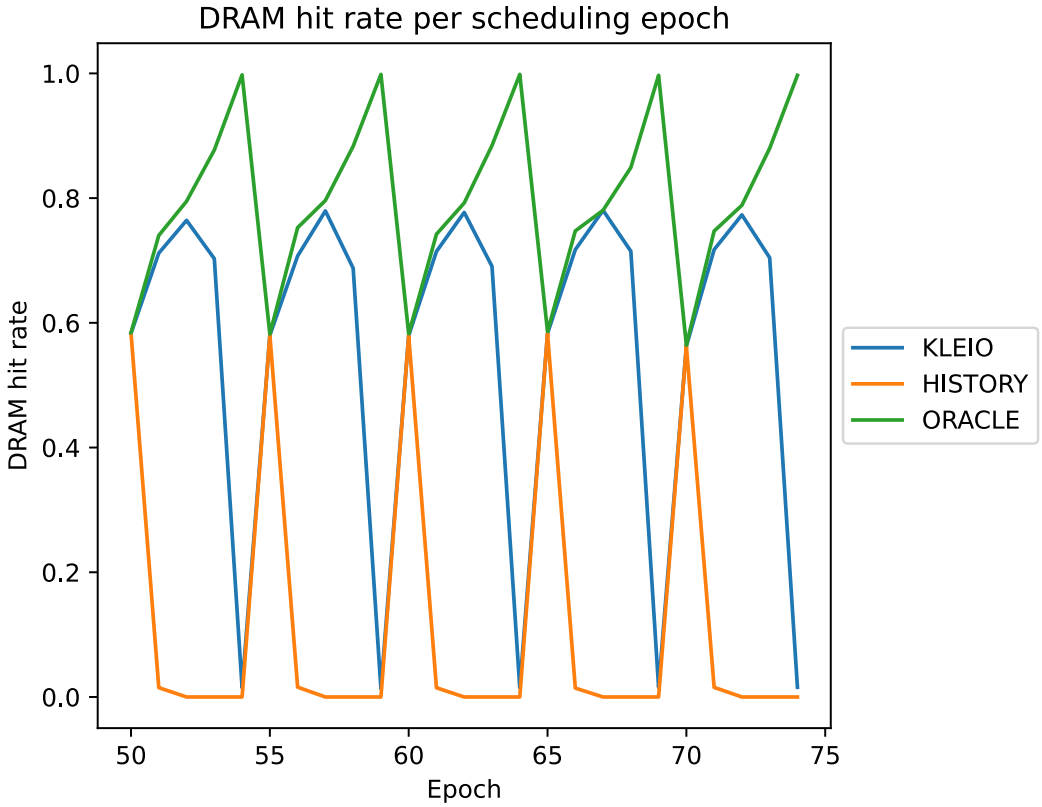
Η παραπάνω ανάλυση επιβεβαιώνει ότι η ενσωμάτωση των μοντέλων στη διαδικασία δρομολόγησης σελίδων μπορεί να βελτιώσει σημαντικά την απόδοση, υπό τις κατάλληλες συνθήκες. Ωστόσο, στην πράξη, οι ιδανικές συνθήκες που δημιουργήθηκαν προηγουμένως είναι σπάνιες. Το σημαντικότερο βήμα για μια πιο ρεαλιστική αξιολόγηση είναι η άρση της ευθυγράμμισης μεταξύ περιόδου δρομολόγησης και επανάληψης του microbenchmark. Για τον λόγο αυτό, χρησιμοποιήθηκε το αρχικό μετροπρόγραμμα, όπου σε κάθε επανάληψη εκτελούνται 10^6 αναγνώσεις για τη δημιουργία αστοχιών κρυφής μνήμης. Αξίζει να σημειωθεί πως στη συγκεκριμένη περίπτωση, ακόμα και ένας δρομολογητής που γνωρίζει εκ των προτέρων με ακρίβεια τις μελλοντικές προσπελάσεις μνήμης (Oracle scheduler) δεν μπορεί να εξασφαλίσει πλήρη επιτυχία, δηλαδή 100% ποσοστό ευστοχίας στη μνήμη DRAM. Αυτό συμβαίνει επειδή, μέσα σε μία και μόνο περίοδο δρομολόγησης, είναι δυνατό να επικαλύπτονται πολλαπλές επαναλήψεις του μετροπρογράμματος. Έτσι, το πλήθος των σελίδων που προσπελαύνονται συνολικά εντός της περιόδου αυτής μπορεί να ξεπεράσει τη διαθέσιμη χωρητικότητα της DRAM και ο δρομολογητής δεν είναι σε θέσει να τις τοποθετήσει όλες στη γρήγορη μνήμη, με αποτέλεσμα ένα ποσοστό αστοχιών να είναι αναπόφευκτο.

Το διάγραμμα της εικόνας 4.10 απεικονίζει το ποσοστό επιτυχίας στη γρήγορη μνήμη DRAM για κάθε περίοδο δρομολόγησης, στο χρονικό διάστημα μεταξύ της 50ης και της 75ης περιόδου, με το ίδιο μοτίβο να επαναλαμβάνεται καθ' όλη τη διάρκεια εκτέλεσης. Τα πειράματα πραγματοποιήθηκαν με την πολιτική εκτέλεσης Kleio, ενώ οι τιμές για τις πολιτικές



Σχήμα 4.9: *Σύγκριση των DRAM hit rate μεταξύ Kleio και History για τη βέλτιστη παραμετροποίηση των μοντέλων*

History και Oracle αντιστοιχούν στα ποσοστά επιτυχίας που θα πετύχαιναν οι αντίστοιχες πολιτικές αν χρησιμοποιούνταν αντί του Kleio στην ίδια εκτέλεση.



Σχήμα 4.10: *DRAM hit rate ανά περίοδο δρομολόγησης*

Εύκολα μπορεί κανείς να διαπιστώσει ότι η χρήση των μοντέλων LSTM υπερισχύει εμφανώς έναντι της πολιτικής History. Αναλυτικότερα, το Kleio επιτυγχάνει μέσο ποσοστό ευστοχίας ίσο με 32%, ενώ ο History scheduler μόλις 0.005%. Παρόλο που το ποσοστό αυτό υπολείπεται σημαντικά της ιδανικής περίπτωσης που παρουσιάστηκε στην προηγούμενη ενότητα, όπου το Kleio άγγιζε το 90% ποσοστό ευστοχίας, θεωρείται απολύτως ικανοποιητικό λαμβάνοντας υπόψη τις τροποποιήσεις που έχουν γίνει στη λειτουργία του μετροπρογράμματος.

Ωστόσο, η επίδοση του Kleio αποκλίνει σημαντικά από τη συμπεριφορά ενός Oracle scheduler, ο οποίος σημειώνει μέσο DRAM hit rate ίσο με 78%. Η μελέτη της διαφοροποίησης μεταξύ των δύο αυτών πολιτικών αποτελεί το κεντρικό θέμα της επόμενης ενότητας.

4.2 Πειραματική αξιολόγηση επίδοσης του δρομολογητή

Στην ενότητα αυτή μελετάται η επίδοση του δρομολογητή σε ένα σύνολο μετροπρογραμμάτων από τις συλλογές PARSEC και SPEC, τα οποία συνοψίζονται στον πίνακα 4.3. Εφόσον δεν ορίζεται ρητά, οι υπόλοιπες παράμετροι —όπως η αναλογία γρήγορης προς αργή μνήμη και η περίοδος δρομολόγησης— ακολουθούν τις ρυθμίσεις που παρουσιάστηκαν στην ενότητα 4.1.

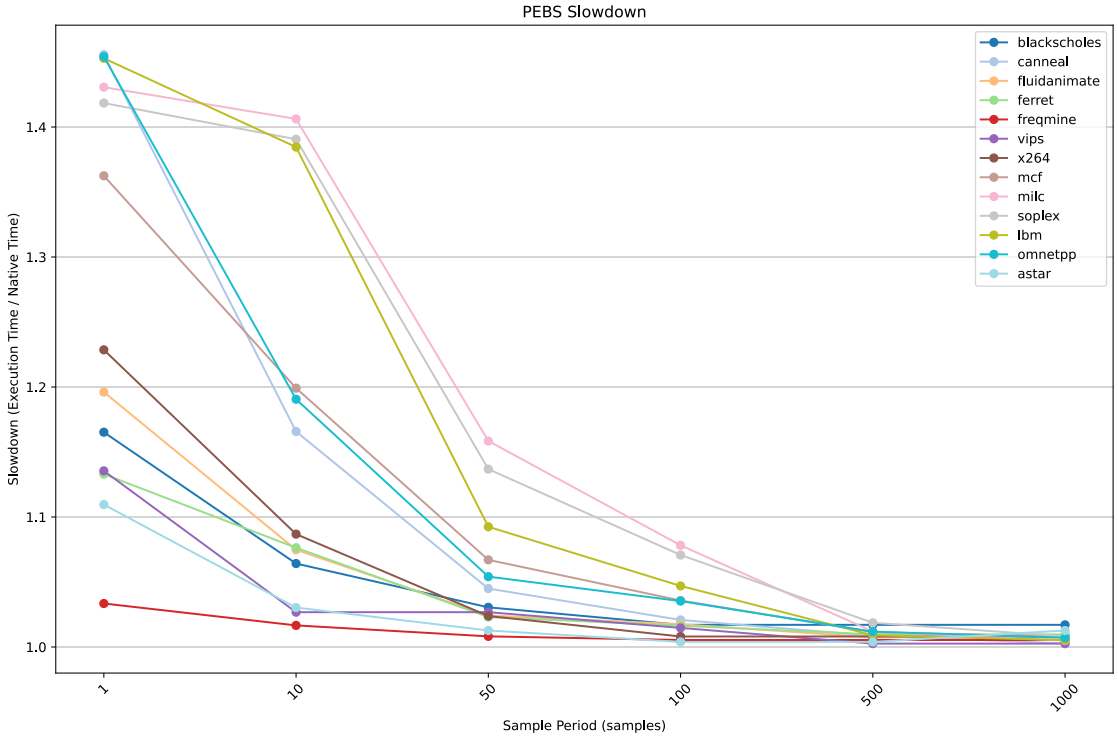
4.2.1 Ρύθμιση παραμέτρων δειγματοληψίας

Όπως έγινε και με το microbenchmark της προηγούμενης ενότητας, πρέπει αρχικά να προσδιοριστεί το μέγεθος των απαιτήσεων και να ρυθμιστούν οι παράμετροι της δειγματοληψίας για κάθε μετροπρόγραμμα. Ο μέγιστος όγκος μνήμης που χρησιμοποιείται σε κάθε benchmark μπορεί εύκολα να βρεθεί καταγράφοντας τη μεγαλύτερη τιμή που εντοπίζεται στη στήλη RES της εντολής `htop` κατά τη διάρκεια εκτέλεσης της εφαρμογής. Η κατάλληλη περίοδος δειγματοληψίας επιλέχθηκε δοκιμάζοντας διαφορετικές τιμές sample period στο εύρος 1–1000, καταγράφοντας την αντίστοιχη επιβράδυνση. Η τελική επιλογή έγινε με βάση δύο κριτήρια: η επιβράδυνση να μην υπερβαίνει το 10% και να μην παρατηρούνται φαινόμενα throttling. Η επιβράδυνση και ο αριθμός των CPU throttling που οφείλονται στη δειγματοληψία απεικονίζονται στα διαγράμματα 4.11 και 4.12.

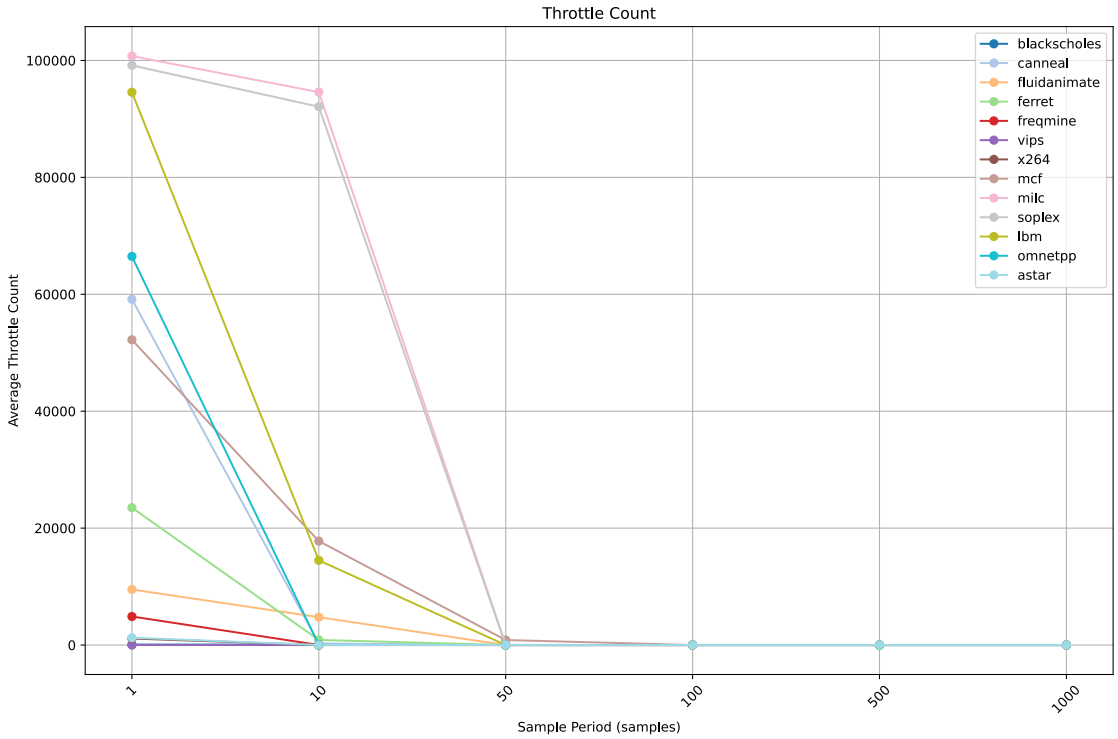
Ο πίνακας 4.3 παρουσιάζει τις τιμές περιόδου δειγματοληψίας που επιλέχθηκαν για κάθε μετροπρόγραμμα καθώς και τον συνολικό αριθμό σελίδων (για μεγέθη σελίδας 4KB και 2MB). Επιπλέον, ένας κυκλικός πίνακας ring buffer μεγέθους 8MB αρκεί για να καλύψει τις ανάγκες της δειγματοληψίας χωρίς να υπάρχει απώλεια εγγραφών.

Suite	Benchmark	Pages (4KB)	Pages (2MB)	Sample Period
PARSEC	blackscholes	172800	338	10
PARSEC	canneal	272896	533	500
PARSEC	fluidanimate	148736	291	50
PARSEC	ferret	25344	50	50
PARSEC	freqmine	158720	310	10
PARSEC	vips	37120	73	10
PARSEC	x264	20992	41	50
SPEC	mcf	429056	838	100
SPEC	milc	173824	340	500
SPEC	soplex	110848	217	500
SPEC	lbm	104704	205	100
SPEC	omnetpp	43520	85	50
SPEC	astar	59648	117	10

Πίνακας 4.3: *Αριθμός συνολικών σελίδων και επιλεγμένη τιμή περιόδου δειγματοληψίας για κάθε benchmark.*



Σχήμα 4.11: *Επιβράδυνση συναρτήσει περιόδου δειγματοληψίας*

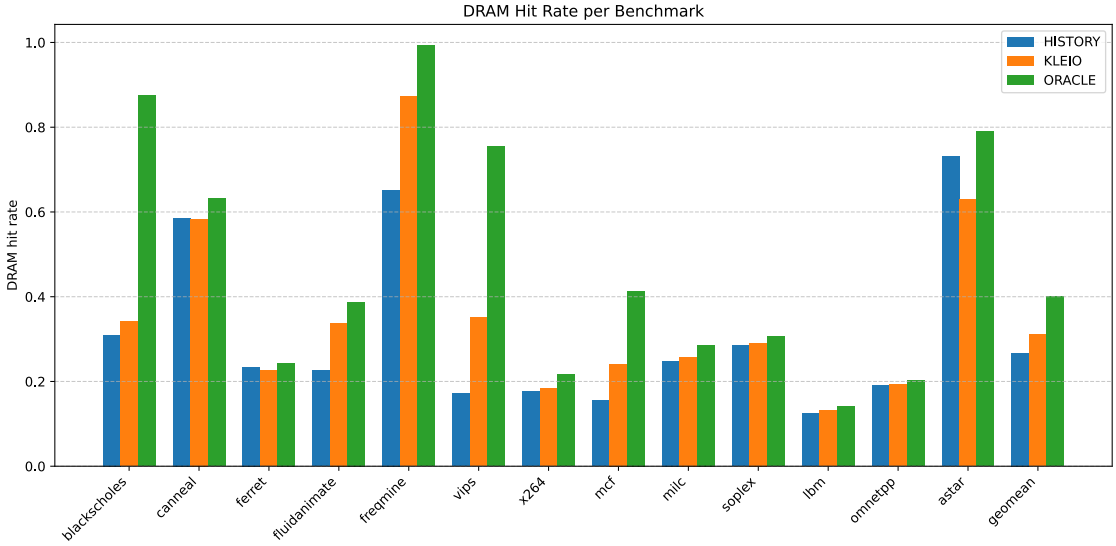


Σχήμα 4.12: *Μέσος αριθμός CPU throttling events ανά περίοδο δρομολόγησης συναρτήσει περιόδου δειγματοληψίας*

4.2.2 Αξιολόγηση επίδοσης

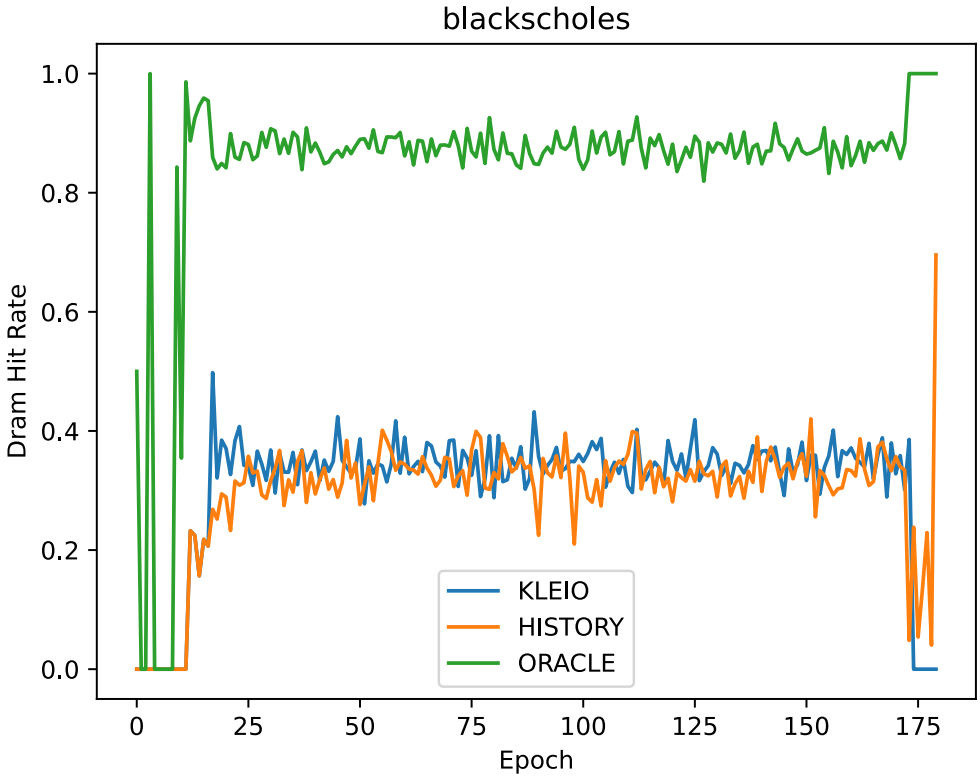
Η χρήση σελίδων μεγέθους 4KB εξετάστηκε εκ νέου για τα συγκεκριμένα μετροπρογράμματα και αποδείχθηκε για ακόμη μια φορά ακατάλληλη, καθώς το πρόβλημα των αραιών ακολουθιών προσβάσεων παρέμεινε αρκετά έντονο, εμποδίζοντας την εκπαίδευση των μοντέλων. Για τον λόγο αυτό, αποφασίστηκε η ενεργοποίηση του μηχανισμού των Transparent Huge Pages σε όλα τα μετροπρογράμματα. Ως βασικό κριτήριο αξιολόγησης των πειραμάτων θεωρήθηκε και πάλι η μετρική DRAM hit rate που αντιπροσωπεύει το ποσοστό εντολών ανάγνωσης ανά περίοδο δρομολόγησης που αστοχούν όλα τα επίπεδα κρυφής μνήμης αλλά εξυπηρετούνται από τη DRAM. Οι παράμετροι των μοντέλων ρυθμίστηκαν σύμφωνα με τις τιμές που προτείνουν οι δημιουργοί του Kleio, ενώ η αναλογία γρήγορης προς αργή μνήμη ορίστηκε ίση με 1/8. Όπως και στην προηγούμενη ενότητα, παρόλο που τα πειράματα εκτελούνται μόνο με πολιτική δρομολόγησης Kleio, συλλέγεται και η πληροφορία σχετικά με την επίδοση των εναλλακτικών αποφάσεων που θα έπαιρναν οι πολιτικές History και Oracle. Κάθε μετροπρόγραμμα εκτελέστηκε 10 φορές προκειμένου να εντοπιστούν πιθανές μεταβολές στα αποτελέσματα, ωστόσο η παρατηρούμενη μεταβλητότητα ήταν αμελητέα· συνεπώς, τα αποτελέσματα που παρουσιάζονται στη συνέχεια, καθώς και στην επόμενη ενότητα, προέρχονται από μία ενδεικτική εκτέλεση, η οποία κρίνεται αντιπροσωπευτική.

Στο συγκεντρωτικό διάγραμμα της εικόνας 4.13 παρουσιάζεται ο γεωμετρικός μέσος όρος της ευστοχίας γρήγορης μνήμης κάθε μετροπρογράμματος για τις πολιτικές Kleio, History και Oracle.

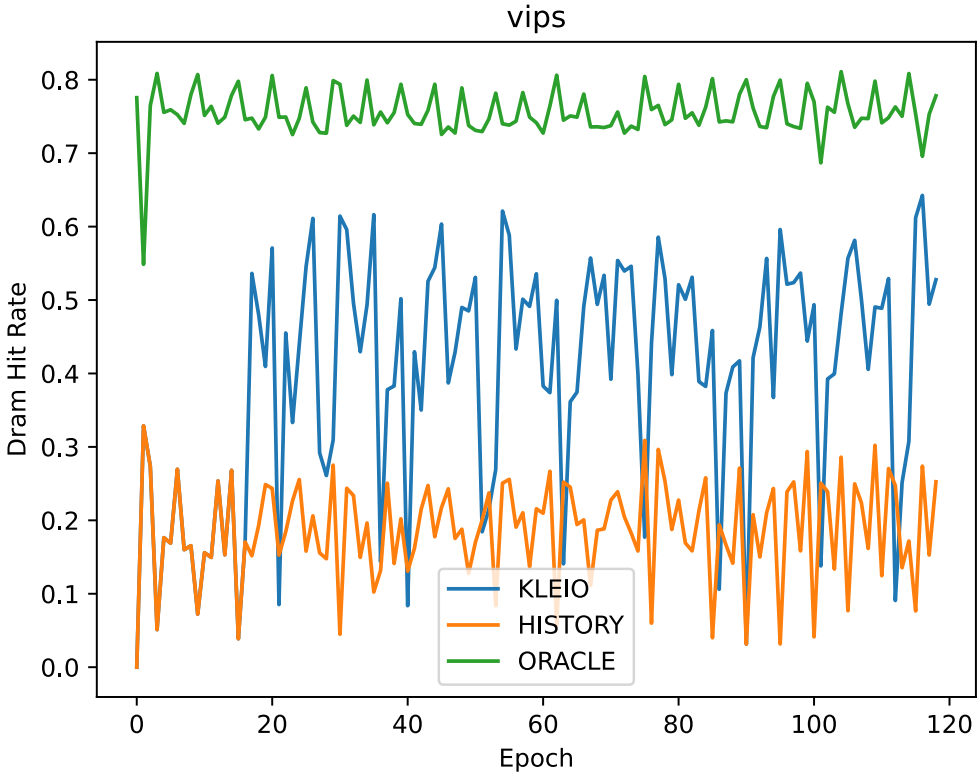


Σχήμα 4.13: *Γεωμετρικός μέσος όρος του DRAM hit rate για κάθε μετροπρόγραμμα. Το geomean αποτυπώνει τον συνολικό γεωμετρικό μέσο όρο για κάθε πολιτική δρομολόγησης.*

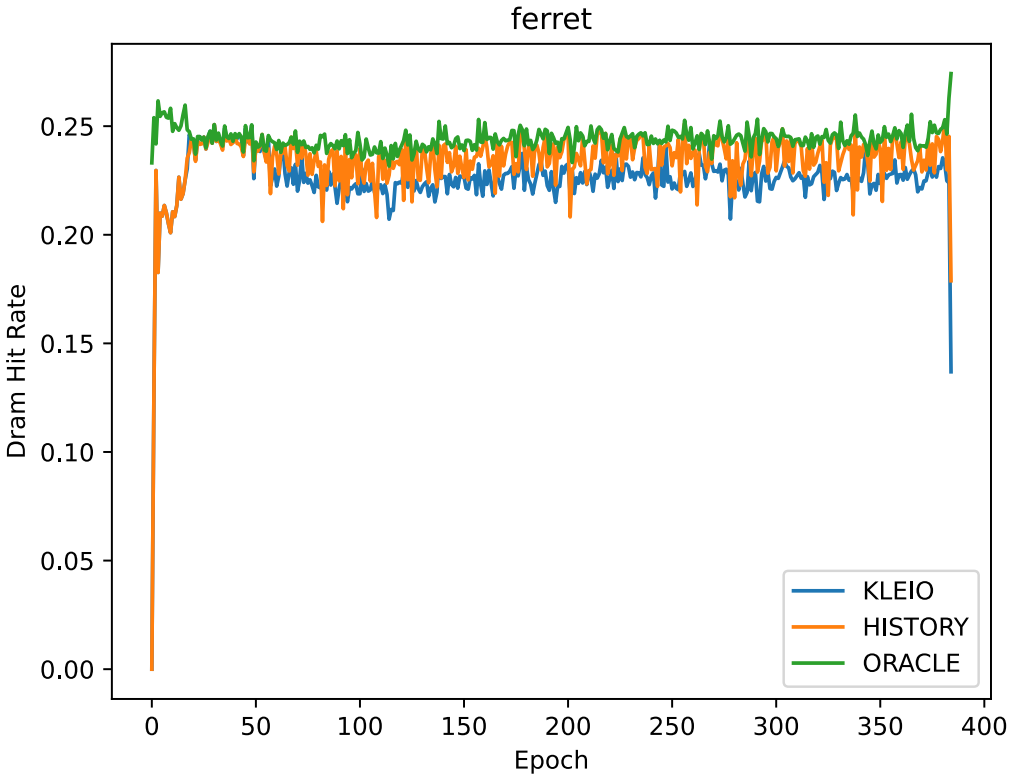
Στην πλειονότητα των περιπτώσεων, το Kleio ξεπερνά σε απόδοση τον κλασικό History scheduler, προσφέροντας κατά μέσο όρο **βελτίωση 15.95%** στο ποσοστό ευστοχίας στη γρήγορη μνήμη. Τα παρακάτω διαγράμματα παρουσιάζουν με ακρίβεια το ποσοστό ευστοχίας ανά περίοδο δρομολόγησης για ορισμένα μετροπρογράμματα.



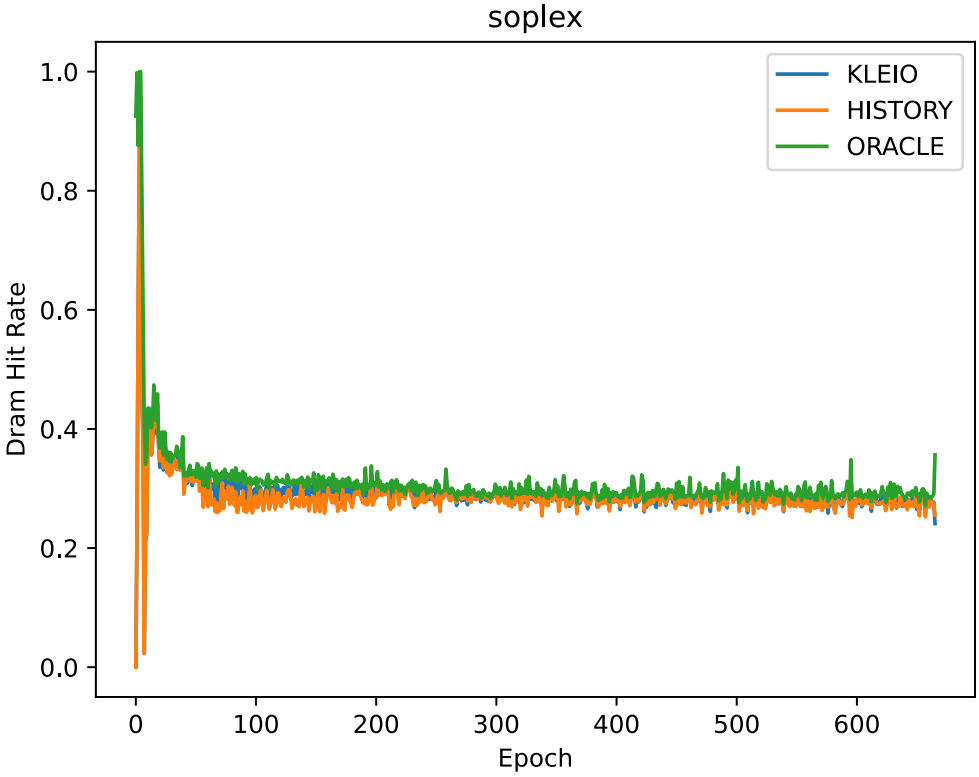
Σχήμα 4.14: *DRAM hit rate ανά περίοδο δρομολόγησης του blackscholes*



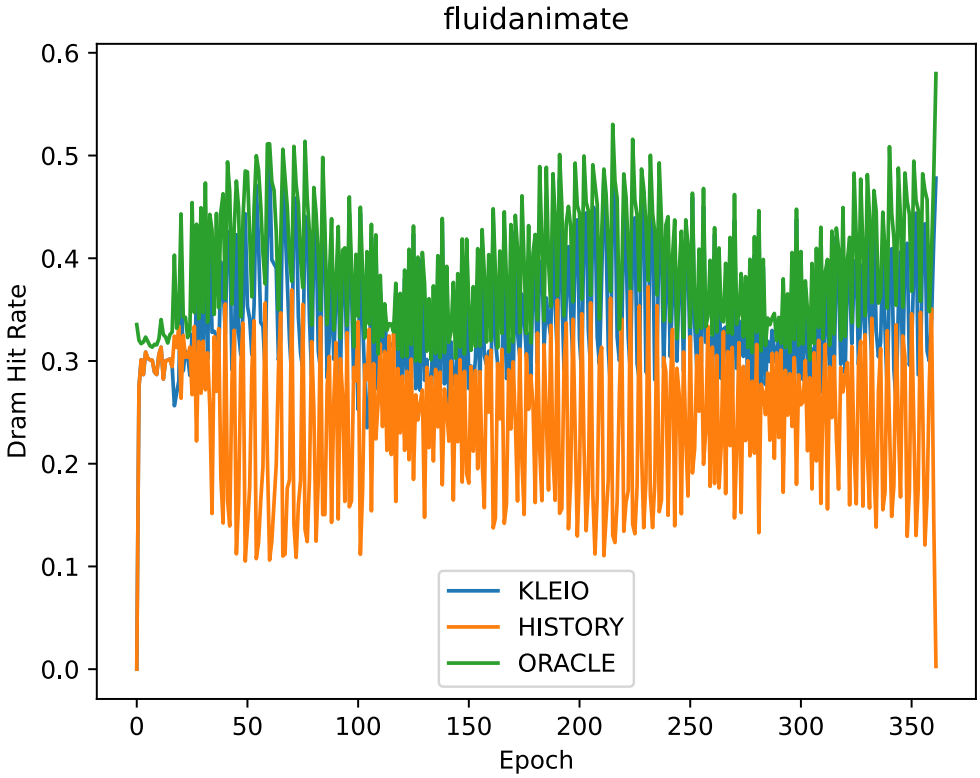
Σχήμα 4.15: *DRAM hit rate ανά περίοδο δρομολόγησης του vips*



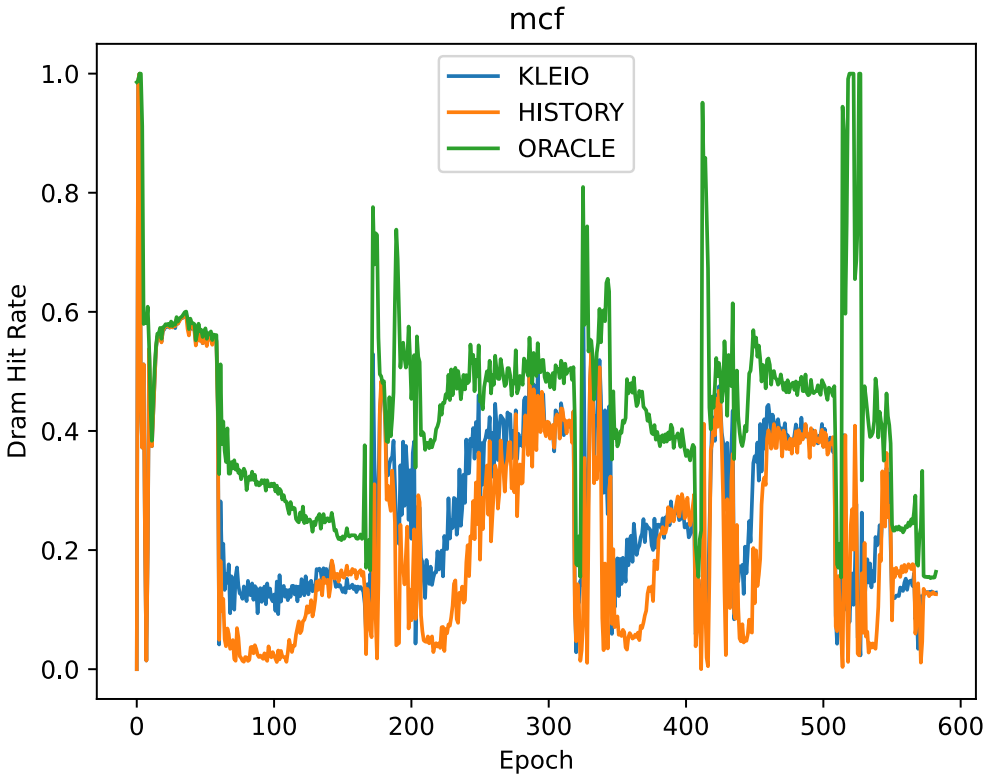
Σχήμα 4.16: *DRAM hit rate ανά περίοδο δρομολόγησης του ferret*



Σχήμα 4.17: *DRAM hit rate ανά περίοδο δρομολόγησης του soplex*



Σχήμα 4.18: *DRAM hit rate ανά περίοδο δρομολόγησης του fluidanimate*



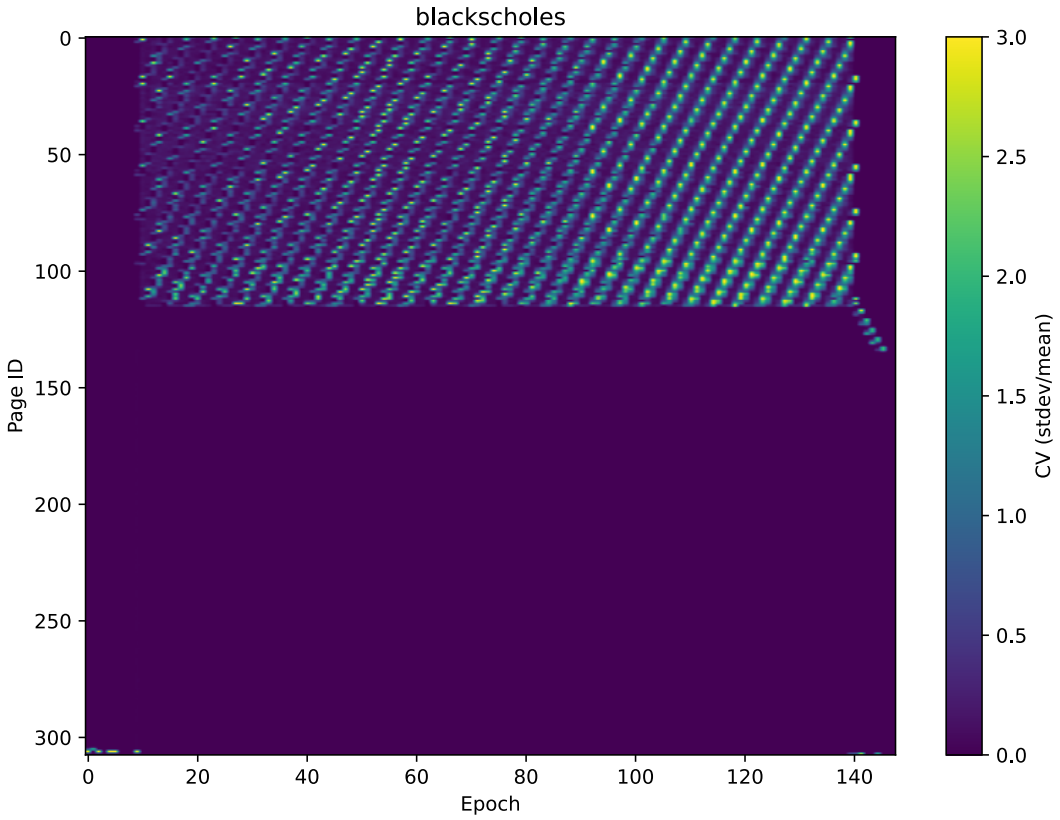
Σχήμα 4.19: *DRAM hit rate ανά περίοδο δρομολόγησης του mcf*

Για το blackscholes (διάγραμμα 4.14), η χρήση των μοντέλων οδηγεί σε υψηλότερα ποσοστά ευστοχίας στις περισσότερες περιόδους δρομολόγησης σε σχέση με τον History scheduler. Παρόλα αυτά, η απόδοσή του Kleio απέχει σημαντικά από την ιδανική, καθώς ο Oracle δρομολογητής επιτυγχάνει υπερδιπλάσιο DRAM hit rate. Το vips (σχήμα 4.15) είναι ίσως η πιο χαρακτηριστική περίπτωση στην οποία φαίνεται πως το Kleio μπορεί πράγματι να γεφυρώσει εμφανώς το χάσμα μεταξύ δρομολογητή και ιστορικού δρομολογητή. Αντίθετα, στο ferret (σχήμα 4.16), η δρομολόγηση με βάση το ιστορικό τελευταίας πρόσβασης υπερτερεί ελαφρώς του Kleio, αποτελώντας μία από τις λίγες περιπτώσεις όπου αυτή η προσέγγιση αποδεικνύεται πιο αποτελεσματική. Ωστόσο, η διαφορά στην απόδοση είναι μικρή και δεν θεωρείται ιδιαίτερα σημαντική. Στο διάγραμμα 4.17, παρατηρείται ότι και οι τρεις πολιτικές δρομολόγησης επηρεάζουν σχεδόν στον ίδιο βαθμό το DRAM hit rate για το μετροπρόγραμμα soplex, χωρίς να παρουσιάζεται ουσιαστική διαφοροποίηση μεταξύ τους. Τέλος, ιδιαίτερο ενδιαφέρον παρουσιάζουν οι περιπτώσεις των fluidanimate και mcf, στα οποία εμφανίζεται σημαντική διακύμανση του ποσοστού ευστοχίας σε κάθε περίοδο.

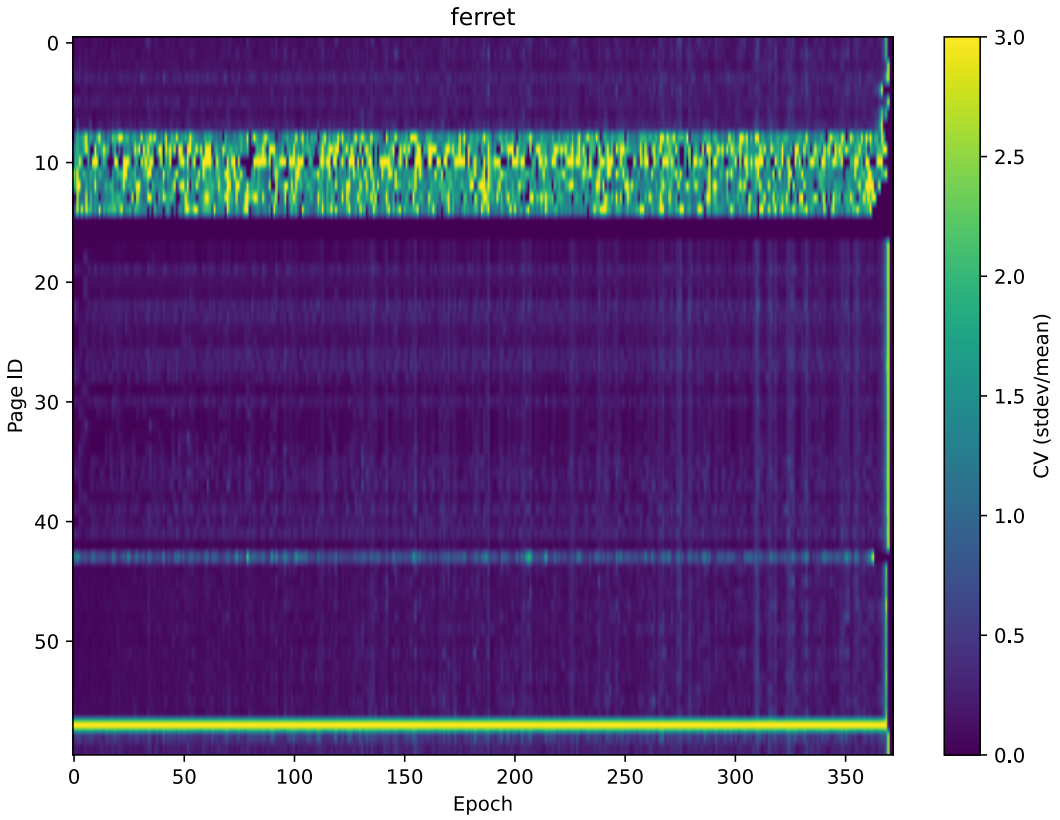
Το βασικό ζήτημα που αναδεικνύεται από την παραπάνω ανάλυση είναι η μεγάλη απόκλιση στην απόδοση μεταξύ του Kleio και του ιδανικού δρομολογητή Oracle, μια διαφορά που σε αρκετές περιπτώσεις αποδεικνύεται ιδιαίτερα έντονη. Μια εύλογη υπόθεση για την αιτία αυτού του φαινομένου είναι η ανεπαρκής ρύθμιση των παραμέτρων κάθε μοντέλου. Ωστόσο, όσες δοκιμές έγιναν με διαφορετική παραμετροποίηση δεν είχαν ως αποτέλεσμα σημαντική αύξηση του ρυθμού ευστοχίας.

Η πραγματική αιτία του προβλήματος εντοπίζεται πιθανότατα στη μεταβλητότητα που εισάγει η διαδικασία της δειγματοληψίας. Για την επαλήθευση του ισχυρισμού, καταγράφηκαν οι ακολουθίες προσπελάσεων όλων των σελίδων σε 10 διαδοχικές εκτελέσεις για κάθε benchmark. Η αξιολόγηση της μεταβλητότητας έγινε με βάση τον **συντελεστή μεταβλητότητας** (coefficient of variation) [37] που ορίζεται ως το πηλίκο της τυπικής απόκλισης προς τη μέση τιμή και εκφράζει τη σχετική διακύμανση ενός συνόλου δεδομένων. Παρακάτω παρουσιάζεται ο συντελεστής μεταβλητότητας ανά σελίδα και ανά περίοδο με τη βοήθεια χρωματικών χαρτών για ορισμένα benchmarks.

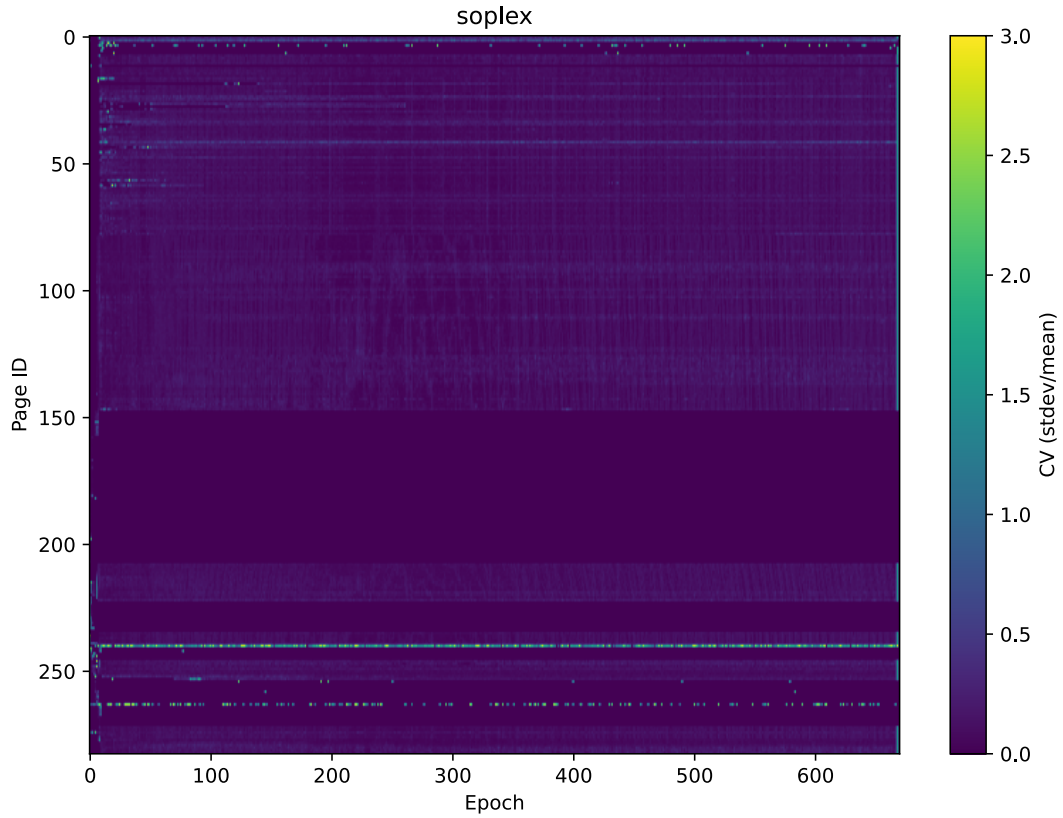
Στο blackscholes (σχήμα 4.20), περίπου το 1/3 των συνολικών σελίδων εμφανίζει έντονη μεταβλητότητα. Αντίστοιχη συμπεριφορά παρουσιάζει και το ferret (σχήμα 4.21) με τη μόνη διαφορά ότι το ποσοστό των σελίδων με έντονη διακύμανση είναι μικρότερο σε σύγκριση με το blackscholes. Η έντονη διακύμανση που παρατηρείται σε ένα υποσύνολο σελίδων εξηγεί σε μεγάλο βαθμό τη διαφορά που εμφανίζεται μεταξύ της καμπύλης του Kleio και του Oracle στα διαγράμματα 4.14 και 4.16, καθώς τα μοντέλα δυσκολεύονται να προβλέψουν με ακρίβεια τη συμπεριφορά αυτών των σελίδων. Από την άλλη, στον αντίστοιχο χρωματικό χάρτη μεταβλητότητας του soplex (σχήμα 4.22), είναι εμφανές πως η συντριπτική πλειονότητα των σελίδων εμφανίζει μικρή μεταβλητότητα γεγονός που έρχεται σε πλήρη συμφωνία με τη μορφή του διαγράμματος 4.17.



Σχήμα 4.20: *Μεταβλητότητας του blackscholes*



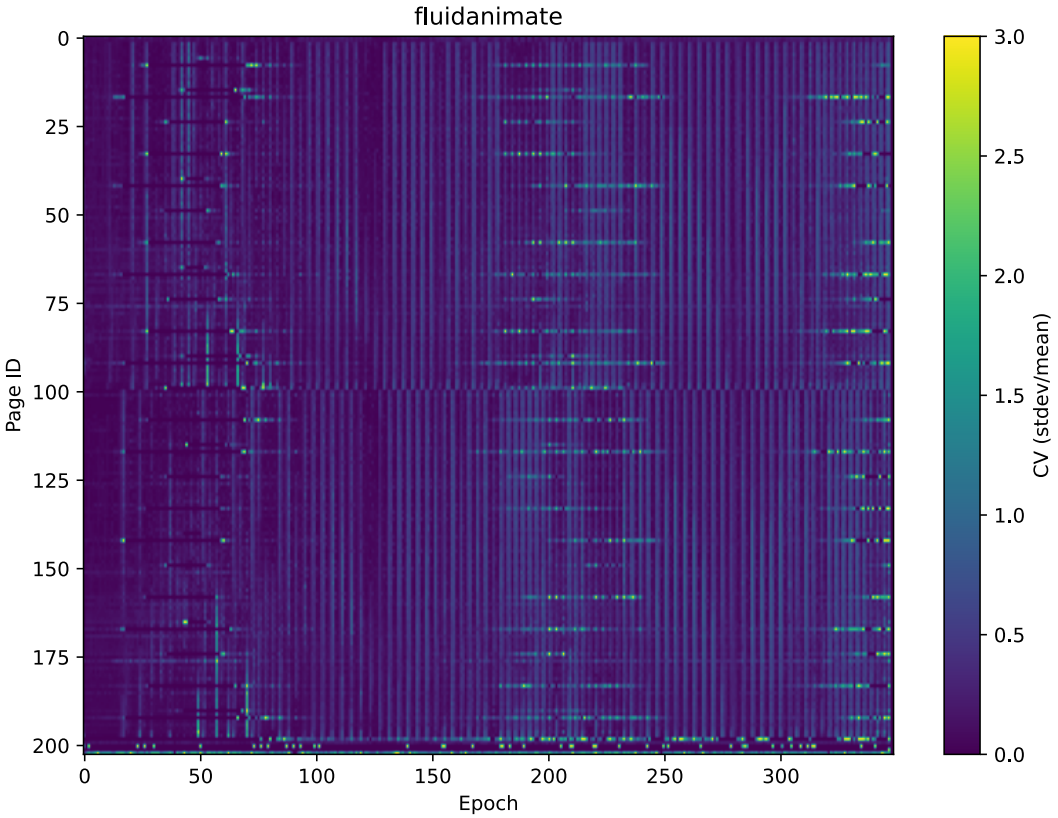
Σχήμα 4.21: *Μεταβλητότητα του ferret*



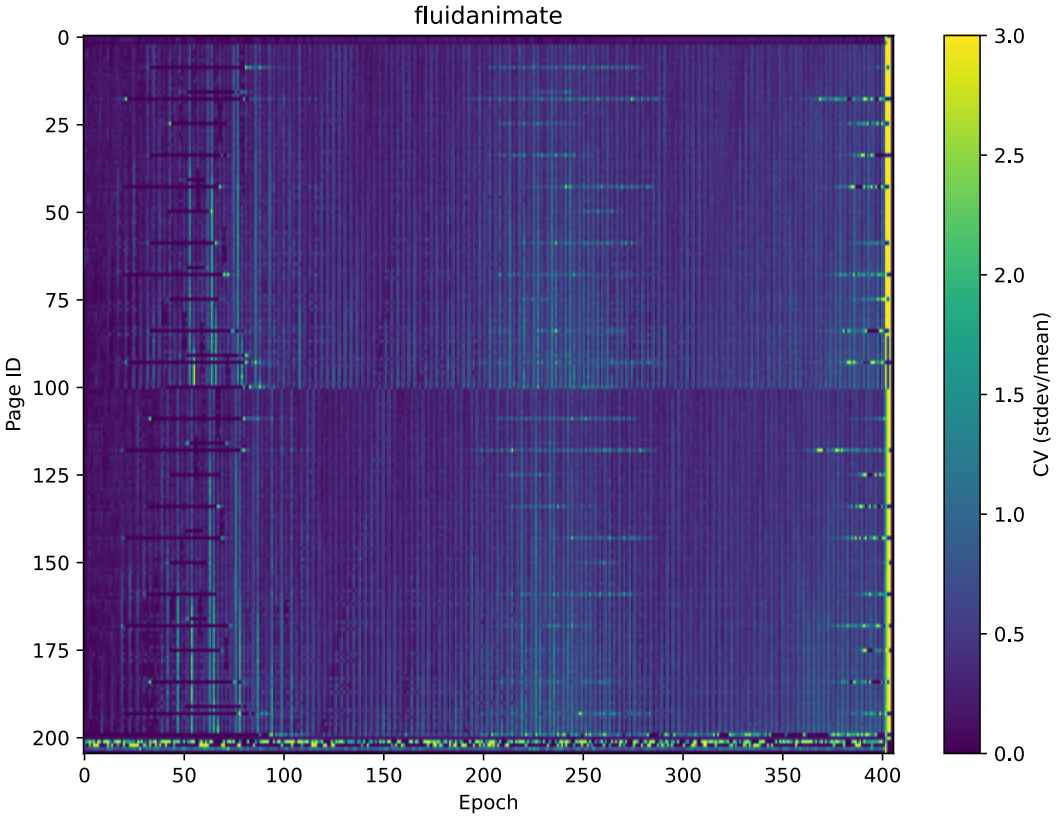
Σχήμα 4.22: *Μεταβλητότητα του soplex*

Το συμπέρασμα που συνάγεται μέχρι στιγμής είναι πως η αστάθεια που επιφέρει ο μηχανισμός δειγματοληψίας στις καταγεγραμμένες προσπελάσεις μνήμης δυσχεραίνει την εκμάθηση των LSTM και εντείνει το χάσμα του Kleio σε σύγκριση με τον ιδανικό δρομολογητή. Ο στόχος είναι η διερεύνηση των πιθανών λύσεων για την αντιμετώπιση των επιπτώσεων που προκαλεί η διακύμανση της δειγματοληψίας. Προς αυτήν την κατεύθυνση, επιλέχθηκαν τα fluidanimate και blackscholes ως αντιπροσωπευτικά μετροπρόγραμμα για περαιτέρω ανάλυση.

Για το fluidanimate, όπως δείχνει το σχήμα 4.23, η μεταβλητότητα των προσπελάσεων σε κάθε σελίδα του είναι έντονη. Μία πιθανή εξήγηση για το ζήτημα αυτό είναι ότι προκύπτει από εσφαλμένη ρύθμιση των παραμέτρων δειγματοληψίας και ότι μια αύξηση της συχνότητας λήψης δειγμάτων θα μπορούσε να μειώσει τις διακυμάνσεις. Για τον λόγο αυτό, η περίοδος δειγματοληψίας μειώθηκε από 50 σε 10. Ωστόσο, τα δεδομένα που παρουσιάζονται στο σχήμα 4.24 αποδεικνύουν το αντίθετο, καθώς η συχνότερη δειγματοληψία συνοδεύεται από μεγαλύτερη συνολική μεταβλητότητα. Επιπλέον, ο πίνακας 4.4 παρέχει ένα ενδεικτικό παράδειγμα, καταγράφοντας τη διακύμανση του αριθμού προσπελάσεων μιας τυχαίας σελίδας μεταξύ δύο διαδοχικών εκτελέσεων.



Σχήμα 4.23: *Μεταβλητότητα του fluidanimate για περίοδο δειγματοληψίας 50*



Σχήμα 4.24: *Μεταβλητότητα του fluidanimate για περίοδο δειγματοληψίας 10*

Περίοδος δρομολόγησης	1η Εκτέλεση	2η Εκτέλεση	Μεταβολή (%)
241	4175	6410	+53.46%
356	5221	7305	+39.87%
348	5235	7249	+38.44%
333	7282	5117	−29.74%
369	7255	5068	−30.13%
347	7512	5186	−30.97%

Πίνακας 4.4: *Μεταβολή α ιθμού π οσπελάσεων μεταξύ δύο διαδοχικών εκτελέσεων*

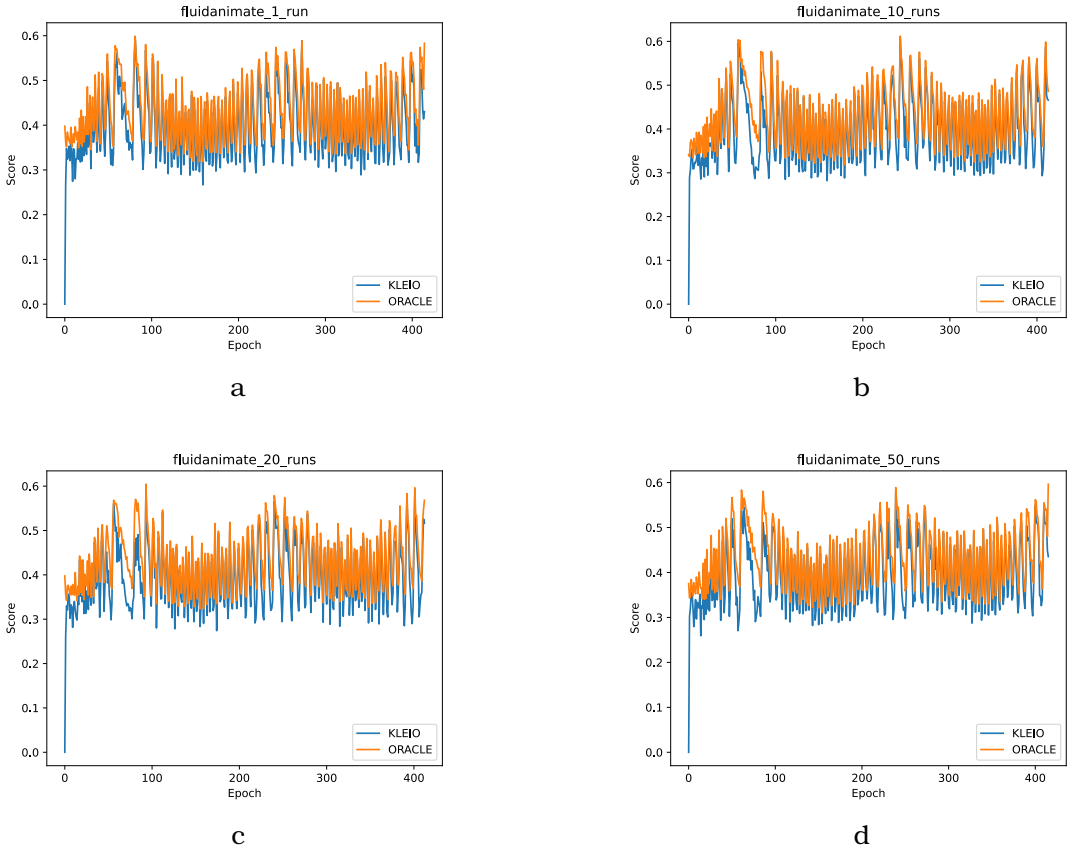
Η τελευταία και ίσως η πιο εύλογη προσέγγιση που επιχειρήθηκε για τον μετριασμό των προβλημάτων που προκαλεί η μεταβλητότητα της δειγματοληψίας είναι η εκπαίδευση των μοντέλου χρησιμοποιώντας δεδομένα από πολλαπλές εκτελέσεις. Συγκεκριμένα, δοκιμάστηκε η εκπαίδευση με σύνολα δεδομένων από 10, 20 και 50 εκτελέσεις. Ωστόσο, τα αποτελέσματα τόσο για το fluidanimate όσο και για το blackscholes δείχνουν ότι αυτή η προσέγγιση δεν επαρκεί για να γεφυρωθεί το χάσμα στην ακρίβεια της κρυφής μνήμης μεταξύ του Oracle και του History, γεγονός που επιβεβαιώνεται τόσο από τους πίνακες 4.5 και 4.6 όσο και από τα διαγράμματα 4.25 και 4.26.

Αριθμός εκτελέσεων	Kleio DRAM hit rate	Oracle DRAM hit rate	Διαφορά (%)
1	0.3904	0.4281	−8.81%
10	0.3919	0.4254	−7.86%
20	0.3915	0.4267	−8.26%
50	0.3896	0.4250	−8.33%

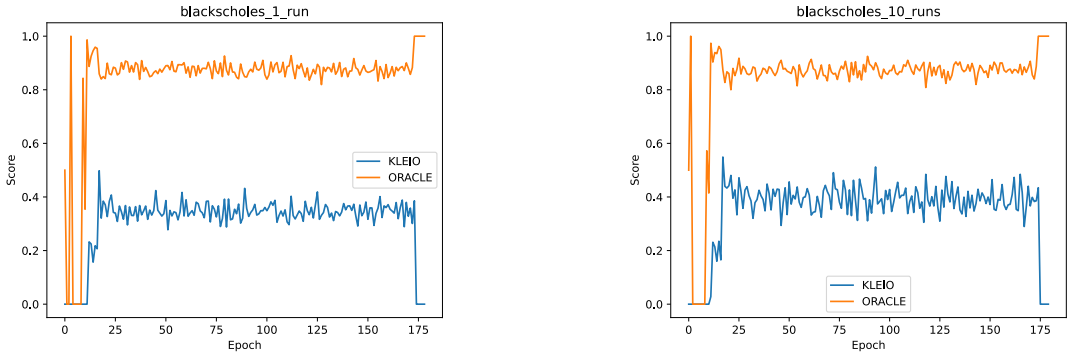
Πίνακας 4.5: *Γεωμετρικός μέσος όρος του ποσοστού ευστοχίας στη γρήγορη μνήμη για το fluidanimate*

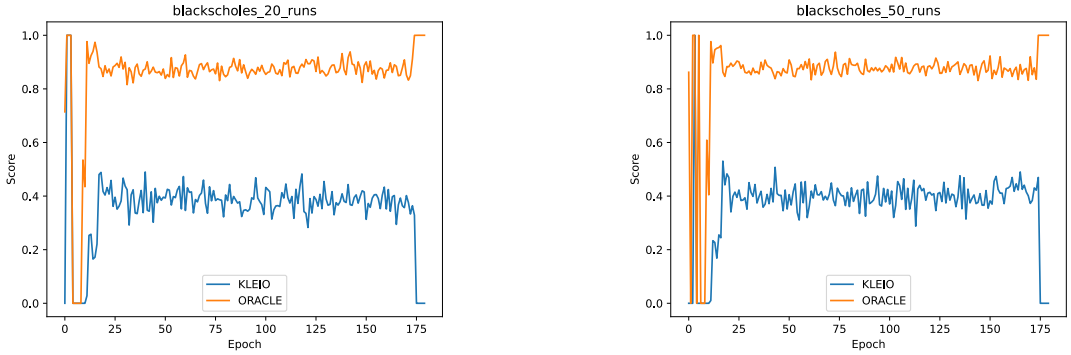
Αριθμός εκτελέσεων	Kleio DRAM hit rate	Oracle DRAM hit rate	Διαφορά (%)
1	0.3412	0.874	−60.96%
10	0.3791	0.871	−56.48%
20	0.3784	0.872	−56.61%
50	0.388	0.876	−55.71%

Πίνακας 4.6: *Γεωμετρικός μέσος όρος του ποσοστού ευστοχίας στη γρήγορη μνήμη για το blackscholes*



Σχήμα 4.25: *Benchmark: fluidanimate Διαγράμματα DRAM hit rate ανά περίοδο δρομολόγησης για σύνολα δεδομένων που προέρχονται από 1 (σχήμα a), 10 (σχήμα b), 20 (σχήμα c) και 50 (σχήμα d) εκτελέσεις*





Σχήμα 4.26: *Benchmark: blackscholes Διαγράμματα DRAM hit rate ανά περίοδο δρομολόγησης για σύνολα δεδομένων που προέρχονται από 1 (σχήμα a), 10 (σχήμα b), 20 (σχήμα c) και 50 (σχήμα d) εκτελέσεις*

Μέρος III

Επίλογος

Κεφάλαιο 5

Συμπεράσματα

Στο κεφάλαιο αυτό γίνεται η τελική αποτίμηση της επίδοσης του δρομολογητή, παρουσιάζονται τα κεντρικά συμπεράσματα της ανάλυσης και εξετάζονται πιθανές μελλοντικές κατευθύνσεις για τη βελτίωση και την περαιτέρω ανάπτυξή του.

5.1 Συνολική αξιολόγηση της υλοποίησης

Το βασικότερο συμπέρασμα της παρούσας διπλωματικής είναι ότι η υλοποίηση ενός δρομολογητή σελίδων βασισμένου σε μεθόδους μηχανικής μάθησης για ένα πραγματικό σύστημα υβριδικής μνήμης συνοδεύεται από σημαντικές προκλήσεις και κρίσιμες σχεδιαστικές επιλογές. Αρχικά, ο υπερβολικά υψηλός χρόνος που απαιτείται για την παραγωγή προβλέψεων (inference) από τα μοντέλα καθιστά αναγκαία την εισαγωγή μηχανισμών παύσης και επανεκκίνησης της εφαρμογής από τον δρομολογητή. Το γεγονός αυτό αναδεικνύει ότι η ενσωμάτωση νευρωνικών δικτύων σε διαδικασίες όπως η δρομολόγηση σελίδων οι οποίες απαιτούν εξαιρετικά χαμηλή καθυστέρηση απόκρισης δεν είναι ακόμη επαρκώς ώριμη για χρήση σε πραγματικές συνθήκες.

Η σημαντικότερη όμως σχεδιαστική απόφαση που υιοθετήθηκε στο πλαίσιο της παρούσας εργασίας αφορά την επιλογή της δειγματοληψίας (PEBS) ως την πλέον κατάλληλη μέθοδο για τη συλλογή δεδομένων σχετικά με προσβάσεις μνήμης κατά τον χρόνο εκτέλεσης, βάσει των οποίων πραγματοποιείται η εκπαίδευση των μοντέλων. Ωστόσο, αν και η χρήση των μετρητών επίδοσης που παρέχουν οι σύγχρονοι επεξεργαστές είναι η πιο συμφέρουσα λύση από πλευράς ακρίβειας και ταχύτητας, αποτελεί συγχρόνως πηγή ορισμένων σημαντικών προβλημάτων. Αρχικά, η εφαρμογή του PEBS σε σελίδες μεγέθους 4KB, σε συνδυασμό πάντα με την κατάλληλη επιλογή περιόδου δειγματοληψίας για την αποφυγή υπερφόρτωσης του συστήματος, έχει ως αποτέλεσμα την καταγραφή εξαιρετικά αραιών ακολουθιών. Σε αυτές τις περιπτώσεις, η πλειονότητα των σελίδων παρουσιάζει μηδενικές προσβάσεις κατά τις περισσότερες περιόδους δρομολόγησης, καθιστώντας έτσι την εκπαίδευση των μοντέλων LSTM ανεπαρκή. Η μετάβαση σε σελίδες μεγέθους 2MB φαίνεται να αντιμετωπίζει αποτελεσματικά το συγκεκριμένο ζήτημα, ωστόσο εισάγει σημαντική μεταβλητότητα στα συλλεγόμενα δείγματα, καθώς ακόμη και δύο διαδοχικές εκτελέσεις της ίδιας εφαρμογής ενδέχεται να διαθέτουν αισθητά διαφορετικά δεδομένα δειγματοληψίας. Το γεγονός αυτό λειτουργεί ως ανασταλτικός παράγοντας στην διαδικασία εκμάθησης των μοντέλων και αποτελεί τη βασικότερη αιτία για την οποία η χρήση του Kleio δεν μπορεί να συγκλίνει στη συμπεριφορά ενός ιδανικού

δρομολογητή με πρότερη γνώση για τις προσβάσεις μνήμης. Ακόμα και η εκπαίδευση των μοντέλων βάσει δεδομένων από πολλαπλές εκτελέσεις δεν φάνηκε ικανή για να βελτιώσει την επίδοση του Kleio.

Παρόλα αυτά, η υλοποίηση, με τις παραπάνω παραδοχές και σχεδιαστικές επιλογές, έδειξε ότι το Kleio μπορεί να προσφέρει αισθητά καλύτερη επίδοση σε σύγκριση με τον κλασικό History scheduler. Σε ένα τεχνητά διαμορφωμένο microbenchmark, το Kleio πέτυχε έως και 23 φορές υψηλότερη απόδοση από τον History scheduler, ενώ με κατάλληλη ρύθμιση των υπερπαραμέτρων η διαφορά αυτή αυξάνεται σε 36.7, προσεγγίζοντας σε μεγάλο βαθμό τη συμπεριφορά ενός ιδανικού Oracle scheduler. Ακόμη, όταν οι ιδανικές συνθήκες χαλαρώνουν και το σύστημα μεταβαίνει σε πιο ρεαλιστικά σενάρια, το Kleio εξακολουθεί να υπερτερεί σαφώς του History scheduler, αν και η απόδοσή του απέχει πλέον σημαντικά από εκείνη του Oracle.

Τέλος, η εφαρμογή του δρομολογητή σε επιλεγμένα μετροπρογράμματα από τις συλλογές PARSEC και SPEC απέδειξε ότι στην πράξη το Kleio επιτυγχάνει κατά μέσο όρο 15.95% βελτίωση στην απόδοση σε σύγκριση με τον κλασικό History scheduler. Παρ' όλα αυτά, σε πολλές περιπτώσεις το περιθώριο περαιτέρω βελτίωσης παραμένει μεγάλο, καθώς το χάσμα μεταξύ Kleio και Oracle scheduler εξακολουθεί να είναι έντονα αισθητό. Το γεγονός αυτό έρχεται σε αντίθεση με τις υψηλές προσδοκίες που δημιουργήθηκαν από την αξιολόγηση του Kleio σε περιβάλλοντα προσομοίωσης, όπου είχε παρατηρηθεί κατά μέσο όρο γεφύρωση του 80% του χάσματος απόδοσης. Συνεπώς, καθίσταται σαφές ότι η μετάβαση από τη θεωρητική προσομοίωση σε ένα πραγματικό σύστημα συνοδεύεται από σημαντικές προκλήσεις.

5.2 Μελλοντικές προεκτάσεις

Το πρόβλημα της μεταβλητότητας που εισάγει η δειγματοληψία με PEBS θα μπορούσε ενδεχομένως να αντιμετωπιστεί είτε με επιλογή ενός διαφορετικού τύπου νευρωνικού δικτύου, ικανό να προσαρμόζεται ευκολότερα σε ακολουθίες εισόδου που εμφανίζουν σημαντική μεταβλητότητα είτε με μια εξαιρετικά λεπτομερή δειγματοληψία, που ουσιαστικά ισοδυναμεί με την πλήρη καταγραφή όλων των προσβάσεων μνήμης και δεν εμφανίζει διακυμάνσεις, αρκεί βεβαίως να μην προστίθεται σημαντική επιβάρυνση.

Η συγκεκριμένη υλοποίηση του δρομολογητή, αν και θέτει τα θεμέλια για μια χρήση κάτω από πραγματικές συνθήκες, πάσχει από ένα βασικό πρόβλημα: την ανάγκη για παύση και επανεκκίνηση των εφαρμογών κατά τη δρομολόγηση. Η προσέγγιση αυτή υιοθετήθηκε επειδή η διαδικασία των προβλέψεων με χρήση μοντέλων (inference) απαιτεί αρκετό χρόνο (έως και 10sec για 200 σελίδες ανά περίοδο δρομολόγησης), γεγονός που εμποδίζει τη σωστή αξιολόγηση της απόδοσης. Όμως το πρόβλημα αυτό οφείλεται κατά βάση στο γεγονός ότι οι προβλέψεις γίνονται σειριακά. Συνεπώς, μπορεί εύκολα να αντιμετωπιστεί τόσο με την αύξηση της παραλληλίας όσο και με την αξιοποίηση ειδικών επιταχυντών σε επίπεδο υλικού όπως GPUs, TPUs.

Επιπλέον, μια πιθανή επέκταση της παρούσας εργασίας αφορά τη συνεκτίμηση των εντολών εγγραφής κατά τη λήψη αποφάσεων δρομολόγησης. Στην τρέχουσα προσέγγιση κάτι τέτοιο παραλείφθηκε, κυρίως λόγω της έλλειψης εξειδικευμένων μετρητών σε επίπεδο υλικού που να μπορούν να καταγράφουν πληροφορίες για τον προορισμό κάθε εντολής εγγραφής

(DRAM ή στη NVM). Τέλος, εφόσον αντιμετωπιστεί το ζήτημα της μεταβλητότητας, θα μπορούσε να εξεταστεί η λειτουργία του δρομολογητή σε εφαρμογές όπου η χρήση μνήμης μεταβάλλεται δυναμικά ανάλογα με την είσοδο. Σε αυτές τις περιπτώσεις, η εκπαίδευση των μοντέλων με πολλαπλούς συνδυασμούς εισόδου θα μπορούσε πιθανώς να αποτελέσει μια αποτελεσματική προσέγγιση.

Βιβλιογραφία

[1] Jalil Boukhobza, Stéphane Rubini, Renhai Chen και Zili Shao. *Emerging NVM: A Survey on Architectural Integration and Research Challenges*. *ACM Trans. Des. Autom. Electron. Syst.*, 23(2), 2017.

[2] Thaleia Dimitra Doudali, Sergey Blagodurov, Abhinav Vishnu, Sudhanva Gurumurthi και Ada Gavrilovska. *Kleio: A Hybrid Memory Page Scheduler with Machine Intelligence*. *Proceedings of the 28th International Symposium on High-Performance Parallel and Distributed Computing*, HPDC '19, σελίδα 37–48, New York, NY, USA, 2019. Association for Computing Machinery.

[3] Luiz E. Ramos, Eugene Gorbatov και Ricardo Bianchini. *Page placement in hybrid memory systems*. *Proceedings of the International Conference on Supercomputing*, ICS '11, σελίδα 85–95, New York, NY, USA, 2011. Association for Computing Machinery.

[4] Midhul Vuppalapati και Rachit Agarwal. *Tiered Memory Management: Access Latency is the Key!* *Proceedings of the ACM SIGOPS 30th Symposium on Operating Systems Principles*, SOSP '24, σελίδα 79–94, New York, NY, USA, 2024. Association for Computing Machinery.

[5] J. A. Mandelman, R. H. Dennard, G. B. Bronner, J. K. DeBrosse, R. Divakaruni, Y. Li και C. J. Radens. *Challenges and future directions for the scaling of dynamic random-access memory (DRAM)*. *IBM Journal of Research and Development*, 46(2.3):187–212, 2002.

[6] Onur Mutlu. *The RowHammer problem and other issues we may face as memory becomes denser*. *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*, σελίδες 1116–1121, 2017.

[7] Onur Mutlu. *Memory scaling: A systems architecture perspective*. *2013 5th IEEE International Memory Workshop, IMW 2013*, σελίδες 21–25, 2013.

[8] Saeed Kargar και Faisal Nawab. *Challenges and future directions for energy, latency, and lifetime improvements in NVMs*. *Distributed and Parallel Databases*, 41(3):163–189, 2023.

[9] Bruce Jacob, Spencer W. Ng και David T. Wang. *CHAPTER 1 - An Overview of Cache Principles*. *Memory Systems*Bruce Jacob, Spencer W. Ng και David T. Wang, επιμελητές, σελίδες 57–77. Morgan Kaufmann, San Francisco, 2008.

[10] Mitesh R. Meswani, Sergey Blagodurov, David Roberts, John Slice, Mike Ignatowski και Gabriel H. Loh. *Heterogeneous memory architectures: A HW/SW approach for mixing die-stacked and off-package memories*. *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*, σελίδες 126–136, 2015.

[11] Amanda Raybuck, Tim Stamler, Wei Zhang, Mattan Erez και Simon Peter. *HeMem: Scalable Tiered Memory Management for Big Data Applications and Real NVM*. *Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles*, SOSP '21, σελίδα 392–407, New York, NY, USA, 2021. Association for Computing Machinery.

[12] Neha Agarwal και Thomas F. Wenisch. *Thermostat: Application-transparent Page Management for Two-tiered Main Memory*. *Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '17, σελίδα 631–644, New York, NY, USA, 2017. Association for Computing Machinery.

[13] Vishal Gupta, Min Lee και Karsten Schwan. *HeteroVisor: Exploiting Resource Heterogeneity to Enhance the Elasticity of Cloud Platforms*. *Proceedings of the 11th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, VEE '15, σελίδα 79–92, New York, NY, USA, 2015. Association for Computing Machinery.

[14] Taekyung Heo, Yang Wang, Wei Cui, Jaehyuk Huh και Lintao Zhang. *Adaptive Page Migration Policy With Huge Pages in Tiered Memory Systems*. *IEEE Transactions on Computers*, 71(1):53–68, 2022.

[15] Zi Yan, Daniel Lustig, David Nellans και Abhishek Bhattacharjee. *Nimble Page Management for Tiered Memory Systems*. *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '19, σελίδα 331–345, New York, NY, USA, 2019. Association for Computing Machinery.

[16] Hasan Al Maruf, Hao Wang, Abhishek Dhanotia, Johannes Weiner, Niket Agarwal, Pallab Bhattacharya, Chris Petersen, Mosharaf Chowdhury, Shobhit Kanaujia και Prakash Chauhan. *TPP: Transparent Page Placement for CXL-Enabled Tiered-Memory*. New York, NY, USA, 2023. Association for Computing Machinery.

[17] Jifei Yi, Benchao Dong, Mingkai Dong και Haibo Chen. *On the precision of precise event based sampling*. *Proceedings of the 11th ACM SIGOPS Asia-Pacific Workshop on Systems*, APSys '20, σελίδα 98–105, New York, NY, USA, 2020. Association for Computing Machinery.

[18] Dr Robert N. M. Watson. *Advanced Operating System: Hardware Performance Counters (HWPMC)*. https://www.cl.cam.ac.uk/teaching/2021/L41/2020-2021-l41-hwpmc.pdf, 2021.

[19] Soramichi Akiyama και Takahiro Hirofuchi. *Quantitative Evaluation of Intel PEBS Overhead for Online System-Noise Analysis*. *Proceedings of the 7th International W-*

orkshop on Runtime and Operating Systems for Supercomputers ROSS 2017, ROSS '17, New York, NY, USA, 2017. Association for Computing Machinery.

[20] The Linux Kernel Documentation. *Perf Ring Buffer*. https://docs.kernel.org/userspace-api/perf_ring_buffer.html, 2025.

[21] Intel Corporation. *What Is Processor Throttling?*, 2023. Αχχεσσεδ: 2025-03-28.

[22] The Linux Kernel Documentation. *Idle Page Tracking*. https://docs.kernel.org/admin-guide/mm/idle_page_tracking.html, 2025.

[23] Andrea Arcangeli. *Autonuma Benchmark*. https://mirrors.edge.kernel.org/pub/linux/kernel/people/andrea/autonuma/autonuma_bench-20120530.pdf, 2012.

[24] Mel Gorman. *Understanding the Linux Virtual Memory Manager, Chapter 13: Page Frame Reclamation*. https://www.kernel.org/doc/gorman/html/understand/understand013.html, 2004. Αχχεσσεδ: Θανυαρψ 12, 2025.

[25] L. C. Jain και L. R. Medsker. *Recurrent Neural Networks: Design and Applications*. CRC Press, Inc., USA, 1στη έκδοση, 1999.

[26] P.J. Werbos. *Backpropagation through time: what it does and how to do it*. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[27] Aston Zhang, Zachary C. Lipton, Mu Li και Alexander J. Smola. *Backpropagation Through Time*. https://d2l.ai/chapter_recurrent-neural-networks/bptt.html, 2020.

[28] Sepp Hochreiter και Jürgen Schmidhuber. *Long Short-term Memory*. *Neural computation*, 9:1735–80, 1997.

[29] Persistent Memory Development Kit. *NDCTL User Guide: Concepts*. https://docs.pmem.io/ndctl-user-guide/concepts, 2025.

[30] Linux Performance Team. *perf: Linux profiling with performance counters*. https://perf.wiki.kernel.org/, ν.δ. Αχχεσσεδ: 2025-03-28.

[31] Hector Marco-Gisbert και Ismael Ripoll Ripoll. *Address Space Layout Randomization Next Generation*. *Applied Sciences*, 9(14), 2019.

[32] The Linux Kernel Documentation. *Perf Event Paranoid*. https://www.kernel.org/doc/Documentation/sysctl/kernel.txt, 2025.

[33] François Chollet και others. *Keras*. https://keras.io, 2015.

[34] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay

Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu και Xiaoqiang Zheng. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*, 2015. Σοφτωαρε ααιλαβλε φρομ τενσορφλοω.οργ.

[35] Thaleia Dimitra Doudali και Ada Gavrilovska. *Coeus: Clustering (A)like Patterns for Practical Machine Intelligent Hybrid Memory Management*. *2022 22nd IEEE International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, σελίδες 615–624, 2022.

[36] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta και Masanori Koyama. *Optuna: A Next-Generation Hyperparameter Optimization Framework*. *The 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, σελίδες 2623–2631, 2019.

[37] Brian S. Everitt. *The Cambridge Dictionary of Statistics*. Cambridge University Press, 2νδη έκδοση, 2002.