



NATIONAL TECHNICAL UNIVERSITY OF
ATHENS

SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING
DIVISION OF COMPUTER SCIENCE

Interference-aware Classification and Deployment of
Database Workloads on Heterogeneous Memory
Systems

DIPLOMA THESIS

of

Niki Doulou

Supervisor: Dimitrios Soudris
Professor NTUA

MICROPROCESSORS LABORATORY AND DIGITAL SYSTEMS LAB
Athens, March 2025



National Technical University of Athens
School of Electrical and Computer Engineering
Division of Computer Science
Microprocessors Laboratory and Digital Systems Lab

Interference-aware Classification and Deployment of Database Workloads on Heterogeneous Memory Systems

DIPLOMA THESIS

of

Niki Doulou

Supervisor: Dimitrios Soudris
Professor NTUA

Approved by the examination committee on 14th March, 2025.

.....
Dimitrios Soudris
Professor NTUA

.....
Sotirios Xydis
Assistant Professor NTUA

.....
Georgios Lentaris
Assistant Professor UNIWA

Athens, March 2025

.....
NIKI DOULOU
Graduate of Electrical and
Computer Engineering NTUA

Copyright © – All rights reserved Niki Doulou, 2025.

The copying, storage and distribution of this diploma thesis, all or part of it, is prohibited for commercial purposes. Reprinting, storage and distribution for non-profit, educational or of a research nature is allowed, provided that the source is indicated and that this message is retained.

The content of this thesis does not necessarily reflect the views of the Department, the Supervisor, or the committee that approved it.

Πνευματική ιδιοκτησία ©– Με επιφύλαξη παντός δικαιώματος Νίκη Δούλου, 2025.

Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ' ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσεως υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τον συγγραφέα και δεν πρέπει να ερμηνευθεί ότι αντιπροσωπεύουν τις επίσημες θέσεις του Εθνικού Μετσόβιου Πολυτεχνείου.

Περίληψη

Τα τελευταία χρόνια η ραγδαία εξέλιξη των εφαρμογών έχει οδηγήσει σε αντίστοιχη αύξηση των υπολογιστικών απαιτήσεων και του όγκου των δεδομένων. Παράλληλα, οι κλασσικές τεχνολογίες μνήμης, οι οποίες βασίζονται σε DRAM τεχνολογίες, έχουν φτάσει τα φυσικά τους όρια σε θέματα κλιμακωσιμότητας και βιωσιμότητας. Έτσι, εισάγεται μια νέα αρχιτεκτονική τεχνολογίας μνημών, η οποία βασίζεται στα ετερογενή συστήματα μνήμης ενσωματώνοντας την χρήση DRAM και μη πτητικών μνημών. Ωστόσο, ταυτόχρονα με την χρήση τους προκύπτουν προκλήσεις για την τοποθέτηση των εφαρμογών σε αυτές και για τον τρόπο διαχείρισής τους. Σε αυτήν την εργασία, πραγματοποιείται εκτενής πειραματική ανάλυση και χαρακτηρισμός εφαρμογών βάσεων δεδομένων. Η πειραματική ανάλυση πραγματοποιείται τόσο σε απομονωμένο υπολογιστικό περιβάλλον, όσο και σε μια πιο ρεαλιστική προσομοίωση συστήματος με την χρήση ειδικών προγραμμάτων που αξιοποιούν στο μέγιστο τους πόρους του συστήματος. Επιπλέον, υλοποιείται η εκπαίδευση μοντέλων μηχανικής μάθησης βασιζόμενη στα δεδομένα που συλλέγονται κατά την πειραματική ανάλυση και οι εφαρμογές ταξινομούνται σε κλάσεις ανάλογα με το αν η λειτουργία τους οριοθετείται από την μνήμη, από το εύρος ζώνης της μνήμης ή από τον επεξεργαστή. Τα μοντέλα επιτυγχάνουν ακρίβεια προβλέψεων που ξεπερνά το 90% . Έτσι, διεξάγονται τελικά συμπεράσματα για την τοποθέτηση των εφαρμογών στο ετερογενές σύστημα.

Λέξεις Κλειδιά — Ετερογενή Συστήματα Μνημών, DRAM, NVM, Βάσεις δεδομένων, Χαρακτηρισμός Εφαρμογών, Μοντέλα Μηχανικής Μάθησης, Ταξινόμηση, Τοποθέτηση

Abstract

Over the recent years, the rapid growth of applications has led to a corresponding increase in the computational requirements and the volume of data. At the same time, traditional memory technologies based on DRAM technologies have reached their physical limits in terms of scalability and sustainability. As a result, a novel memory technology architecture has been introduced, based on heterogeneous memory systems, that integrate DRAM and NVM technologies. However, some several challenges arise regarding application placement and their management. In this work, it is conducted an extensive profiling analysis and characterization of database applications. The experimental analysis is performed both in an isolated computing system environment and in a interference-aware system using specific benchmarks that stress the resources of the system. Additionally, machine learning models are trained based on the data collected during the profiling analysis, and applications are classified into memory-bound, cpu-bound or bandwidth-bound depending on their performance. The models achieve prediction accuracy over 90%. Finally, conclusions can be drawn regarding the placement of applications in the heterogeneous system.

Keywords — Heterogeneous Memory Systems, DRAM, NVM, Databases, Profiling Analysis, Machine Learning, Classification, Placement

Ευχαριστίες

Πρώτα από όλα, θα ήθελα να εκφράσω τις ειλικρινείς μου ευχαριστίες στον επιβλέποντα καθηγητή μου, Σούντρη Δημήτριο, και στον Διδάκτορα Κατσαραγάκη Μανώλη για την άριστη συνεργασία μας, την συνεχή βοήθεια και υποστήριξη που μου παρείχαν σε ό,τι χρειαζόμουν κατά την εκπόνηση της διπλωματικής μου εργασίας. Επιπλέον, θα ήθελα να ευχαριστήσω τους γονείς μου, οι οποίοι με στηρίζουν και πιστεύουν σε μένα, αλλά και τις αδερφές μου, οι οποίες αποτελούν συνεχώς πηγή έμπνευσης για μένα. Τέλος, δεν θα μπορούσα να μην ευχαριστήσω τις φίλες μου, τον άνθρωπο που με στηρίζει σε κάθε μου βήμα, αλλά και τους φίλους και συμφοιτητές μου που μου στάθηκαν σε όλα τα χρόνια των σπουδών μου και παράλληλα μου χάρισαν πολύ όμορφες αναμνήσεις.

Νίκη Δούλου
Μάρτιος 2025

Contents

Περίληψη	5
Abstract	7
Ευχαριστίες	9
Contents	11
Figure List	13
Table List	15
Εκτεταμένη Ελληνική Περίληψη	17
1 Ελληνική Περίληψη	17
1.1 Εισαγωγή	17
1.2 Σχετική Βιβλιογραφία	18
1.3 Θεωρητικό Υπόβαθρο	19
1.3.1 Ετερογενή συστήματα μνήμης	19
1.3.2 Σχεσιακές βάσεις δεδομένων	21
1.3.3 Αλγόριθμοι Επιβλεπόμενης Μηχανικής Μάθησης	22
1.3.4 Random Forest	22
1.3.5 Support Vector Machine	23
1.3.6 Multi-Layer Perceptron	23
1.3.7 Logistic Regression	23
1.4 Σκιαγράφηση Εφαρμογών	23
1.4.1 Διεξαγωγή Πειραμάτων σε Απομονωμένο Σύστημα	24
1.4.2 Διεξαγωγή Πειραμάτων σε Interference-aware Σύστημα	25
1.5 Προτεινόμενη Μεθοδολογία	26
1.6 Αξιολόγηση	27
1.7 Συμπεράσματα και μελλοντικές επεκτάσεις	29
1.7.1 Συμπεράσματα	29
1.7.2 Μελλοντικές επεκτάσεις	29
2 Introduction	31
3 Related Work	33
4 Background	35
4.1 Heterogeneous memory systems	35
4.2 Relational Databases	37
4.3 Supervised Machine Learning Algorithms	37
4.3.1 Random Forest	38

4.3.2	Support Vector Machine	39
4.3.3	Multi-Layer Perceptron	39
4.3.4	Logistic Regression	40
5	Profiling Analysis and Characterization	41
5.1	Experiments in Isolated System	41
5.1.1	Bank	44
5.1.2	Kv	48
5.1.3	TPCC	52
5.1.4	YCSB	56
5.2	Experiments in Interference-aware System	67
5.2.1	iBench Benchmarks	67
5.2.2	Results	67
5.2.3	Bank	71
5.2.4	Kv	72
5.2.5	TPCC	72
5.2.6	YCSB	72
6	Proposed Methodology	77
7	Experimental Setup and Evaluation	81
8	Conclusion and Future Work	85
8.1	Conclusion	85
8.2	Future Work	85

Figure List

1.3.1 Διαφορετικές Λειτουργίες Μνήμης.[1]	20
4.1.1 Operating modes of Optane DCPMM.[1]	36
4.3.1 Supervised machine learning method.[2]	38
4.3.2 Random Forest Algorithm.[?]	39
4.3.3 Random Forest Algorithm.[3]	39
4.3.4 Mlp algorithm for one hidden layer.[4]	40
5.1.1 Persistent memory accesses	42
5.1.2 P99 in DRAM execution	43
5.1.3 P99 in PMEM execution	44
5.1.4 Memory DRAM Reads/Writes plots for different concurrency values - Bank Workload	45
5.1.5 Memory Optane plots for different concurrency values - Bank Workload	47
5.1.6 DRAM throughput plots for different concurrency values - kv Workload	49
5.1.7 PMM throughput plots for different concurrency values - kv Workload	51
5.1.8 DRAM throughput plot for different concurrency values - tpcc Workload	53
5.1.9 PMM throughput plot for different concurrency values - tpcc Workload	55
5.1.10 DRAM throughput plots for different concurrency values - ycsb A Workload	57
5.1.11 PMM throughput plots for different concurrency values - ycsb A Workload	58
5.1.12 DRAM throughput plots for different concurrency values - ycsb B Workload	59
5.1.13 PMM throughput plots for different concurrency values - ycsb B Workload	60
5.1.14 Memory DRAM plots for different concurrency values - ycsb C Workload	61
5.1.15 PMM Throughput plots for different concurrency values - ycsb C Workload	62
5.1.16 DRAM throughput plots for different concurrency values - ycsb F Workload	63
5.1.17 PMM Throughput plots for different concurrency values - ycsb F Workload	64
5.2.1 Memory Optane accesses plots for different concurrency values and benchmarks	68
5.2.2 Query latency's for different benchmarks in DRAM execution	69
5.2.3 Query latency's for different benchmarks in Optane execution	70
5.2.4 Ops/sec with benches for Bank	71
5.2.5 Ops/sec with benches for kv	72
5.2.6 Ops/sec with benches for tpcc	72
5.2.7 Ops/sec with benches for ycsb A	73
5.2.8 Ops/sec with benches for ycsb B	73
5.2.9 Ops/sec with benches for ycsb C	74
5.2.10 Ops/sec with benches for Ycsb F	74
6.0.1 Distribution of Input Samples into the Classes	78
6.0.2 Proposed Methodology	80
7.0.1 Heatmaps of Mean Fit time and Accuracy for Random Forest	81
7.0.2 Heatmaps of Mean Fit time and Accuracy for SVM	82
7.0.3 Heatmaps of Mean Fit time and Accuracy for MLP	82

7.0.4 Heatmaps of Mean Fit time and Accuracy for Logistic Regression	83
7.0.5 Accuracy of Workloads Predictions for Different Models	84

Table List

1.1	Τιμές Μέγιστου Bandwidth	27
5.1	YCSB Memory DRAM Throughput-DRAM Execution	65
5.2	YCSB Memory PMM Throughput-DRAM Execution	65
5.3	YCSB Memory DRAM Throughput-Optane Execution	65
5.4	YCSB Memory PMM Throughput-Optane Execution	65
6.1	Max Bandwidth Values	78
6.2	User-Level Thresholds for Classification	78
7.1	ML Model's Best Parameters	83
7.2	Accuracy,Precision and Recall for each model	83
7.3	Accuracy of Workloads predictions for the different models	84

Chapter 1

Ελληνική Περίληψη

1.1 Εισαγωγή

Τα τελευταία χρόνια, η ραγδαία ανάπτυξη εφαρμογών δεδομένων έχει οδηγήσει σε αύξηση των απαιτήσεων μνήμης, του χώρου αποθήκευσης, αλλά και των αναγκών για διεργασίες υψηλού ρυθμού. Αρκετοί τομείς εφαρμογών που βασίζονται σε βάσεις δεδομένων, σε έξυπνα δίκτυα IoT, σε υπηρεσίες cloud ή σε εφαρμογές μηχανικής μάθησης απαιτούν την επεξεργασία δεδομένων εντός και επιτόπου πάνω στις πολύπλοκες ιεραρχίες μνήμης για την αποφυγή δαπανηρών μεταφορών δεδομένων[5]. Έτσι, ασκείται μεγάλη πίεση στο σύστημα κύριας μνήμης.

Ταυτόχρονα, οι τεχνολογίες DRAM έχουν φτάσει στα φυσικά όριά τους όσον αφορά την επεκτασιμότητα και τη βιωσιμότητα [6]. Η αύξηση των DRAM Dimms σε ένα σύστημα μνήμης για την επίτευξη μεγαλύτερης απόδοσης και μεγαλύτερης χωρητικότητας αποδεικνύεται αναποτελεσματική, λόγω της αυξημένης κατανάλωσης ενέργειας που αποδίδεται στη διαρροή και την ισχύ ανανέωσης της τεχνολογίας DRAM[5]. Η μνήμη ενός συστήματος αποδεικνύεται ότι καταναλώνει σχεδόν το 50% της συνολικής καταλισκώμενης ενέργειας ενός τυπικού συστήματος HPC [7]. Στο πλαίσιο αυτό, οι νέες τεχνολογίες των Μη Πτητικών Μνημών έχουν κεντρίσει το ενδιαφέρον των ερευνητών. Τεχνολογίες όπως το 3D-XPoint [8], η Resistive RAM (ReRAM) [9] και η Spin Transfer Torque Magnetic RAM (STTMRAM) [10] αποτελούν παραδείγματα Μη πτητικών μνημών (Non-Volatile Memories). Αυτές οι τεχνολογίες συνήθως προσφέρουν ανθεκτικότητα δεδομένων, μειωμένη κατανάλωση ενέργειας και μεγαλύτερη πυκνότητα από τη DRAM, με αποτέλεσμα να προσφέρουν αυξημένες χωρητικότητες μνήμης σε χαμηλότερο κόστος [11]. Σε αυτή την εργασία, διερευνούμε τη λειτουργία της Intel Optane DPCM, η οποία είναι η μοναδική εμπορική Non Volatile μνήμη που υπάρχει μέχρι τώρα και βασίζεται στην τεχνολογία 3D-XPoint.

Ωστόσο, οι τεχνολογίες μη πτητικών μνημών παρουσιάζουν αρκετά μειονεκτήματα στο κύριο σύστημα μνήμης. Εισάγουν υψηλότερες τιμές καθυστερήσεων, χαμηλότερο εύρος ζώνης και περιορισμένη αντοχή εγγραφών. Ταυτόχρονα, παρουσιάζουν ασύμμετρη καθυστέρηση και κατανάλωση ενέργειας μεταξύ των λειτουργιών ανάγνωσης και εγγραφής. Η απευθείας αντικατάσταση της DRAM, λοιπόν, με μια NVM μειώνει την απόδοση. Έτσι, οι διαστάσεις των NVMs συνήθως συνδυάζονται με μονάδες DRAM και συνιστούν τελικά ένα ετερογενές σύστημα μνήμης DRAM/NVM. Αυτά τα συστήματα στοχεύουν να επεκτείνουν τη χωρητικότητα της μνήμης βιώσιμα και οικονομικά, προσφέροντας στους χρήστες ένα ευρύ φάσμα τεχνολογιών μνήμης.

Η συνεισφορά αυτής της εργασίας βασίζεται στην εκτέλεση, στην πειραματική ανάλυση και στον χαρακτηρισμό εφαρμογών βάσεων δεδομένων, οι οποίες εκτελούνται σε ένα ετερογενές σύστημα μνήμης DRAM/NVM και στην ταξινόμησή τους σε κλάσεις ανάλογα με την απόδοσή τους, προκειμένου να δωθούν κατευθύνσεις για την τοποθέτησή τους στις μνήμες του ετερογενούς συστήματος μνήμης. Πιο συγκεκριμένα, εστιάζουμε στην ανάλυση και στον χαρακτηρισμό εφαρμογών βάσεων δεδομένων, καθώς

αυτές αποτελούν εφαρμογές με μεγάλες απαιτήσεις μνήμης και ανάγκες χειρισμού μεγάλων όγκων δεδομένων. Ως εκ τούτου, είναι ενδιαφέρον να εξεταστεί η συμπεριφορά και η απόδοσή τους χρησιμοποιώντας πολλαπλούς πυρήνες σε ένα ετερογενές σύστημα μνήμης. Τα αιτήματα κάθε εφαρμογής στο σύστημα εκτελούνται με την χρήση DRAM τεχνολογίας και με την χρήση της Optane. Στην συνέχεια συγκρίνονται οι επιδόσεις τους. Επιπλέον, μια καινοτόμος προσέγγιση αυτής της εργασίας είναι ότι όλες οι εφαρμογές δεν εκτελούνται μόνο σε ένα απομονωμένο περιβάλλον συστήματος, αλλά και σε ένα σύστημα με επίγνωση των πιθανών παρεμβολών, όπου πολλαπλές εφαρμογές εκτελούνται ταυτόχρονα και οι πόροι του συστήματος ενδέχεται να είναι κατειλημμένοι. Ως τελικό βήμα, εκπαιδεύουμε διαφορετικά μοντέλα μηχανικής μάθησης για να προβλέψουμε εάν οι εφαρμογές οριοθετούνται από την μνήμη, από το εύρος ζώνης ή από τον επεξεργαστή, με βάση τα δεδομένα εισόδου που αντλούμε από την πειραματική ανάλυση. Η απόδοση και η ακρίβεια αυτών των μοντέλων συγκρίνονται και τελικά μπορούν να καθοριστούν κατευθύνσεις για την τοποθέτηση των εφαρμογών στο ετερογενές σύστημα.

Το υπόλοιπο αυτής της εργασίας οργανώνεται ως εξής. Το Κεφάλαιο 3 παρουσιάζει μια επισκόπηση της προηγούμενης σχετικής έρευνας και το Κεφάλαιο 4 επεξηγεί το βασικό θεωρητικό υπόβαθρο. Στο Κεφάλαιο 5 περιγράφονται αναλυτικά η πειραματική ανάλυση και ο χαρακτηρισμός των εφαρμογών. Στο Κεφάλαιο 6 παρουσιάζεται η προτεινόμενη μεθοδολογία για την ταξινόμηση των εφαρμογών και στο Κεφάλαιο 7 παρουσιάζεται η πειραματική αξιολόγηση των μοντέλων. Τέλος, το Κεφάλαιο 8 συνοψίζει αυτή την εργασία και προτείνει πιθανές μελλοντικές προεκτάσεις της.

1.2 Σχετική Βιβλιογραφία

Σε ένα ετερογενές σύστημα μνήμης, η απόφαση για την τοποθέτηση δεδομένων ή αντικειμένων σε κάθε υποσύστημα μνήμης, είτε στη DRAM είτε στην PMEM, δεν είναι τετριμμένη και έχει σημαντικές επιπτώσεις στην απόδοση. Κατά την διάρκεια των χρόνων, έχουν προταθεί τόσο υλοποιήσεις τοποθέτησης βασισμένες στο υλικό (hardware-based) όσο και στο λογισμικό (software-based). Οι μηχανισμοί που βασίζονται στο υλικό είναι συνήθως αποδοτικοί, αλλά στερούνται ευελιξίας και μερικές φορές οδηγούν σε ασυνεπή απόδοση. Οι προσεγγίσεις που βασίζονται στο λογισμικό επιφέρουν διαχειριστικά κόστη, αλλά απαιτούν επίσης γνώση και επεμβατικές αλλαγές στον κώδικα. Ορισμένες από τις μεθόδους που έχουν προταθεί για την τοποθέτηση δεδομένων σε ένα ετερογενές σύστημα μνήμης παρουσιάζονται σε αυτή την ενότητα.

Οι Kannan et al. εισάγουν τα KLOCs, τα οποία είναι λογικές ομαδοποιήσεις που καταγράφουν τα αντικείμενα πυρήνα που σχετίζονται με οντότητες του λειτουργικού συστήματος που ζητούνται από εφαρμογές. Πρόκειται για ένα αφαιρεμένο λειτουργικό σύστημα που επιτρέπει την ευέλικτη διαβάθμιση των αντικειμένων του πυρήνα και τη γρήγορη κατανομή τους, ενώ υποστηρίζει τη δυνατότητα μετεγκατάστασης. Τα KLOCs μετακινούν ομαδικά τα ψυχρά αντικείμενα του πυρήνα και τα knodes συσχετίζονται περαιτέρω με μία μεταβλητή που παρακολουθεί την προτεραιότητά τους (ηλικία) και την ενεργή κατάστασή τους (χρήση). Μία άλλη μέθοδος ταξινόμησης και κατανομής αντικειμένων στα ετερογενή συστήματα μνήμης είναι το MOCA, που προτείνεται από τον Narayan [12]. Περιλαμβάνει έναν profiler που ονομάζει, καταγράφει και ταξινομεί τα αντικείμενα μνήμης μιας εφαρμογής με βάση τη συμπεριφορά πρόσβασής τους, καθώς και ένα σύστημα κατανομής σε επίπεδο λειτουργικού συστήματος. Το MOCA εστιάζει κυρίως σε αντικείμενα σωρού (heap objects), καθώς αυτά συνήθως επηρεάζουν σημαντικά τη συνολική απόδοση και την ενεργειακή κατανάλωση μιας εφαρμογής με έντονη χρήση μνήμης. Η ταξινόμηση αποθηκεύεται ως μέρος του δυαδικού αρχείου της εφαρμογής, ενώ το λειτουργικό σύστημα είναι υπεύθυνο για την κατανομή σελίδων από ένα συγκεκριμένο διαμέρισμα εικονικής μνήμης σε φυσικές σελίδες μνήμης με τα επιθυμητά χαρακτηριστικά.

Οι Kammerdiener et al. [13] εισάγουν ένα αριθρωτό πλαίσιο λογισμικού σε επίπεδο χρήστη που συλλέγει και αναλύει πληροφορίες σε επίπεδο αντικειμένων για την καθοδήγηση της διαβάθμισης της μνήμης. Χρησιμοποιεί έναν προσαρμοσμένο διαχειριστή μνήμης, τον bkmalloc, καθώς και ένα εργαλείο προφίλ και διαχείρισης που ονομάζεται Memory Auto Tiering (MAT) Daemon. Ο bkmalloc εκχωρεί νέα αντικείμενα σε περιοχές εικονικής μνήμης ευθυγραμμισμένες σελίδα-προς-σελίδα, ενώ ο MAT Daemon κατα-

γράφει τη χρήση μνήμης κάθε γνωστού αντικειμένου δεδομένων και χρησιμοποιεί αυτές τις πληροφορίες για να δημιουργήσει και να επιβάλλει συστάσεις τοποθέτησης μνήμης σε όλη την πλατφόρμα. Άλλες μελέτες προτείνουν ένα σύστημα κατανομής αντικειμένων που εκτελεί σε πραγματικό χρόνο εισαγωγή και απομάκρυνση αντικειμένων στη μνήμη υψηλού εύρους ζώνης (HBM), βασιζόμενο σε χαρακτηριστικά τόσο λογισμικού όσο και υλικού, τα οποία ενσωματώνονται σε ένα μοντέλο κόστους. Ειδικότερα, στο [14] το μοντέλο κόστους εκτιμά το όφελος της τοποθέτησης ενός αντικειμένου στη μία μνήμη έναντι της άλλης, και στη συνέχεια εφαρμόζεται ο αλγόριθμος του Σακιδίου (Knapsack). Ως έξοδος, ο αλγόριθμος παρέχει μια λίστα με τα αντικείμενα που είναι οι καλύτεροι υποψήφιοι για αποθήκευση στην HBM για μια συγκεκριμένη φάση της εφαρμογής. Η προτεινόμενη μέθοδος στο [15] βασίζεται στην ιδέα ενός Σύμβουλου Ετερογενούς Μνήμης (Heterogeneous Memory Advisor), ο οποίος στοχεύει στη μείωση των καθυστερήσεων του επεξεργαστή λόγω προσβάσεων στη μνήμη, τοποθετώντας τα πιο προσπελάσιμα αντικείμενα στο υποσύστημα μνήμης με το υψηλότερο εύρος ζώνης. Ο βασικός αλγόριθμος του Advisor βασίζεται σε μία απλοποιητική χαλάρωση του προβλήματος πολλαπλού σακιδίου 0/1 (0/1 multiple knapsack problem). Η ροή εργασίας ξεκινά με την διαδικασία καταγραφής του προφίλ ενός εκτελέσιμου προγράμματος εκτός σύνδεσης, δημιουργώντας ένα αρχείο ιχνών και εξάγοντας μετρικές απόδοσης για τα αντικείμενα δεδομένων που δημιουργούνται κατά την εκτέλεση της εφαρμογής.

Άλλες μελέτες, όπως αυτές των [16] [5], επικεντρώνονται στην τοποθέτηση σελίδων σε ετερογενή συστήματα μνήμης DRAM/NVM. Συγκεκριμένα, στο [16] προτείνεται ένας προγραμματιστής λήψης αποφάσεων κατά την εκτέλεση, ο οποίος τοποθετεί δυναμικά τις σελίδες ώστε να βελτιστοποιήσει την απόδοση και την κατανάλωση ενέργειας. Στόχος είναι η μεγιστοποίηση των αιτημάτων μνήμης της εφαρμογής που εξυπηρετούνται από τη DRAM, ενώ τα δεδομένα που δεν προσπελάζονται συχνά μεταφέρονται στη μνήμη NVM για εξοικονόμηση ενέργειας. Τέλος, η εργασία [5] ακολουθεί μία προσέγγιση τριών βημάτων: βελτιστοποίηση της οργάνωσης δεδομένων, ανάλυση αντικειμένων μνήμης και τελική τοποθέτηση στη μνήμη. Το αποτέλεσμα της μεθοδολογίας είναι ένα σύνολο βέλτιστων λύσεων Pareto, όπου κάθε σημείο της καμπύλης Pareto αντιστοιχεί σε μία συγκεκριμένη εκδοχή της εφαρμογής, τοποθετημένη σε ένα συγκεκριμένο σύστημα DRAM/NVM.

Στην παρούσα εργασία, προτείνουμε μια νέα προσέγγιση λογισμικού για την αποτελεσματική τοποθέτηση δεικτών και δεδομένων μεταξύ DRAM και Optane persistent memory. Εστιάζουμε στην ανάλυση προφίλ εφαρμογών που βασίζονται σε NewSQL σχεσιακή βάση δεδομένων, όπως η CockroachDB. Οι εφαρμογές αυτές επεξεργάζονται μεγάλους όγκους δεδομένων και απαιτούν υψηλή μνήμη και υπολογιστικούς πόρους. Επιπλέον, χαρακτηρίζουμε αυτές τις εφαρμογές σε ένα περιβάλλον που λαμβάνει υπόψη τις πιθανές παρεμβολές ενός συστήματος, εισάγοντας θόρυβο μέσω benchmarks που καταναλώνουν τους πόρους του συστήματος. Το τελευταίο βήμα είναι η πρόβλεψη της απόδοσης των εφαρμογών μέσω ενός μοντέλου μηχανικής μάθησης, χωρίς να απαιτούνται αλλαγές στον κώδικα των εφαρμογών ή του λειτουργικού συστήματος. Τα μοντέλα αυτά ταξινομούν τις εφαρμογές σε πέντε βασικές κατηγορίες, memory-bound, cpu-bound, DRAM Bandwidth bound, PMM Read or Write Bound. Με βάση αυτή την ταξινόμηση μπορούν να δοθούν κατευθυντήριες για την τοποθέτηση των εφαρμογών στο ετερογενές σύστημα.

1.3 Θεωρητικό Υπόβαθρο

1.3.1 Ετερογενή συστήματα μνήμης

Στον τομέα της επιστήμης των υπολογιστών, παραδοσιακά, τα συστήματα μνήμης αποτελούνται από κύριες μνήμες βασιζόμενες σε DRAM τεχνολογίες. Οι DRAM (Dynamic random-access memory) μνήμες αποθηκεύουν τα δεδομένα σε πίνακες, των οποίων κάθε στοιχείο αποτελεί ένα κύτταρο μνήμης και μπορεί να περιέχει μέχρι ένα bit δεδομένων. Κάθε κύτταρο μνήμης αποτελείται από ένα ζεύγος τρανζίστορ και πυκνωτή, ο οποίος περιοδικά ανανεώνεται προκειμένου να διατηρείται η πληροφορία σε αυτόν [17]. Σε πιο μοντέρνες τεχνολογίες, τα δεδομένα στην μνήμη οργανώνονται σε βαθμίδες που περιέχουν πολλαπλά banks, όπου κάθε bank είναι μια συστοιχία γραμμών και στηλών με έναν buffer γραμμών και μπορεί να προσπελαστεί παράλληλα μέσω διαύλων. Η δυναμικότητα των μνημών DRAM έγκειται στην περι-

οδική ανανέωση των πυκνωτών της για να διατηρεί την πληροφορία σε αυτούς. Ωστόσο, οι μνήμες αυτές δεν διατηρούν τα δεδομένα τους χωρίς την ύπαρξη τροφοδοσίας, δηλαδή αποτελούν πτητικές (volatile) τεχνολογίες μνημών.

Οι μνήμες τεχνολογίας DRAM αποτελούν γρήγορες μνήμες με αρκετή χωρητικότητα σε ένα σχετικά χαμηλό κόστος. Αν ανατρέξουμε στο παρελθόν, όσο οι απαιτήσεις των εφαρμογών μεγάλωναν τόσο κλιμακωνόταν η τεχνολογία των DRAM μνημών. Η μείωση των καθυστερήσεων πρόσβασης σε μια DRAM μνήμη, ωστόσο, δεν ακολουθούσε την συνεχώς αύξηση του μεγέθους της. Αυτό το γεγονός επιφέρει μεγάλες επιπτώσεις στον χρόνο εκτέλεσης των εφαρμογών, στην απόδοση και στην ενέργεια που καταναλώνεται κατά την χρήση της. Οι τεχνολογίες DRAM μνημών πλέον έχουν φτάσει στα όριά τους σε επίπεδο κλιμακωσιμότητας, αλλά και σε επίπεδο βιωσιμότητας- διάρκειας ζωής [5]. Παράλληλα όμως τα τελευταία χρόνια ο όγκος των δεδομένων των εφαρμογών αυξάνεται ραγδαία, ιδίως σε εφαρμογές βάσεων δεδομένων, σε εφαρμογές IoT που χρειάζονται επεξεργασία δεδομένων σε πραγματικό χρόνο, αλλά και σε αλγόριθμους μηχανικής μάθησης για την εκπαίδευση πολύπλοκων μοντέλων. Η ενσωμάτωση πολλαπλών DRAM DIMMs για την επίτευξη υψηλότερης απόδοσης και την κάλυψη αυξημένων απαιτήσεων μνήμης γίνεται ολοένα και πιο αναποτελεσματική, λόγω της αυξημένης κατανάλωσης ενέργειας που αποδίδεται στις απώλειες και την ισχύ ανανέωσης της τεχνολογίας DRAM. Επομένως, το κόστος συντήρησης των υπερ-σύγχρονων συστημάτων υπερυπολογιστών καθιστά εξαιρετικά μη ελκυστική την επένδυση στην αύξηση της χωρητικότητας μνήμης που βασίζεται αποκλειστικά στη DRAM. Έτσι, έχει αρχίσει η ανάπτυξη μνημών μη-πτητικών με διαφορετικές τεχνικές αποθήκευσης δεδομένων. Μια από αυτές είναι και η Intel Optane DPCM (Data Center Persistent Memory), η οποία μελετάται στην παρούσα διπλωματική εργασία [18].

Η Intel Optane DPCM αποτελεί μια νέα τεχνολογία μνήμης, η οποία είναι μη-πτητική και βασίζεται στην 3D XPoint αρχιτεκτονική. Κάθε κύτταρο μνήμης μπορεί να προσπελαστεί μεμονωμένα σε μια αρχιτεκτονική στην οποία στοιβάζονται πλέγματα μνημών σε ένα τρισδιάστατο πλέγμα για τη βελτίωση της πυκνότητας, την αύξηση της απόδοσης και την παροχή ανθεκτικότητας [19]. Σε σύγκριση με τις υπάρχουσες SSD μνήμες (συμπεριλαμβανομένων των σχετικών SSD Optane) που συνδέονται σε εξωτερική διεπαφή, όπως το PCIe, η Optane DC PMM προσφέρει καλύτερη απόδοση και χρησιμοποιεί διεπαφή μνήμης με δυνατότητα προσπέλασης ανά byte. Σε σύγκριση με τη DRAM, διαθέτει μεγαλύτερη πυκνότητα και ανθεκτικότητα. Συνολικά, ένας επεξεργαστής (CPU socket) μπορεί να υποστηρίξει έως και έξι Optane DC PMMs, επιτυγχάνοντας μέγιστη χωρητικότητα 6 TB μνήμης Optane DC. Η Intel Optane μπορεί να λειτουργεί σε δύο διαφορετικές λειτουργίες, σε λειτουργία Μνήμης (Memory Mode) κατά την οποία η Optane λειτουργεί ως μια επιπλέον μνήμη στην ιεραρχία μνημών, μεταξύ της DRAM και της κύριας μνήμης, και σε λειτουργία άμεσης πρόσβασης εφαρμογών (App-Direct Mode), όπου η Optane λειτουργεί σαν μια κρυφή μνήμη της κύριας μνήμης, χωρίς την ταυτόχρονη χρήση dram μνήμης. Οι λειτουργίες αυτές πρέπει να επιλέγονται με βάση τη συγκεκριμένη περίπτωση χρήσης και τις απαιτήσεις του φόρτου εργασίας. Οι λειτουργίες αυτές παρουσιάζονται στην παρακάτω εικόνα:

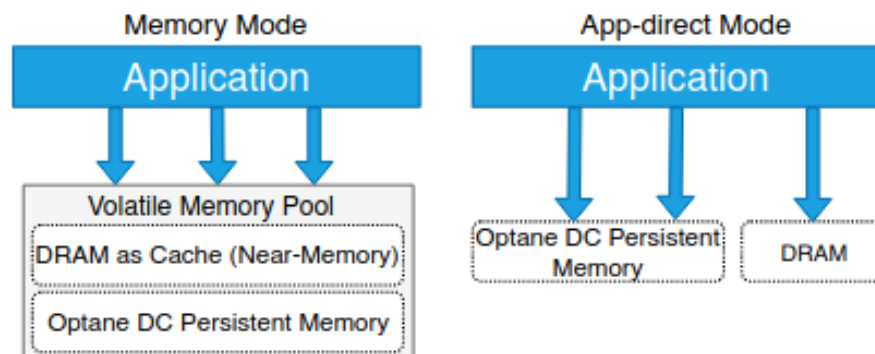


Figure 1.3.1: Διαφορετικές Λειτουργίες Μνήμης.[1]

Σε αυτήν την εργασία, η μνήμη Optane χρησιμοποιείται σε λειτουργία app direct ώστε να έχουμε την ευελιξία να διαχειριζόμαστε την τοποθέτηση των εφαρμογών και να παρακολουθούμε τις προσβάσεις και την μνήμη μέσω και των δύο τύπων μνήμης. Έχουμε τη δυνατότητα να αποφασίζουμε ποια δεδομένα αποθηκεύονται σε κάθε τύπο μνήμης. Οι δείκτες μπορούν να αποθηκευτούν στη μόνιμη μνήμη, αποφεύγοντας την ανάγκη επαναδημιουργίας τους κατά την επανεκκίνηση και τελικά επιτυγχάνεται σημαντική μείωση του χρόνου επανεκκίνησης. Επιπλέον, αποφεύγουμε την μείωση της απόδοσης λόγω αποτυχιών στις προσβάσεις (conflict misses), όταν η DRAM χρησιμοποιείται ως μνήμη cache για την PMEM.

Σε σύγκριση με τη μνήμη DDR4 DRAM, η καθυστέρηση της Optane αυξάνεται από 2x έως 6x για αναγνώσεις και από 6x έως 30x για εγγραφές, ανάλογα με το μοτίβο πρόσβασης, με καθυστέρηση ανάγνωσης που φτάνει έως και 374.1 ns και καθυστέρηση εγγραφής 391.2 ns. Οι προσβάσεις στη μνήμη που περιλαμβάνουν write-backs στην DCPMM εμφανίζουν καθυστέρηση 458.4 ns, δηλαδή 16.1% υψηλότερη από τις αποκλειστικά αναγνωστικές προσβάσεις. Όσον αφορά το εύρος ζώνης, η Optane DCPM παρουσιάζει μείωση περίπου 7% στις αναγνώσεις και 40% στις εγγραφές.[20]. Η μη πτητική κύρια μνήμη (NVMM) έχει σημαντικά χαμηλότερη αντοχή στις εγγραφές (10^8 κύκλοι) σε σύγκριση με τη DRAM (10^{16} κύκλοι), γεγονός που την καθιστά ακατάλληλη για άμεση αντικατάσταση της DRAM. Ωστόσο, από άποψη ενεργειακής απόδοσης, οι Optane DC PMMs καταναλώνουν λιγότερη ισχύ από τη DRAM όταν βρίσκονται σε αδράνεια, καθώς δεν απαιτούν συνεχή ανανέωση. Ένα δεύτερο σημείο αφορά την αξία τους: οι μονάδες μόνιμης μνήμης είναι πιο οικονομικές από τη DRAM. Συνεπώς, η Optane DCPM μπορεί να επιλεγεί για την κατασκευή διακομιστών με λιγότερη DRAM αλλά περισσότερη Optane DC μνήμη, ώστε να διατηρηθεί ένα ικανοποιητικό ή και μεγαλύτερο αποτύπωμα μνήμης για φορτία εργασίας που δεν απαιτούν απαραίτητα τη νανοδευτερόλεπτη καθυστέρηση της DRAM.

Παρά τα αδιαμφισβήτητα οφέλη των τεχνολογιών μη πτητικής μνήμης (NVM), οι περιορισμοί τους επιβάλλουν σημαντικές προκλήσεις για τους προγραμματιστές εφαρμογών. Οι NVM μνήμες παρουσιάζουν αυξημένη καθυστέρηση πρόσβασης, χαμηλό εύρος ζώνης και μειωμένη αντοχή σε σύγκριση με τις τεχνολογίες DRAM. Επιπλέον, εμφανίζουν ασυμμετρία στην καθυστέρηση και την κατανάλωση ενέργειας κατά τις λειτουργίες ανάγνωσης/εγγραφής, ενώ η άμεση αντικατάσταση της DRAM με NVMs, όπως η Intel Optane DCPM, μειώνει σημαντικά την απόδοση λόγω φαινομένων όπως ο περιορισμός εγγραφών (write throttling) και οι συγχρούσεις ταυτόχρονης προσπέλασης. Επομένως, οι NVM DIMMs συνδυάζονται συνήθως με μονάδες DRAM, δημιουργώντας ένα ετερογενές σύστημα DRAM/NVM.

1.3.2 Σχεσιακές βάσεις δεδομένων

Οι σχεσιακές βάσεις δεδομένων αποτελούν μια διαδεδομένη τεχνολογία εδώ και δεκαετίες για την αποθήκευση και την διαχείριση δεδομένων, χρησιμοποιώντας τη γλώσσα SQL (Structured Query Language) για τον συνδυασμό και την σύνοψη των δεδομένων[?]. Μια σχεσιακή βάση δεδομένων είναι ένας τύπος βάσης δεδομένων που οργανώνει τα δεδομένα σε γραμμές και στήλες, οι οποίες συλλογικά σχηματίζουν έναν πίνακα στον οποίο τα δεδομένα συνδέονται μεταξύ τους. Τα δεδομένα δομούνται σε πολλαπλούς πίνακες, οι οποίοι μπορούν να συνδεθούν μεταξύ τους μέσω ενός πρωτεύοντος κλειδιού (primary key) ή ενός ξένου κλειδιού (foreign key). Αυτά τα κλειδιά καταδεικνύουν τις διάφορες σχέσεις που υπάρχουν μεταξύ των πινάκων, οι οποίες συχνά απεικονίζονται μέσω διαφορετικών μοντέλων δεδομένων. Οι σχεσιακές βάσεις δεδομένων συσχετίζονται επίσης με τις συναλλακτικές (transactional) βάσεις δεδομένων, οι οποίες εκτελούν εντολές ή συναλλαγές συλλογικά. Οι συναλλαγές χαρακτηρίζονται από συγκεκριμένες ιδιότητες, γνωστές με το ακρωνύμιο ACID (Atomicity, Consistency, Isolation, Durability), δηλαδή Ατομικότητα, Συνέπεια, Απομόνωση και Διατηρησιμότητα. Κάθε συναλλαγή γίνεται ατομικά, σαν μια μοναδική λειτουργία, ανεξάρτητα και απομονωμένα από τις υπόλοιπες, ενώ παράλληλα εξασφαλίζεται η συνέπεια των δεδομένων καθόλη την διάρκεια των συναλλαγών, αλλά και η διατήρηση των αλλαγών στα δεδομένα.

Ωστόσο, η εμφάνιση νέων τεχνολογιών, όπως ο παγκόσμιος ιστός και το cloud, έφερε την ανάγκη διαχείρισης πιο σύνθετων δεδομένων σε βάσεις δεδομένων που μπορούν να υποστηρίξουν την κλιμακωσιμότητα και μια υψηλή απόδοση. Η τεχνολογία NewSQL είναι η πιο πρόσφατη εξέλιξη, η οποία ενσωματώνει την ικανότητα κλιμάκωσης και εξασφαλίζει τη διαθεσιμότητα που παρέχουν οι βάσεις NoSQL (Not Only SQL), διατηρώντας τις ιδιότητες ACID, οι οποίες χαρακτηρίζουν τις σχεσιακές βάσεις

δεδομένων[21]. Στο πλαίσιο αυτής της εργασίας, μελετήθηκε η βάση δεδομένων CockroachDB, η οποία αποτελεί ένα παράδειγμα μιας NewSQL σχεσιακής βάσης δεδομένων.

Η CockroachDB [22] είναι μια βάση δεδομένων βασισμένη στην γλώσσα SQL, είναι source-available και με την χρήση της η διαχείριση δεδομένων είναι απλή και αποδοτική. Χωρίς την χρήση εξειδικευμένου υλικού μπορεί να πραγματοποιεί οριζόντια σειριακή κλιμάκωση, δηλαδή κατανέμει τον φόρτο εργασιών εντός ενός server σε πολλά μηχανήματα του συστήματος ή σε πολλά nodes ενός cluster, σε αντίθεση με την κάθετη κλιμάκωση κατά την οποία αυξάνεται η χωρητικότητα και η ισχύς μιας μηχανής προκειμένου να μπορεί να διαχειριστεί αυξημένο φόρτο εργασιών. Η CockroachDB χρησιμοποιεί έναν κατανεμημένο αλγόριθμο συναίνεσης (βασισμένο στον Raft) για την αναπαραγωγή δεδομένων μεταξύ nodes, εξασφαλίζοντας ανθεκτικότητα και αυτόματη ανάκαμψη από αποτυχίες κόμβων (fault tolerance). Η CockroachDB είναι προσιτή στους χρήστες, καθώς κάνει χρήση μιας SQL διεπαφής συμβατή με την PostgreSQL, ενώ παράλληλα αξιοποιεί τον κατανεμημένο χαρακτήρα της με σκοπό την βελτιωμένη κλιμάκωση και απόδοση[23].

1.3.3 Αλγόριθμοι Επιβλεπόμενης Μηχανικής Μάθησης

Η μηχανική μάθηση αντιπροσωπεύει ένα ευρύ πεδίο ανάπτυξης αλγορίθμων που επιτρέπουν στον υπολογιστή να μαθαίνει και να προβλέπει μέσω μιας διαδικασίας εκπαίδευσης που βασίζεται στην ανεύρεση στατιστικών κανονικοτήτων ή άλλων προτύπων στα δεδομένα. Αυτοί οι αλγόριθμοι έχουν σχεδιαστεί ώστε να μπορούν να αναπαράγουν την ανθρώπινη προσέγγιση στην εκμάθηση μιας εργασίας.[24] Μπορούν να διακριθούν σε διάφορες κατηγορίες, ανάλογα με τη μέθοδο εκμάθησης ή το πρόβλημα που επιλύουν. Οι δύο πιο διαδεδομένες υποκατηγορίες είναι οι αλγόριθμοι επιβλεπόμενης και μη επιβλεπόμενης μάθησης. Η επιβλεπόμενη μάθηση αφορά την εκμάθηση μιας συνάρτησης που αντιστοιχίζει μία είσοδο σε μια έξοδο με βάση δείγματα εισόδου-εξόδου. Αντίθετα, οι μη επιβλεπόμενες μέθοδοι μαθαίνουν χαρακτηριστικά από τα δεδομένα και χρησιμοποιούν τα ήδη γνωστά χαρακτηριστικά για να αναγνωρίσουν την κλάση των δεδομένων. Κυρίως εφαρμόζονται για ομαδοποίηση και μείωση χαρακτηριστικών.

Σε αυτήν την εργασία, επιχειρούμε να ταξινομήσουμε και να χαρακτηρίσουμε εφαρμογές αξιοποιώντας τα δεδομένα που λαμβάνονται από την ανάλυση προφίλ των εφαρμογών. Η μοντελοποίηση της απόδοσης με χρήση εμπειρικής αξιολόγησης μπορεί να πραγματοποιηθεί μέσω αλγορίθμων επιβλεπόμενης μηχανικής μάθησης[25]. Στις μεθόδους επιβλεπόμενης μάθησης, χρησιμοποιείται ένα σύνολο δεδομένων, το οποίο έχει χαρακτηρισμένες τις κλάσεις που ανήκει το κάθε στοιχείο, για να επιτρέψει στον αλγόριθμο να μάθει τη συνάρτηση $f(x)$, η οποία αντιστοιχίζει τις εισόδους X_i σε μια έξοδο Y_i [25]. Το σύνολο δεδομένων διαχωρίζεται σε σύνολο εκπαίδευσης και σύνολο δοκιμών. Το σύνολο εκπαίδευσης περιλαμβάνει την μεταβλητή εξόδου που πρέπει να προβλεφθεί ή να ταξινομηθεί. Όλοι οι αλγόριθμοι ανιχνεύουν μοτίβα από το σύνολο εκπαίδευσης και τα εφαρμόζουν στο σύνολο δοκιμών για πρόβλεψη ή ταξινόμηση[2]. Στα προβλήματα ταξινόμησης, οι κλάσεις είναι προκαθορισμένες. Αυτές οι κλάσεις δημιουργούνται ως πεπερασμένο σύνολο, ορισμένο από τον άνθρωπο, που σημαίνει ότι ένα συγκεκριμένο τμήμα δεδομένων θα επισημανθεί με αυτές τις ταξινομήσεις [24],

Σκοπός μας σε αυτήν την εργασία είναι να ταξινομήσουμε τις εκτελούμενες εφαρμογές με βάση την απόδοσή τους και τη συμπεριφορά τους στη μνήμη, διακρίνοντάς τις σε memory-bound, cpu-bound ή bandwidth-bound εφαρμογές. Για να πραγματοποιήσουμε την ταξινόμηση, επιλέγουμε να εφαρμόσουμε και να συγκρίνουμε τέσσερις αλγορίθμους μηχανικής μάθησης, οι οποίοι βασίζονται σε διαφορετικές μεθόδους εκμάθησης και πρόβλεψης του τελικού αποτελέσματος. Αυτές οι μέθοδοι παρουσιάζονται παρακάτω.

1.3.4 Random Forest

Το μοντέλο μηχανικής μάθησης Random Forest είναι ένας ταξινομητής που αποτελείται από μια συλλογή ταξινομητών δομημένων ως δέντρα αποφάσεων $\{h(x, \Theta_k), k = 1, \dots\}$, όπου τα $\{\Theta_k\}$ είναι ανεξάρτητα και ομοιόμορφα κατανεμημένα τυχαία διανύσματα, και κάθε δέντρο προσδίδει μια ψήφο για την πιο δημοφιλή κλάση στην είσοδο x [26]. Τυχαία διαφορετικά μέρη του συνόλου δεδομένων λαμβάνονται για να εκπαιδευτεί κάθε δέντρο και στη συνέχεια τα αποτελέσματα συνδυάζονται με τη μέθοδο του μέσου όρου.

Κάθε δέντρο απόφασης κάνει προβλέψεις ανεξάρτητα. Κάθε κόμβος σε ένα δέντρο απόφασης αντιπροσωπεύει ένα χαρακτηριστικό μιας υπό εξέταση περίπτωσης, ενώ κάθε κλάδος αναπαριστά μια τιμή που μπορεί να λάβει το χαρακτηριστικό. Οι περιπτώσεις ταξινομούνται ξεκινώντας από τη ρίζα του δέντρου και κατηγοριοποιούνται με βάση τις τιμές των χαρακτηριστικών τους[27]. Αυτή η προσέγγιση ονομάζεται ensemble learning και συμβάλλει στη βελτίωση της ακρίβειας των προβλέψεων[?]. Η ακρίβεια ενός random forest εξαρτάται από τη δύναμη των μεμονωμένων δέντρων και από τον βαθμό εξάρτησης μεταξύ τους [26].

1.3.5 Support Vector Machine

Το μοντέλο (SVM) είναι μια τεχνική που διακρίνει τα δεδομένα και στοχεύει στην εύρεση μιας διαχωριστικής συνάρτησης, η οποία μπορεί να προβλέψει σωστά τις ετικέτες για νέα δεδομένα. Η βασική ιδέα είναι η εύρεση ενός υπερεπιπέδου (hyperplane) που διαχωρίζει τις κλάσεις μεγιστοποιώντας το περιθώριο μεταξύ τους[3]. Για μη γραμμικές συναρτήσεις, χρησιμοποιείται το "kernel trick" για τη χαρτογράφηση των δεδομένων σε υψηλότερες διαστάσεις, ώστε να βελτιωθεί η ακρίβεια των προβλέψεων[25].

1.3.6 Multi-Layer Perceptron

Το MLP είναι ένα είδος τεχνητού νευρωνικού δικτύου, το οποίο αποτελείται από μεγάλο αριθμό διασυνδεδεμένων νευρώνων που εργάζονται παράλληλα για την ταξινόμηση των δεδομένων εισόδου. Είναι οργανωμένο σε επίπεδα, με μια προωθητική ροή πληροφορίας. Ο αλγόριθμος εκτελείται έως ότου όλες οι προβλέψεις είναι σωστές στο σύνολο εκπαίδευσης, με τη χρήση της μεθόδου backpropagation για τη ρύθμιση των βαρών των συνάψεων[28].

1.3.7 Logistic Regression

Η λογιστική παλινδρόμηση (Logistic Regression) είναι ένας διακριτικός ταξινομητής που εξάγει χαρακτηριστικά από τα δεδομένα και τα συνδυάζει γραμμικά. Στόχος είναι η εξεύρεση ενός συνόλου βαρών που ελαχιστοποιεί την αρνητική λογαριθμική πιθανότητα μέσω τεχνικών βελτιστοποίησης όπως η στοχαστική κλίση (stochastic gradient descent)[29]. Η συνάρτηση κόστους δίνεται από:

$$J(\theta) = -\frac{1}{m} \sum [p_i \log(y_i) + (1 - p_i) \log(1 - y_i)]$$

1.4 Σκιαγράφηση Εφαρμογών

Πρωταρχικό βήμα για την εκπόνηση της εργασίας αποτελεί η σκιαγράφηση των εφαρμογών (workloads), οι οποίες βασίζονται σε transactions δεδομένων με την χρήση της CockroachDB. Η σκιαγράφηση (profiling) έχει σκοπό να διαμορφώσει μια εικόνα της συμπεριφοράς της κάθε εφαρμογής ως προς την διαχείριση της μνήμης και τις προσβάσεις σε αυτήν.

Στην παρούσα εργασία μελετάται η εκτέλεση τεσσάρων workloads βασισμένων στην CockroachDB. Τα workloads είναι τα εξής:

- **Bank:** Μοντελοποιεί ένα σύνολο λογαριασμών με τραπεζικά υπόλοιπα.
- **Kv:** Εκτελεί reads και writes στο σύνολο των κλειδιών που καταναίμονται σε ένα cluster.
- **TPCC:** Προσομοιώνει μια διαδικασία μεταβιβάσεων χρησιμοποιώντας ένα πλούσιο σχήμα πολλαπλών πινάκων.
- **YCSB:** Προσομοιώνει ένα υψηλής κλίμας key-value store workload, είτε read-heavy, write-heavy είτε συνδυασμό αυτών.

Πιο συγκεκριμένα, εξετάζεται το throughput - η ρυθμιαπόδοση - της δυναμικής DRAM και της μη πτητικής μνήμης κατά την εκτέλεση των παραπάνω εφαρμογών. Εξετάζονται επίσης οι προσβάσεις στην Persistent μνήμη είτε για αναγνώσεις είτε για εγγραφές και τέλος καταγράφεται η καθυστέρηση που

εισάγεται στο σύστημα με την εκτέλεση των διαφόρων queries που εκτελούνται κατά την διάρκεια των εφαρμογών. Τα βασικά εργαλεία που χρησιμοποιούνται για αυτόν τον σκοπό είναι το Intel® Performance Counter Monitor (Intel® PCM)[30] και το πακέτο ipmctl [31]. Τα πειράματα εκτελούνται σε έναν server με ενσωματωμένο πυρήνα 2x20 Intel Xeon Gold 5218R CPU @2.10GHz με 2 sockets and συνολικά 80 CPUs, με 4x32GB DDR4 DIMMs και 6x256GB Optane DC DIMMs. Το εικονικό σύστημα αποτελείται από 3 numa nodes, 2 είναι συνδεδεμένα με DRAM Dimms και 1 με την Optane DCPM. Intel Optane DCPM ρυθμίζεται σε App-Direct mode με το EXT4-DAX file-system. Version 1.11 of the Persistent Memory Development Kit (PMDK) and gcc-13.3 εφαρμόζονται για την ανάπτυξη των εκάστοτε εφαρμογών.

1.4.1 Διεξαγωγή Πειραμάτων σε Απομονωμένο Σύστημα

Σε αυτό το μέρος, εκτελούμε τις τέσσερις εφαρμογές ανεξάρτητα δύο φορές, μια χρησιμοποιώντας DRAM και μια χρησιμοποιώντας την Optane DCPM. Οι δυο διαφορετικές εκτελέσεις πραγματοποιούνται με την χρήση της εντολής numactl –membind, η οποία διασφαλίζει το numa node στο οποίο θα εκτελεστεί η εφαρμογή. Ερευνούμε την συμπεριφορά των εφαρμογών ως προς την μνήμη και την απόδοσή τους στο απομονωμένο σύστημα. Κάθε εφαρμογή εκτελείται για 5 λεπτά με μια διαφορετική τιμή νημάτων. Γίνεται η χρήση 1,2,4,8,16,32,64 και 80 νημάτων και τελικά καταγράφουμε το throughput της μνήμης, τις προσβάσεις στην persistent μνήμη και τις εξόδους των εφαρμογών.

Πρώτα από όλα, με την χρήση της εντολής ipmctl παρατηρούμε τις προσβάσεις στην Optane DCPM. Τα αποτελέσματα φαίνονται στο διάγραμμα 5.1.1. Μπορεί να παρατηρηθεί ότι οι προσβάσεις στην persistent μνήμη για όλες τις εφαρμογές κυμαίνονται μεταξύ 10^{10} και 10^{11} με τις αναγνώσεις να ξεπερνούν πάντα τις εγγραφές. Μάλιστα όσο αυξάνεται ο αριθμός των νημάτων, οι προσβάσεις στην μνήμη αυξάνονται, χωρίς να παρατηρούνται ακραίες μεταβολές. Οι εφαρμογές Bank, Kv, Trcc παρουσιάζουν ένα σταθερό μοτίβο με μικρές αυξήσεις ανάλογα με τον αριθμό των νημάτων. Μάλιστα, αυτές οι εφαρμογές παρουσιάζουν τις λιγότερες προσβάσεις στην μνήμη. Αντίθετα, η εφαρμογή Ycsb, και ιδιαίτερα ο τύπος A και B, σταθερά παρουσιάζει τις υψηλότερες τιμές.

Στην συνέχεια, από τις εξόδους των εφαρμογών συλλέγουμε τις καθυστερήσεις που εισάγουν τα queries στο σύστημα. Παρατηρώντας τα διαγράμματα 5.2.2 και 5.2.3 στο κεφάλαιο 5 5, μπορούμε να πούμε ότι οι καθυστερήσεις παρουσιάζουν μεγάλο εύρος τιμών και δεν συγκεντρώνονται γύρω από μια συγκεκριμένη τιμή. Οι τιμές κυμαίνονται μεταξύ 10 και 10^5 ms κατά την εκτέλεση στην DRAM και κατά την εκτέλεση στην Optane. Οι εφαρμογές Bank και Trcc εισάγουν μεγαλύτερες καθυστερήσεις στο σύστημα και η Ycsb τις μικρότερες, ιδιαίτερα ο τύπος C, που εκτελεί μόνο αναγνώσεις στην μνήμη. Όσο αυξάνεται ο αριθμός των νημάτων, οι καθυστερήσεις τείνουν να αυξάνονται και το εύρος των τιμών μεγαλώνει.

Τελικά, μέσω του εργαλείου PCM-Memory και του PCM, εξετάζουμε το throughput της κάθε μνήμης για κάθε εφαρμογή και στις δύο εκτελέσεις, σε DRAM και σε Optane, για όλες τις τιμές των νημάτων. Τα αποτελέσματα παρουσιάζονται αναλυτικά στα διαγράμματα του κεφαλαίου 5. Διαχωρίζονται οι αναγνώσεις και οι εγγραφές στην DRAM και στην Optane και για στις δύο εκτελέσεις. Μελετώντας τα διαγράμματα για όλες τις εφαρμογές, μπορούμε να συμπεράνουμε ότι όλες οι εφαρμογές παρουσιάζουν μια ασύμμετρη συμπεριφορά όσον αφορά στις προσβάσεις στην μνήμη και στην απόδοση στην DRAM και στην Optane DCPM, με αρκετές αυξομειώσεις ανά δευτερόλεπτο. Σε όλες τις εφαρμογές, και στις δύο εκτελέσεις το throughput της persistent μνήμης είναι χαμηλότερο από το throughput της DRAM. Ωστόσο, όσο οι τιμές του throughput μειώνονται, η διαφορά αυτή μικραίνει και παρατηρείται μια παρόμοια απόδοση. Μπορούμε να παρατηρήσουμε ακόμη, ότι το PMM throughput παραμένει παρόμοιο με την αύξηση των νημάτων εκτέλεσης, οι εφαρμογές δηλαδή δεν κλιμακώνουν σωστά. Πιθανώς, η NVM μνήμη να προκαλεί ένα bottleneck. Οι εγγραφές στην μνήμη είναι μικρότερες από τις αναγνώσεις σχεδόν σε όλες τις περιπτώσεις. Η εφαρμογή με το χαμηλότερο throughput προκύπτει να είναι η Trcc. Πιθανή αιτία της χαμηλής απόδοσής της είναι ότι τα queries, κατά την διάρκεια εκτέλεσης, δεν εκτελούνται κάθε δευτερόλεπτο, αλλά εκτελούνται ανα ταχτά, πιο αραιά, χρονικά διαστήματα, της τάξης των 5 δευτερολέπτων. Το υψηλότερο throughput παρατηρείται κατά την εκτέλεση της εφαρμογής Ycsb, ιδιαίτερα στον τύπο A.

1.4.2 Διεξαγωγή Πειραμάτων σε Interference-aware Σύστημα

Σε αυτό το σημείο, εισάγουμε την ιδέα της πειραματικής ανάλυσης και του χαρακτηρισμού των εφαρμογών σε ένα σύστημα με επίγνωση των πιθανών παρεμβολών ενός υπολογιστικού συστήματος. Σε ένα ρεαλιστικό υπολογιστικό σύστημα, πολλές εφαρμογές μπορεί να εκτελούνται παράλληλα, οι επεξεργαστές μπορεί να εκτελούν διαφορετικές διεργασίες την ίδια στιγμή και το εύρος ζώνης της μνήμης πολλές φορές εξυπηρετεί αιτήματα άλλων εφαρμογών. Προκειμένου να προσομοιώσουμε ένα τέτοιο σύστημα με επίγνωση των πιθανών παρεμβολών, εκτελούμε παράλληλα με τις εφαρμογές, κάποια ειδικά benchmarks τα οποία καταλαμβάνουν τους πόρους του συστήματος και έπειτα εξετάζουμε τα αποτελέσματα.

iBench Benchmarks

Το IBench [32] αποτελείται από ένα σύνολο προσεκτικά σχεδιασμένων benchmarks τα οποία προσθέτουν παρεμβολές κλιμακούμενης έντασης στους πόρους του συστήματος. Συγκεκριμένα, καλύπτουν ένα ένα σταδιακά όλα τα νήματα του επεξεργαστή, καταλαμβάνουν την χωρητικότητα των μνημών σε όλα τα επίπεδα της ιεραρχίας, καθώς και το σύστημα δικτύων. Υπάρχουν επτά benchmarks και κάθε benchmark καλύπτει έναν πόρο του συστήματος ξεχωριστά χωρίς να επικαλύπτονται. Κάθε benchmark ασκεί προοδευτικά μεγαλύτερη πίεση σε έναν συγκεκριμένο πόρο του συστήματος. Τα benchmarks αφορούν τα 'cpu', 'memBw', 'memCap', 'l1d', 'l1i', 'l2' και 'l3'. Σε αυτήν την εργασία, εχμεταλλευόμαστε τρία από αυτά, τα l3, memBw και cpu. Έπειτα εκτελούμε παράλληλα με αυτά τις εφαρμογές με τον ίδιο τρόπο όπως στο απομονωμένο σύστημα, για δύο εκτελέσεις σε DRAM και Optane με τιμές νημάτων 1,2,4,8,16,32,64 και 80. Το L3 benchmark αρχικοποιεί ένα μπλοκ στο μέγεθος της κρυφής μνήμης και ενώ εκτελείται αποκτά πρόσβαση στο μπλοκ μνήμης, μέσω της εντολής memcopy. Το MemBw ορίζει έναν πίνακα μεγέθους 2N και εκτελεί αναγνώσεις στο ένα μισό του πίνακα και γράφει στο άλλο μισό του πίνακα παράλληλα. Τέλος, το Cpu απασχολεί σταδιακά τα νήματα του συστήματος.

Αποτελέσματα

Πρώτα απ' όλα, με την χρήση της εντολής ipmctl, παίρνουμε μετρήσεις από τις προσβάσεις στην persistent μνήμη κατά τη εκτέλεση της κάθε εφαρμογής με την παράλληλη εκτέλεση κάθε benchmark ξεχωριστά. Τα αποτελέσματα παρουσιάζονται στα σχεδιαγράμματα 5.2.1a, 5.2.1b και 5.2.1c. Μπορούμε να παρατηρήσουμε ότι με την παράλληλη εκτέλεση των bandwidth και L3 benchmarks οι συνολικές προσβάσεις στην μνήμη αυξάνονται. Παρατηρείται, επομένως, μια μείωση της αποδοσης των εφαρμογών. Αντίθετα, με το cpu benchmark οι προσβάσεις στην μνήμη παραμένουν σε παρόμοιο εύρος τιμών με την εκτέλεση σε απομονωμένο περιβάλλον. Σε όλες τις περιπτώσεις, οι αναγνώσεις στην μνήμη ξεπερνούν τις εγγραφές της μνήμης. Επιπλέον, οι συνολικές προσβάσεις στην μνήμη αυξάνονται καθώς ο αριθμός των νημάτων αυξάνεται, χωρίς ακραίες μεταβολές. Κατά την εκτέλεση των benchmarks L3 και Bandwidth, η εφαρμογή με τα περισσότερα accesses είναι η Ycsb με τιμές που κυμαίνονται στο διάστημα $(0.9 - 2.5) \times (10^{11})$ για τις αναγνώσεις και στο διάστημα $(0.4 - 1.5) \times (10^{11})$ για τις εγγραφές. Αντίστοιχα, για τις άλλες εφαρμογές οι τιμές των αναγνώσεων είναι $(0.1 - 0.3) \times (10^{11})$ και των εγγραφών $(0.1 - 0.2) \times (10^{11})$. Στο cpu benchmark οι προσβάσεις μειώνονται. Οι χαμηλότερες τιμές παρατηρούνται στα Trcc και Ycsb F στο εύρος $(0.7 - 1.5) \times (10^{10})$, ενώ οι υψηλότερες εμφανίζονται στους υπόλοιπους τύπους της Ycsb εφαρμογής με τιμές στο διάστημα $(3 - 4) \times (10^{10})$ για τις αναγνώσεις και $(0.6 - 1.1) \times (10^{10})$ για τις εγγραφές.

Στην συνέχεια, καταγράφουμε και πάλι τις καθυστερήσεις που εισάγονται στο σύστημα με την εκτέλεση των queries των εφαρμογών κατά την παράλληλη εκτέλεση των benchmarks. Από τα διαγράμματα 5.2.3 και 5.2.2, μπορεί να παρατηρηθεί ότι οι καθυστερήσεις αυτές δεν μεταβάλλονται πολύ συγκριτικά με την εκτέλεση στο απομονωμένο περιβάλλον. Οι τιμές κυμαίνονται στο εύρος $(10 - 10^5)$ ms και για τις δύο εκτελέσεις. Στην εκτέλεση στην Optane DCPM οι καθυστερήσεις αυτές τείνουν να είναι υψηλότερες. Για κάθε εφαρμογή, οι τιμές φαίνονται να έχουν μια μεγάλη διασπορά και να μην παρουσιάζουν μια συγκεκριμένη συμπεριφορά ή να συγκεντρώνονται γύρω από μια κεντρική τιμή. Η Kv εφαρμογή εισάγει τις μικρότερες καθυστερήσεις στο σύστημα, ενώ η Trcc και η Bank από την τιμή συγχρονισμού 32 και μετά, εισάγουν τις πιο υψηλές τιμές καθυστερήσεων και το μεγαλύτερο εύρος τιμών.

Κάτα την μελέτη του throughput των μνημών κατά τις εκτελέσεις των εφαρμογών παράλληλα με τα benchmarks, διαπιστώσαμε ότι το σύστημα πραγματικά επηρεάζεται από την ύπαρξη παρεμβολών και δεν παρουσιάζονται ακραίες μεταβολές στις τιμές της απόδοσης των εφαρμογών. Έτσι, προκειμένου να μελετήσουμε την επιρροή των benchmarks στις εφαρμογές, καταγράψαμε μέσα από τις εξόδους των εφαρμογών, τις εντολές που εκτελεί η κάθε εφαρμογή ανά δευτερόλεπτο καθόλη την διάρκεια εκτέλεσής τους. Απο τα διαγράμματα αυτά, τα οποία παρουσιάζονται στο κεφάλαιο 5 καταλαβαίνουμε ότι τα benchmarks επηρεάζουν την απόδοση των εφαρμογών στις περισσότερες περιπτώσεις. Οδηγούν είτε σε αύξηση των εντολών, καθώς πιθανά έχει δημιουργηθεί κάποιου είδους bottleneck στο σύστημα, αλλά και σε μείωση των εντολών σε κάποιες περιπτώσεις, το οποίο αντιστοιχεί σε αντίστοιχη μείωση απόδοσης. Τα benchmarks που παρουσιάζουν την μεγαλύτερη επίδραση στην λειτουργία των εφαρμογών είναι το Bandwidth και το CPU benchmark, ενώ το L3 παρουσιάζει μέτρια επίδραση στην απόδοση των εφαρμογών. Η επίδραση ωστόσο των benchmarks οφείλεται όχι μόνο στο είδος του benchmark που εκτελούμε, αλλά και στην μνήμη στην οποία εκτελούνται οι εφαρμογές. Στην Optane μνήμη, παρατηρείται μεγαλύτερη επιβάρυνση της απόδοσης, ενώ στην DRAM μνήμη κάποιες φορές μπορεί να μην παρατηρηθεί καν επιβάρυνση και να παρουσιάζεται μια αρκετά παρόμοια συμπεριφορά. Επιπλέον, παρατηρούμε ότι με την αύξηση του αριθμού των νημάτων, σε όλες τις εφαρμογές ο αριθμός των εντολών ανά δευτερόλεπτο αυξάνεται.

1.5 Προτεινόμενη Μεθοδολογία

Σε αυτό το μέρος, παρουσιάζεται η προτεινόμενη μεθοδολογία προκειμένου να αποφασιστεί η τοποθέτηση των εφαρμογών στις μνήμες του ετερογενούς συστήματος. Ειδικότερα, εισάγουμε την ιδέα της εκπαίδευσης μοντέλων μηχανικής μάθησης με σκοπό την ταξινόμηση και τον χαρακτηρισμό των εφαρμογών με βάση την απόδοση και την συμπεριφορά τους ως προς την διαχείριση της μνήμης. Οι κλάσεις στις οποίες θα διαχωρίζουμε τις εφαρμογές είναι οι: Memory bound, DRAM-Bandwidth Bound, PMM-Read Bandwidth Bound, PMM-Write Bandwidth Bound, CPU-Bound. Με βάση αυτή την ταξινόμηση, μπορούν να παρθούν αποφάσεις για την τελική τοποθέτηση των εφαρμογών και την διαχείριση των μνημών στο ετερογενές σύστημα μνήμης.

Το πρώτο βήμα για την εκπαίδευση των μοντέλων μηχανικής μάθησης είναι η δημιουργία ενός κατάλληλου συνόλου δεδομένων εκπαίδευσης. Συλλέγουμε τα δεδομένα εισόδου από τα δεδομένα που έχουν αντληθεί κατά την πειραματική ανάλυση των εφαρμογών με την βοήθεια των εργαλείων PCM και PCM Memory, τα οποία παράγουν δεδομένα κάθε δευτερόλεπτο κατά την εκτέλεση των εφαρμογών. Τα δεδομένα αυτά περιέχουν πληροφορίες για το σύστημα, τον επεξεργαστή, την κατανάλωση μνήμης και τις προσβάσεις σε όλη την ιεραρχία των μνημών. Επιλέγουμε να χρησιμοποιήσουμε τα δεδομένα που προκύπτουν από την ανάλυση στο σύστημα με επίγνωση των παρεμβολών. Από αυτά τα δεδομένα συλλέγουμε τα δεδομένα που αναφέρονται στις μετρικές του συστήματος από το PCM και το PCM Memory και τα συγκεντρώνουμε σε ένα σύνολο δεδομένων. Ο επόμενος στόχος είναι να καθορίσουμε την κλάση του κάθε στοιχείου του συνόλου δεδομένου.

Για να ταξινομήσουμε τα δεδομένα εισόδου στις κλάσεις και έπειτα να εκπαιδεύσουμε τα μοντέλα να προβλέπουν τις κλάσεις των δεδομένων, είναι απαραίτητο να ορίσουμε συγχεκριμένα όρια-thresholds στις μετρικές της μνήμης και του επεξεργαστή. Τα όρια αυτά βασίζονται στην τιμή του IPC προκειμένου να αποφασίσουμε αν μια εφαρμογή είναι cpu bound ή όχι. Για να αποφασίσουμε αν μια εφαρμογή είναι memory-bound ή όχι, εξετάζουμε το ποσοστό των αστοχιών στο τελευταίο επίπεδο κρυφών μνημών L3 (L3 miss rate) σε σχέση με τις συνολικές προσβάσεις σε αυτήν κατά την διάρκεια της εκτέλεσης των εφαρμογών στην DRAM και στην Optane. Στο τέλος, υπολογίζουμε το μέγιστο εύρος ζώνης που μπορεί να επιτευχθεί στην DRAM και στην Optane για τις αναγνώσεις και τις εγγραφές και τελικά το συγκρίνουμε με το εύρος ζώνης των εφαρμογών προκειμένου να κατατάξουμε τις εφαρμογές σε DRAM bandwidth bound, PMM Read bandwidth bound, PMM Write bandwidth bound ή τίποτα από αυτά.

Για τον υπολογισμό του μέγιστου εύρους ζώνης των μνημών εκτελούμε συγκεκριμένα benchmarks που χρησιμοποιούν στο μέγιστο το bandwidth των μνημών. Συγκεκριμένα, εκμεταλλευόμαστε το Intel®

Memory Latency Checker and την εντολή 'mlc -max_bandwidth', η οποία εκτυπώνει το μέγιστο bandwidth των μνημών για διαφορετικά ποσοστά αναγνώσεων και εγγραφών. Έτσι προκύπτουν οι παρακάτω τιμές:

DRAM Max Bw	42324.47 MB/s
Optane Max Read Bw	1700.55 MB/s
Optane Max Write Bw	930.76 MB/s

Table 1.1: Τιμές Μέγιστου Bandwidth

Στον πίνακα 6.2 φαίνεται πιο αναλυτικά ο ορισμός των κλάσεων και στο διάγραμμα 6.0.1 παρουσιάζεται η τελική κατανομή των δεδομένων εκπαίδευσης στις κλάσεις. Η πλειονότητα των εφαρμογών ταξινομούνται ως memory-bound. Τα υπόλοιπα δείγματα κατανέμονται στις κλάσεις PMM Read ή Write Bound και CPU-bound. Τα λιγότερα δείγματα ανήκουν στις κλάσεις DRAM BW Bound και στην undefined, η οποία υποδηλώνει μια πιο βασική λειτουργία των εφαρμογών.

Στην συνέχεια, εφαρμόζουμε ποσοστιαίο διαχωρισμό των δεδομένων εκπαίδευσης σε δυο υποσύνολα, το σύνολο της εκπαίδευσης και το σύνολο των δοκιμών με παράγοντα τυχαιότητας 42. Χρησιμοποιούμε το 70% των δεδομένων για την εκπαίδευση των μοντέλων και το 30% των δεδομένων για τις δοκιμές των εκπαιδευμένων μοντέλων. Η σύγκριση των κλάσεων που έχει προβλέψει το μοντέλο και των πραγματικών κλάσεων των δεδομένων θα αποτελέσει βασικό μέσο αξιολόγησης της ακρίβειας των μοντέλων. Τα δεδομένα που αξιοποιούμε για την εκπαίδευση των μοντέλων είναι τα δεδομένα από τις εφαρμογές Bank και Trcc. Επιπλέον, ένα σημαντικό βήμα είναι η προεπεξεργασία των αριθμητικών δεδομένων για την διευκόλυνση της εκπαίδευσης των μοντέλων, ειδικά εκείνων που βασίζονται στην μέθοδο της οπισθοδιάδοσης και των επαναλήψεων, όπως το μοντέλο MLP και Logistic Regression. Για αυτόν τον σκοπό, γίνεται χρήση της συνάρτησης SimpleImputer(), η οποία συμπληρώνει με την μέση τιμή ενός συγκεκριμένου χαρακτηριστικού τις κενές τιμές και της συνάρτησης StandardScaler(), η οποία κανονικοποιεί τα αριθμητικά δεδομένα αφαιρώντας την μέση τιμή από κάθε χαρακτηριστικό και διαιρώντας το με την τυπική απόκλιση του χαρακτηριστικού. Με αυτόν τον τρόπο, αποτρέπονται σφάλματα που μπορεί να οφείλονται σε ελλιπείς τιμές ή στην κυριαρχία συγκεκριμένων χαρακτηριστικών κατά την εκπαίδευση των μοντέλων.

Επιλέγουμε να ερευνήσουμε την απόδοση τεσσάρων μοντέλων επιβλεπόμενης μηχανικής μάθησης: MLP, Random Forests, SVM and Logistics Regression. Αυτοί οι αλγόριθμοι βασίζονται σε διαφορετικές μεθόδους εκπαίδευσης και είναι κατάλληλα και για μη γραμμικά προβλήματα ταξινόμησης. Μπορούν να υποστηρίξουν την επεξεργασία μεγάλου όγκου δεδομένων και την εκπαίδευσή τους ώστε να προβλέπουν πολλαπλές κλάσεις δεδομένων.

Πριν την έναρξη της εκπαίδευσης των μοντέλων, είναι σημαντικό να προσδιορίσουμε τις παραμέτρους των μοντέλων, ώστε να αυξήσουμε την απόδοσή τους και να επιλεγεί ο καλύτερος συνδυασμός. Ειδικότερα, για το Random Forest εκτελούμε GridSearch, εξετάζοντας τις τιμές [50,100,150,200,250] για τον αριθμό των δέντων εκτίμησης και τις τιμές [10,15,20,25,30] για το μέγιστο βάθος. Για το μοντέλο SVM(), εξετάζουμε τους πυρήνες ['rbf','sigmoid','poly'] και τις τιμές [0.1,1,10,100] για τον παράγοντα κανονικοποίησης C. Για το μοντέλο MLP, δοκιμάζουμε τις συναρτήσεις ενεργοποίησης ['tanh','relu','logistic'] και τους βελτιστοποιητές ['adam','lbfgs','sgd']. Τέλος, για το Logistic Regression, χρησιμοποιούμε τον ταξινομητή OneVsRestClassifier για τις πολλαπλές κλάσεις και δοκιμάζουμε τριών διαφορετικών βελτιστοποιητών ['newton-cg','saga','lbfgs'] και τις τιμές [0.1,1,10,100] για τον παράγοντα C.

1.6 Αξιολόγηση

Σε αυτό το κεφάλαιο, παρουσιάζεται η αξιολόγηση των μοντέλων μετά την εκπαίδευσή τους. Προκειμένου να αξιολογήσουμε τα διαφορετικά μοντέλα μηχανικής μάθησης και να καταλήξουμε στην τελική επι-

λογή του μοντέλου αναλύουμε διάφορες μετρικές. Πρώτα από όλα, εξετάζουμε την απόδοση των μοντέλων ανάλογα με τις υπερπαραμέτρους που έχουμε επιλέξει να διερευνήσουμε. Για αυτόν τον σκοπό, είναι σημαντικό να λάβουμε υπόψη την ακρίβεια των μοντέλων, αλλά και τον μέσο χρόνο εκπαίδευσης. Άλλωστε η τελική επιλογή του μοντέλου είναι ένας συνυπολογισμός ακρίβειας και χρόνου εκπαίδευσης. Σε επόμενο βήμα, αξιολογούμε τα μοντέλα με βάση το accuracy, το precision και το recall στο σύνολο δεδομένων δοκιμών. Το Accuracy υπολογίζει πόσες κλάσεις προβλέπονται σωστά, το precision υπολογίζει τον αριθμό των δειγμάτων που προβλέπονται να ανήκουν σε μια κλάση και πραγματικά ανήκουν σε αυτήν και το recall μετρά πόσα δείγματα ανά κλάση έχουν προβλεφθεί σωστά. Τέλος, αξιολογούμε την ακρίβεια των μοντέλων και στα δεδομένα των εφαρμογών από την πειραματική ανάλυση, στα οποία δεν έχουν εκπαιδευτεί τα μοντέλα.

Πρώτα από όλα, με βάση την επιλογή των παραμέτρων που περιγράφονται στο κεφάλαιο 6 εκτελούμε GridSearch και συγκεντρώνουμε τα αποτελέσματα σε heatmaps. Τα heatmaps αναδεικνύουν τον τρόπο με τον οποίο μεταβάλλονται η ακρίβεια και ο μέσος χρόνος εκπαίδευσης των μοντέλων ανάλογα με την επιλογή των παραμέτρων. Στα heatmaps 7.0.1 μπορεί να παρατηρηθεί ότι το Random Forest επιτυγχάνει τα υψηλά ποσοστά ακρίβειας για όλες τις τιμές των παραμέτρων. Ο μέσος χρόνος εκπαίδευσης ωστόσο αυξάνεται ανάλογα με τον αριθμό των εκτιμητών. Όσο ο αριθμός των εκτιμητών-των επιμέρους δέντρων αποφάσεων- αυξάνεται, ο χρόνος εκπαίδευσης αυξάνεται. Δεδομένου ενός αριθμού εκτιμητών, το μέγιστο βάθος δεν επηρεάζει τον χρόνο εκπαίδευσης, αλλά όσο το μέγιστο βάθος μεγαλώνει παρατηρείται και σχετική αύξηση στην ακρίβεια του μοντέλου.

Το μοντέλο SVM, όπως μπορούμε να δούμε στα heatmaps 7.0.2, παρουσιάζει χαμηλό χρόνο εκπαίδευσης για όλες τις παραμέτρους και ποσοστά ακρίβειας πάνω από 87%. Ειδικότερα, ο πυρήνας-kernel 'rbf' επιτυγχάνει την υψηλότερη ακρίβεια και όσο ο παράγοντας C αυξάνεται το ποσοστό ακρίβειας αυξάνεται αντίστοιχα. Για όλα τα kernels, η αύξηση του παράγοντα C οδηγεί σε αύξηση της ακρίβειας. Ο χρόνος εκπαίδευσης παραμένει σε όλες τις περιπτώσεις αρκετά χαμηλός, κάτω από 1 δευτερόλεπτο, ιδιαίτερα για τον πυρήνα 'sigmoid'.

Από τα heatmaps 7.0.3, μπορούμε να παρατηρήσουμε ότι το μοντέλο MLP επιτυγχάνει υψηλά ποσοστά ακρίβειας, αλλά την ίδια στιγμή οι χρόνοι εκπαίδευσης είναι αρκετά μεγάλοι. Ιδιαίτερα, για τον βελτιστοποιητή 'sgd' και για την συνάρτηση ενεργοποίησης 'tanh' ο μέσος χρόνος εκπαίδευσης λαμβάνει την μέγιστη τιμή. Επίσης, στον βελτιστοποιητή 'sgd' τα ποσοστά ακρίβειας είναι τα χαμηλότερα. Η αλλαγή των συναρτήσεων ενεργοποίησης δεν επιφέρει μεγάλη αλλαγή στην ακρίβεια των μοντέλων.

Τα heatmaps 7.0.4 δείχνουν ότι το μοντέλο Logistic Regression επιτυγχάνει τα χαμηλότερα σκορ ακρίβειας συγκριτικά με τα υπόλοιπα μοντέλα. Επιπλέον, οι χρόνοι εκπαίδευσης δεν είναι αρκετά χαμηλοί. Ιδιαίτερα για τον βελτιστοποιητή 'saga' ο χρόνος αυτός αποδεικνύεται υψηλός. Τα καλύτερα ποσοστά ακρίβειας και οι χαμηλότεροι χρόνοι εκπαίδευσης σημειώνονται με τους βελτιστοποιητές 'lbfgs' και 'newton-cg'. Μάλιστα, όσο ο παράγοντας C αυξάνεται η ακρίβεια του μοντέλου αυξάνεται.

Η τελική παραμετρικοποίηση των μοντέλων παρουσιάζεται στον πίνακα 7.1. Αφού πραγματοποιηθεί η εκπαίδευση των μοντέλων με αυτές τις παραμέτρους, εξετάζεται η ακρίβεια που σημειώνουν τα μοντέλα στο σύνολο δεδομένων δοκιμής, αλλά και στα δεδομένα των εφαρμογών στις οποίες δεν έχουν εκπαιδευτεί τα μοντέλα. Τα ποσοστά ακρίβειας των μοντέλων στο σύνολο δοκιμής παρουσιάζονται στον πίνακα 7.2 και τα ποσοστά ακρίβειας στα δεδομένα των εφαρμογών συνοψίζονται στον πίνακα 7.3.

Από την μελέτη αυτών των ποσοστών ακρίβειας, μπορεί να παρατηρηθεί ότι όλα τα μοντέλα επιτυγχάνουν υψηλά ποσοστά ακρίβειας, πάνω από 90%. Μάλιστα, τα μοντέλα MLP και SVM καταφέρνουν να φτάσουν τις υψηλότερες τιμές ακρίβειας σε όλα τα σύνολα δεδομένων όπου δοκιμάζονται. Οπότε θα μπορούσαν να είναι κατάλληλες λύσεις για την πρόβλεψη της συμπεριφοράς των εφαρμογών. Ωστόσο, αναλογιζόμενοι τους χρόνους εκπαίδευσης, το μοντέλο MLP παρουσιάζει αρκετά υψηλότερους χρόνους εκπαίδευσης και πολυπλοκότητας συγκριτικά με τους χρόνους εκπαίδευσης του SVM. Μπορούμε, λοιπόν,

να ισχυριστούμε ότι η επιλογή του μοντέλου SVM θα ήταν μια πιο ικανοποιητική και αποδοτική επιλογή μοντέλου μηχανικής μάθησης.

Με βάση την ταξινόμηση των εφαρμογών, θα μπορούσαν να παρθούν αποφάσεις για τον τρόπο τοποθέτησης των εφαρμογών στις μνήμες του ετερογενούς συστήματος. Οι εφαρμογές που οριοθετούνται από την μνήμη (memory-bound), ιδιαίτερα αυτές που δεν εισάγουν υψηλές καθυστερήσεις στο σύστημα, θα μπορούσαν να τοποθετηθούν στην μνήμη Optane DCPM λόγω της αυξημένης χωρητικότητάς της συγκριτικά με την μνήμη DRAM. Αντίθετα, εφαρμογές που περιορίζονται από τον επεξεργαστή του συστήματος (cpu-bound) θα ήταν προτιμότερο να τοποθετούνται στην μνήμη DRAM λόγω της αυξημένης ταχύτητας ανταπόκρισής της στο σύστημα. Οι εφαρμογές που περιορίζονται από το εύρος ζώνης είτε της DRAM μνήμης είτε της Optane DCPM θα μπορούσαν να τοποθετούνται ανάλογα στην DRAM και να μεταφέρονται κατά την διάρκεια εκτέλεσης στην Optane. Άλλωστε, η συμπεριφορά όλων των εφαρμογών είναι δυναμική και μπορεί να μεταβάλλεται κατά την εκτέλεσή τους. Έτσι θα πρέπει να γίνονται προβλέψεις ανα ταχτά χρονικά διαστήματα και να λαμβάνονται εκ νέου οι αποφάσεις τοποθέτησης και διαχείρισης του συστήματος μνημών.

1.7 Συμπεράσματα και μελλοντικές επεκτάσεις

1.7.1 Συμπεράσματα

Συνοψίζοντας, στην παρούσα εργασία παρουσιάζεται η εκτέλεση, η πειραματική ανάλυση και ο χαρακτηρισμός εφαρμογών βάσεων δεδομένων σε ένα σύστημα ετερογενούς μνήμης. Τα πειράματα διεξήχθησαν σε ένα απομονωμένο σύστημα και σε ένα σύστημα με επίγνωση πιθανών παρεμβολών, οι οποίες επιτυγχάνονται με την εκτέλεση παράλληλων προγραμμαμάτων που αξιοποιούν πλήρως τους πόρους του συστήματος. Κάθε εφαρμογή εκτελέστηκε ξεχωριστά τόσο στην DRAM όσο και στην Optane DCPM. Με την ολοκλήρωση και την ανάλυση των εκτελέσεων αυτών, μπορεί να παρατηρηθεί ότι κάθε εφαρμογή παρουσιάζει διαφορές στην απόδοσή της ανάλογα με την μνήμη στην οποία εκτελείται. Όλες οι εφαρμογές παρουσιάζουν μια ασυμμετρία στις προσβάσεις στην μνήμη και στην καθυστέρηση που εισάγουν στο σύστημα. Μάλιστα, η ύπαρξη παρεμβολών στο σύστημα οδηγεί σε επιβάρυνση στην απόδοση των εφαρμογών στις περισσότερες περιπτώσεις. Προκύπτει, λοιπόν, ότι η συμπεριφορά τους δεν μπορεί να προβλεφθεί εύκολα και εξαρτάται από πολλαπλούς παράγοντες. Για τον σκοπό αυτό, πραγματοποιείται η εκπαίδευση μοντέλων μηχανικής μάθησης για να ταξινομούν αυτές τις εφαρμογές. Τα δεδομένα εκπαίδευσής τους συλλέγονται καθόλη την διάρκεια εκτέλεσης των εφαρμογών από τα εργαλεία καταγραφής της απόδοσης (pcm, pcm-memory). Οι εφαρμογές ταξινομούνται με βάση αυτά τα δεδομένα, ως memory-bound, cpu-bound, DRAM Bw bound, PMM read and write bandwidth bound. Αποδεικνύεται ότι τα μοντέλα μπορούν να επιτύχουν ακρίβεια προβλέψεων πάνω από 90%. Μάλιστα, το μοντέλο SVM μπορεί να επιτύχει τις καλύτερες βαθμολογίες ακρίβειας στον ελάχιστο μέσο χρόνο εκπαίδευσης. Με βάση αυτή την ταξινόμηση, θα μπορούσαν να δοθούν συγκεκριμένες κατευθυντήριες γραμμές για τη δυναμική τοποθέτηση των εφαρμογών στο Ετερογενές Σύστημα Μνήμης.

1.7.2 Μελλοντικές επεκτάσεις

Αυτή η εργασία θα μπορούσε να βελτιστοποιηθεί περαιτέρω με την ενσωμάτωση των εκπαιδευμένων μοντέλων ML στην πολιτική λήψης αποφάσεων για την τελική τοποθέτησή αυτών των εφαρμογών στο σύστημα μνημών κατά την εκτέλεσή τους, σε DRAM και σε Optane DCPM. Επιπλέον, μια πιθανή βελτίωση θα ήταν η εκπαίδευση ενός προηγμένου νευρωνικού δικτύου ή ενός LLM για την πρόβλεψη της απόδοσης των εφαρμογών και τη λήψη αυτόματα αποφάσεων για την τελική τοποθέτηση. Όσον αφορά την πειραματική ανάλυση, το σύνολο των εφαρμογών θα μπορούσε να αξιολογηθεί και από την κατανάλωση ενέργειας. Επίσης, η απόδοσή τους θα μπορούσε να ελεγχθεί για τα διαφορετικά χαρακτηριστικά βάσεων δεδομένων, όπως το σχήμα των πινάκων, ο αριθμός των χαρακτηριστικών, ο βασικός κατακερματισμός των κλειδιών κ.λπ. Τέλος, θα μπορούσαν να εξεταστεί και η συμπεριφορά επιπλέον εφαρμογών με μεγάλες απαιτήσεις μνήμης, όπως οι εφαρμογές μηχανικής μάθησης.

Chapter 2

Introduction

Over the recent years, the growth of data applications have caused great demands of big storage and need for process at high pace. Several domains of applications based on databases, IoT networks, cloud services or Machine learning algorithms often rely on in-memory and in-place data processing over complex memory hierarchies to avoid expensive data transfers.[5]In this way, severe pressure is exerted on the main memory system.

At the same time, DRAM technologies have reached their physical limits in terms of scalability and sustainability [6]. Increasing DRAM Dimms in a memory system to achieve greater performance and higher capacity is becoming inefficient, due to increased energy consumption attributed to leakage and refresh power of the DRAM technology.[5] Main memory subsystem consumes up to 50% of the energy of a typical HPC system [7]. Given that information, new technologies of Non Volatile or Persistent Memories have gained interest. Technologies such as 3D-XPoint [8], Resistive RAM (ReRAM) [9], and Spin Transfer Torque Magnetic RAM (STTMRAM) [10] serve as examples of NVMs. These technologies typically offer data persistence, reduced energy consumption, and higher density than DRAM, resulting in increased memory capacities at lower costs [11]. In this work, we investigate the operation of the Intel Optane DPCM, which is the only commercial NVM up to now and it is based on 3D-XPoint technology.

However, Non Volatile memories technologies poses several disadvantages into the memory system. They introduce higher access latency, lower bandwidth, and limited write endurance. At the same time, they present asymmetrical latency and energy consumption between read and write operations. Replacing DRAM directly with an NVM reduces performance. So,NVM DIMMs are typically paired with DRAM modules, consisting a heterogeneous DRAM/NVM memory system. These systems aim to expand memory capacity sustainably and cost-effectively, offering users a diverse range of memory technologies.

The contribution of this work is based on the deployment and the profiling analysis of database workloads that execute in a heterogeneous DRAM/NVM memory system and on their classification and characterization in order to manage their placement in this memory system. More specifically, we focus on the profiling analysis and characterization of database workloads, due to the fact that they have to handle large volumes of data and consequently they have increased memory requirements. Therefore, it is interesting to examine their behavior and their performance using multiple cores in a Heterogeneous memory system. The queries that each workload pose into the system are executed either using DRAM or Optane and then their performance is compared. Furthermore, an innovative approach of this work is that all the workloads are tested not only in an isolated environment, but also in an interference-aware system, where multiple applications run simultaneously and the system's resources may be occupied.This simulation is achieved by concurrently running some specific benchmarks that

stress the system’s memory and the processor. As a final step, we train different machine learning models to predict if the workloads are memory-bound, bandwidth-bound or cpu-bound based on input data from the profiling analysis. The performance and the accuracy of these models are compared and guidelines are established for the placement of the workloads in the heterogeneous system.

The rest of this paper is organized as follows. Chapter 3 presents an overview of the related work and Chapter 4 illustrates the fundamental background. In Chapter 5 the profiling analysis and characterization are described in detail. Chapter 6 shows the proposed methodology to classify the workloads and Chapter 7 presents the experimental evaluation of the models. Finally, Chapter 8 concludes this work and proposes future work.

Chapter 3

Related Work

In an heterogeneous memory system, deciding which data or objects to host in each memory subsystem, in DRAM or in PMEM, is not trivial and poses notable performance implications. Hardware based and software based methods have been proposed through years. Hardware-based mechanisms are usually efficient, but they lack flexibility and sometimes yield to inconsistent performance. Software-based approaches pose management overheads, but in the same time they require knowledge and intrusive code changes. Some of the methods that have been proposed for placement in an heterogeneous memory system, are presented in this section.

Kannan et al. introduce KLOCs, which are logical groupings that capture the kernel objects associated with OS entities requested by applications. It is an OS abstraction which permits fluid tiering of kernel objects and fast allocation of kernel objects while supporting relocatability. KLOCs migrate related cold kernel objects en masse and the knodes are further associated with a variable for tracking their hotness (age) and whether they are active (inuse). MOCA is another method for object classification and allocation in an Heterogeneous memory system, proposed by Narayan [12]. It includes a profiler that names, profiles, and classifies memory objects within an application based on their memory access behavior as well as an allocation scheme at the OS level. MOCA focus particularly on heap objects as these often substantially contribute to the performance or energy bottlenecks of a memory-intensive application. The classification is stored as part of the application binary. OS is responsible for allocating pages from a particular virtual memory heap partition to physical pages in memory device with the desired characteristics.

Kammerdiener et al. [13] introduces a software modular user-level framework that collects and analyzes object-level information to guide memory tiering. Employs a custom memory allocator, known as bkmalloc, and a system-wide profiling and management tool called the Memory Auto Tiering (MAT) Daemon. bkmalloc allocates new objects into page-aligned regions of virtual memory and MAT Daemon profiles the memory usage of each known data object and then uses this information to create and enforce tier recommendations across the platform. Other works propose object allocation scheme which performs on the fly object admission and eviction, to and from the High Bandwidth Memory (HBM), based on software and hardware related attributes, which are incorporated into a cost model. Particular in [14] the cost model estimates the benefit of having an object in one memory over the other and then the Knapsack algorithm is applied. As an output, the algorithm lists the objects that are the best candidates to reside on HBM for a particular phase. The proposed method, presented in [15], is based on the idea of an Heterogeneous Memory Advisor, which aims to reduce the CPU stall cycles due to memory accesses by placing the most accessed objects in the memory subsystem with highest bandwidth. The base algorithm of the Advisor is based on a greedy relaxation of the 0/1 multiple knapsack problem. The workflow starts by an offline profiling a production-ready binary to generate a trace-file and extract the performance metrics around the data-objects created during the

application execution. The Advisor considers memory access patterns, bandwidth requirements, and object lifetimes to optimize the placement of objects in DRAM or PMEM.

Some other works, such as [16] [5] focus on placing pages over heterogeneous DRAM/NVM memory systems. More specifically, authors in [16] provide a run-time decision making scheduler for efficiently placing pages, in order to optimize performance and energy consumption. The main objective is to maximize the number of application's memory requests that are served from DRAM, thus to maximize performance, while reducing the energy consumption by placing data on low-power NVM when they are not frequently processed. The proposed scheduler consists of three major components: the Critical Page Selector, the Adjacent LSTM-based Predictor and the History based Predictor. Finally, the work that is presented in [5] is based on three mainly steps. First, it consists of the optimization of data organization, which leverages design space exploration for performing source-to-source data-flow transformations at application-level that improve the data organization via pareto optimal solutions. Next step is the memory object analysis and profiling, which is based on memory trace and profiling tools and analyzes the pareto optimal solutions. The last step is the placement on the heterogeneous memory system. The output of the methodology is a set of Pareto optimal solutions. Each point of the Pareto curve corresponds to a version of the application with a specific data organization, placed on a specific DRAM/NVM system.

In this work, we propose a novel software and high-level approach, that is efficient for indexes and data placement among DRAM and Optane persistent memory. We focus on the profiling analysis of applications, that are based on a NewSQL relational database, CockroachDB. These applications process big amount of data and have great memory and cpu demands. Furthermore, we characterise these apps in a realistic system environment, where applications and other commands run in parallel, by introducing noise into the system via benchmarks that span the system resources. The last step is to predict the performance of the applications by training a ML model. Our approach is an user-friendly approach without requiring changes in the code applications or in the OS. We propose a model that predict the performance of applications and classify them into memory-bound, cpu-bound or bandwidth-bound. Given that predictions, we manage to pose some guidelines in order to determine their placement and achieve better performance.

Chapter 4

Background

4.1 Heterogeneous memory systems

In the field of computer science, traditionally, memory systems consists of main memories based on DRAM technologies. DRAM memories store data in arrays, where each element is a one bit-memory cell. Each memory cell consists of a pair of a transistor and a capacitor, which is periodically refreshed to retain information. In modern technologies, memory data is organized into multiple banks hierarchies, where each bank is an array of rows and columns and can be accessed in parallel through buses. The dynamic nature of DRAM memories relies on periodic refresh of the capacitors to maintain the information within them.[17] These memories do not retain their data without power supply, meaning they are volatile memories. Memories based on DRAM technologies have high speed, low latencies and they offer significant capacity. However, their endurance and their power performance are low.

Throughout the years, as application demands grew, DRAM memory technology scaled accordingly. However, the reduction in accesses latency did not keep pace with the increase in their capacity. This has significant impact on application execution time, performance and their energy consumption during usage. DRAM technology is among the biggest energy consumers of high-performance computing (HPC) systems, posing over 25% of the system energy consumption. In addition, its evolution shows a flattening of the power consumption curve in the last years, being a factor of 1.5 per generation from 2000 to 2010, while being only a factor of 1.2 from 2010 to 2020.[33] DRAM memories have reached their physical limits in terms of scalability, sustainability and lifespan.[5]

At the same time, the amount of application data is growing rapidly, particular in applications such as databases, IoT applications, which require real time processing, and in machine learning algorithms for training complex models. The integration of multiple DRAM DIMMs to enable higher performance and satisfy increased memory capacity requirements is becoming inefficient, due to increased energy consumption attributed to leakage and refresh power of the DRAM technology. Therefore, the maintainability cost of state-of-the-art super-computing systems makes it highly unattractive to invest in memory capacity increase, entirely based on DRAM. Thus, the development of other memory technologies has begun, including non-volatile memory technologies, such as Intel Optane DPCM (Data Center Persistent Memory)[18].

Intel Optane DPCM memory is a non-volatile memory based on the 3D XPoint architecture. Each memory cell can be accessed individually within a memory grid in a three-dimensional array [19]. In this way, the density, the enhance performance are improved and durability is provided. Compared to existing SSD technologies (including Optane SSDs), which connect via external interfaces such as PCIe, Optane DC PMM delivers superior performance and utilizes a memory interface capable of

byte-level access. Compared to DRAM, Optane offers higher density and durability.[?] A single CPU socket can support up to six Optane DC PMMs, achieving a maximum capacity of 6 TB of Optane DC memory. It is connected directly to the memory bus and interfaces with the integrates memory controller (iMC) on the CPU, similar to a traditional DRAM. These support up to 2666 MT/s.[19] To ensure data persistence, Optane DC PMM integrates with Intel’s asynchronous DRAM refresh (ADR) feature, which guarantees that CPU stores reaching the ADR domain will survive a power failure and be flushed to the NVDIMM within a hold-up time of less than 100 μ s. Optane DCPM uses a 72-bit data bus, similar to DDR4, and transfers data in 64-byte cache-line granularity.[19]

Optane memory can operate in two distinct modes, Memory mode and App-Direct Mode, which must be selected based on the specific use case and workload requirements. In Memory mode, DRAM becomes a direct-mapped write-back cache to PMM and can only cache accesses to NVDIMMs attached to the same iMC [19]. The PMem is exposed as the main system memory to the operating system. One impact of this cache mechanism is that DRAM on one socket cannot cache accesses to PMM on the other socket, which contributes to NUMA effects in Memory mode [1]. This design is likely a trade-off between flexibility and performance to avoid routing requests among iMCs. As a write-back cache, writes are automatically buffered in DRAM, which is critical for avoiding performance degradation due to low write bandwidth to PMM. [34] However, when DRAM is used as cached memory for Optane it is common suffering from numerous conflict misses, especially for workloads with non-sequential access patterns. In the app direct mode, the DRAM is used as the system volatile memory, while the Optane PMM modules are treated as byte-addressable persistent memory.

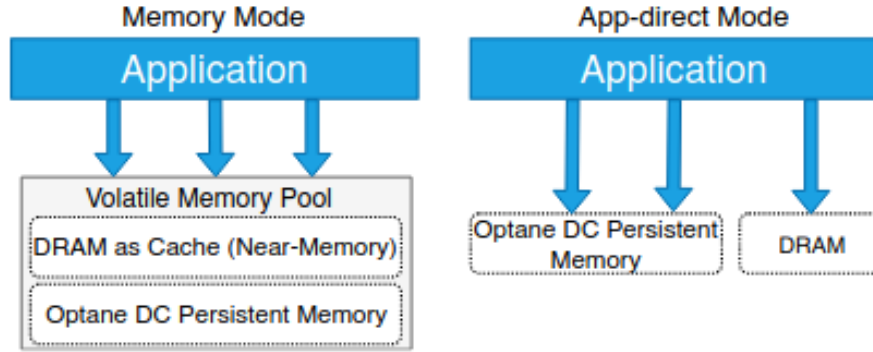


Figure 4.1.1: Operating modes of Optane DCPMM.[1]

In this work, Optane is used in app direct mode to have the flexibility to manage the placement of the applications and monitor the accesses and the memory throughput of both memories. We have the ability to decide which data is stored in each memory. Indices can be stored in persistent memory to avoid rebuilding them on reboot, achieving a significant reduction in restart time. Also, we avoid the performance degradation due to the conflict misses, when DRAM is used as a cache memory to PMEM.

Compared to DDR4 DRAM, PMEM latencies increase 2x–6x for reads and 6x–30x for writes depending on the access pattern, with a read latency of up to 374.1 ns and a write latency of 391.2 ns and memory accesses involving write-backs in DCPMM show a latency of 458.4 ns, which is 16.1% higher than read-only accesses. Whereas bandwidth in Optane DCPM decreases around 7% for reads and 40% for writes.[20] NVMM has significantly lower write endurance (10^8 cycles) compared to DRAM’s 10^{16} cycles, making NVMM unsuitable as a direct DRAM replacement. However, in terms of power efficiency, Optane DC PMMs consume less power than DRAM when idle since they do not require constant refreshing. When it comes to value, a second point can be made: persistent memory modules

are more affordable than DRAM. So, Optane DCPM can be chosen to build servers with less DRAM but more Optane DC persistent memory to maintain a reasonable, or larger, memory footprint for workloads that might not require as much of the nanosecond latency that DRAM offers.

Despite the undeniable benefits on NVM technologies, their limitations present significant challenges for developers. NVMs exhibit increased access latency and lower bandwidth compared to DRAM technologies. Also, they display asymmetry in latency and energy consumption during read and write operations. The direct replacement of DRAM with NVMs, such as Intel Optane DCPM, significantly reduces performance, resulted from phenomena such as write throttling and access contention. That is why, NVM DIMMs are typically combined with DRAM modules to form a heterogeneous DRAM/NVM system. This approach combines the advantages of both technologies, utilizing DRAM for high-speed and high-frequency operations and NVM for higher capacity and persistence.

4.2 Relational Databases

Relational databases have been a common used technology for many decades for storing and managing data, employing SQL to combine and summarize data[?]. A relational database organizes data into rows and columns, which collectively form a table where the data is interconnected. The data is structures across multiple tables, which can be linked through a primary key or a foreign key, which define the various relationships between tables. Relational databases are also associated with transactional databases, which execute commands or transactions as cohesive units. Transactions are characterized by specific properties known by the acronym ACID (Atomicity,Consistency,Isolation,Durability). These ensure that each transaction is executed as an independent and isolated operation, maintaining data consistency and ensuring that changes made to the data are preserved.

The emergence of new technologies, such as World Wide Web and cloud computing, created the need to manage more complex databases capable of supporting scalability and high performance. NewSQL technology is the latest evolution, combining the scalability and availability offered by Not Only SQL databases while maintaining the ACID properties that characterize relational databases[21]. In this work, CockroachDB database is used, which is an example of a NewSQL relational database.

CockroachDB [22] is a SQL-based database that is source-available and enables simple and efficient data management. It supports horizontal scaling, distributing workloads across multiple machines or across multiple nodes in a cluster, without the need for specialized hardware. In addition, CockroachDB employs a distributed consensus algorithm, based on Raft, to replicate data across nodes, ensuring durability and fault tolerance. It is user-friendly database using a SQL interface compatible with PostgreSQL, while also leveraging its distributed nature to enhance scalability and performance [23].

In this work, we investigate the memory profile of workloads based on CockroachDB. More specifically, these workloads are bank,kv,tpcc and ycsb. Bank models a set of accounts with currency balances. Kv executes reads and writes to keys spread (by default, uniformly at random) across the cluster. Tpcc simulates a transaction processing workload using a rich schema of multiple tables and Ycsb simulates a high-scale key value workload, either read-heavy, write-heavy, or scan-based, with additional customization.

4.3 Supervised Machine Learning Algorithms

Machine learning represents a large field of creating algorithms that allow the computer to learn and predict via a training process of finding statistical regularities or other patterns of data. These algorithms are created to be able to represent the human approach of learning some task[24]. They can be discriminated into multiple categories of algorithms based on their learning method or the problem that are able to solve. The two most commonly used subcategories are the supervised and

the unsupervised algorithms. Supervised learning is the task of learning a function that maps an input to an output based on example input-output pairs. Unlike, supervised methods, the unsupervised learning algorithms learn few features from the data and it uses the previously learned features to recognize the class of the data. It is mainly used for clustering and feature reduction.

In this work, we try to classify and characterise the applications by leveraging the data obtained from the profiling analysis on the applications. Performance modeling using empirical evaluation can be done using supervised machine learning algorithms[25]. In supervised learning methods, a labelled dataset is used to make algorithm learn the function $f(x)$ which maps inputs X_i to an output Y_i [25]. The input dataset is divided into train and test dataset. The train dataset has output variable which needs to be predicted or classified. All algorithms learn some kind of patterns from the training dataset and apply them to the test dataset for prediction or classification[2]. In classification problems, the classes are predetermined. These classes are created in a manner of finite set, defined by the human, which in practice means that a certain segment of data will be labeled with these classifications [24].

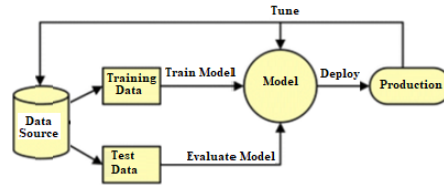


Figure 4.3.1: Supervised machine learning method.[2]

Our purpose is to classify the running applications based on their performance and their memory behavior, into memory-bound, cpu-bound or bandwidth-bound applications. In order to classify our applications, we choose to apply and compare four machine learning algorithms, that are based on different methods of learning and predicting the final output. These methods are presented below.

4.3.1 Random Forest

A random forest is a classifier consisting of a collection of tree-structured classifiers $h(x, \Theta_k)$, $k = 1, \dots$ where the Θ_k are independent identically distributed random vectors and each tree casts a unit vote for the most popular class at input x [26]. It takes different random parts of the dataset to train each tree and then it combines the results by averaging them. Each Decision Tree make predictions independently. Each node in a decision tree represents a feature in an instance to be classified, and each branch represents a value that the node can assume. Instances are classified starting at the root node and sorted based on their feature values[27]. This approach is called as model ensemble learning and helps improve the accuracy of predictions [26]. The accuracy of a random forest depends on the strength of the individual tree classifiers and a measure of the dependence between them[35]. Decision trees usually employ post-pruning techniques that evaluate the performance of decision trees, as they are pruned by using a validation set[27]. Random Forest are used for classification and for regression problems.

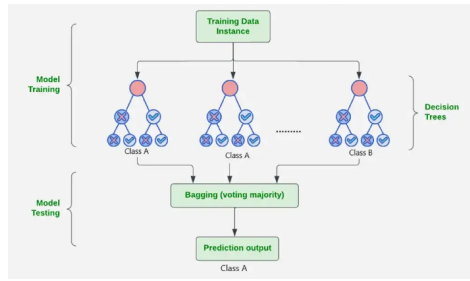


Figure 4.3.2: Random Forest Algorithm.[?]

4.3.2 Support Vector Machine

SVM is a discriminant technique that aims at finding, based on an independent and identically distributed training dataset, a discriminant function that can correctly predict labels for newly acquired instances. Unlike generative machine learning approaches, which require computations of conditional probability distributions, a discriminant classification function takes a data point x and assigns it to one of the different classes[36]. The key idea is to find the hyperplane, a decision boundary separating different classes in feature space, that best separates two classes by maximizing the space -margin between them. This margin is the distance from the hyperplane to the nearest data points (support vectors) on each side[37]. In the following image the hyperplanes for two classes classification problem are presented.

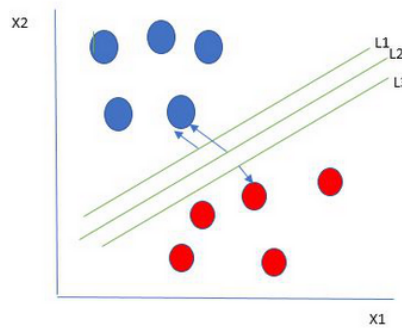


Figure 4.3.3: Random Forest Algorithm.[3]

For non-linear functions, the kernel trick is used to map X_i into a higher dimensional space called kernel space to make better predictions. Svm uses insensitive loss function and penalizes predictions that are far from the actual output [25].

4.3.3 Multi-Layer Perceptron

MLP is a sort of an artificial neural network and is consisted of a large number of interconnected neurons that are working in parallel to complete their task and classify the input data. It is organised in layers with a feed-forward information flow. The main architecture of an MLP network consists of signals that flow sequentially through the different layers from the input to the output layer. Between the input layer and the output layer are intermediate layers, that are also called hidden layers. Each unit is first used to calculate the difference between a vector of weights and the vector given by the outputs of the previous layer. To generate the input for the next layer, a transfer function also called activation function is applied to the result [28]. The algorithm is executed , until all the predictions are correct in the training set. Each time the error signal between the output of the network and the

desired output is back-propagated to the network and then the adjustments that are needed, are made on the synaptic weights.

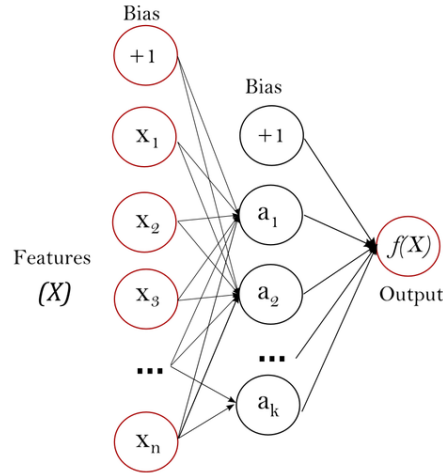


Figure 4.3.4: Mlp algorithm for one hidden layer.[4]

4.3.4 Logistic Regression

Logistic Regression is a discriminative classifier algorithm, which extracts some set of weighted features from the input, takes logs and combine them linearly. Each feature is multiplied by a weight and then added up[38]. So, the goal of logistic regression is to map a function from the features of the dataset to the targets to predict the probability that a new example belongs to one of the target classes. The objective is to find a set of weights such that the negative log likelihood is minimized over the defined training set using optimization techniques such as gradient descent or stochastic gradient descent [29]. Minimizing the negative log likelihood also means maximizing the probability the parameter estimate, π , of selecting the correct class. The loss function that measures the difference between the ground truth label and the predicted class label is referred to as the cross-entropy. If the prediction is very close to the ground truth label, the loss value will be low. Alternatively, if the prediction is far from the true label, the resulting log loss will be higher. [29] The cost function for logistic regression is given as:

$$J(\theta) = -\frac{1}{m} \sum [p_i \log(y_i) + (1 - p_i) \log(1 - y_i)]$$

Chapter 5

Profiling Analysis and Characterization

The first step, in order to determine the placement of the databases applications in the heterogeneous memory system, is the profiling analysis and the characterization of the workloads, that are based on CockroachDB[22], as far as it concerns their memory behavior and their performance. These workloads are the following:

- **Bank**: models a set of accounts with currency balances.
- **Kv**: executes reads and writes to keys spread (by default, uniformly at random) across the cluster.
- **TPCC**: simulates a transaction processing workload using a rich schema of multiple tables.
- **YCSB**: simulates a high-scale key value workload, either read-heavy, write-heavy, or scan-based, with additional customization. For Ycsb workload, we distinguish 4 types of the same workload. Type A is update heavy(50% reads and 50% writes), type B is read mostly version, type C is read only and type F is read-modify-write.

More specifically, we examine the memory throughput (DRAM reads, DRAM writes, PMEM reads, PMEM writes) of the above workloads, the read and write accesses in persistent memory and finally the query latency that they incorporate into the system. The basic tools that are used for this purpose, are Intel® Performance Counter Monitor (Intel® PCM)[30] and ipmctl tool [31]. The experiments were conducted on a server integrating a 2x20 core Intel Xeon Gold 5218R CPU @2.10GHz with 2 sockets and totally 80 CPUs, with 4x32GB DDR4 DIMMs and 6x256GB Optane DC DIMMs. The virtualized system is consisted by 3 numa nodes, 2 nodes connected to DRAM and 1 node connected to Optane DCPM. Intel Optane DCPM was configured in App-Direct mode with the EXT4-DAX file-system. Version 1.11 of the Persistent Memory Development Kit (PMDK) and gcc-13.3 were employed for the deployment of the applications.

5.1 Experiments in Isolated System

In this section, we execute the four workloads independently two times, one using DRAM and the other using Optane DCPM. The two different executions are implemented through the use of the numactl command `-membind`. We examine their memory behavior and their performance in an isolated environment, while the server is not occupied with other procedures. Every workload is executed for 5 minutes for each different concurrency value. We use 1,2,4,8,16,32,64 and 80 threads and we keep records about the memory throughput, the persistent memory accesses and the outputs of the workloads.

First, with the use of `ipmctl` command, we observe the memory accesses in the 6 Optane DC DIMMs. The results are presented below.

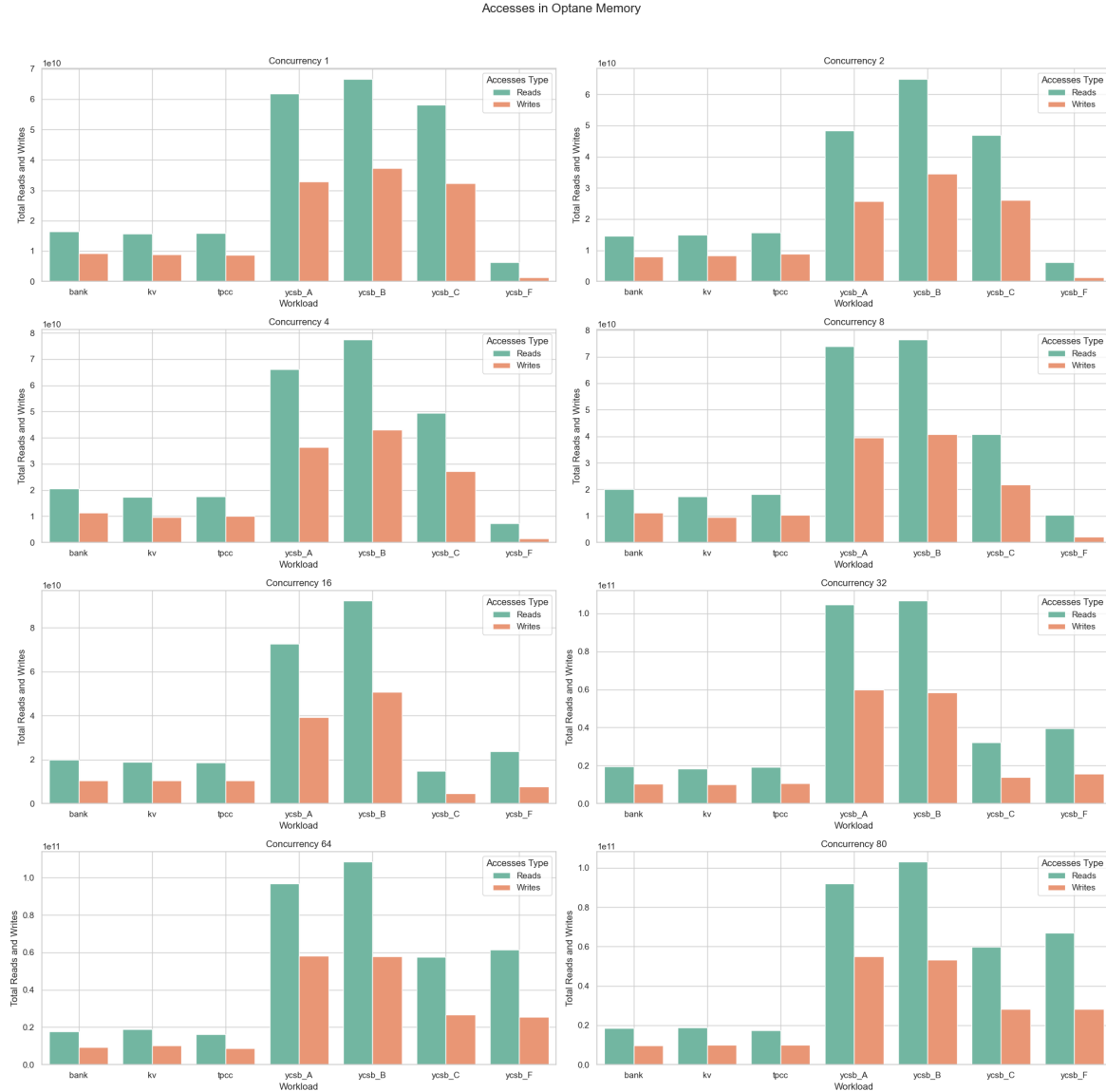


Figure 5.1.1: Persistent memory accesses

It can be noticed that persistent memory accesses for all workloads are between 10^{10} and 10^{11} and read accesses are always greater. As the number of threads increases, the memory accesses also increase, without observing abrupt increases. More specifically, workloads bank, kv, tpcc have a sustain behavior with small increases depending on the number of threads. They also record the least memory accesses. Ycsb type A and type B consistently record the highest values.

From the outputs of the workloads, we collect the latency's that the queries add to the system and plot them for each workload and each concurrency value in violin plots.

Latency (p99) in DRAM Memory

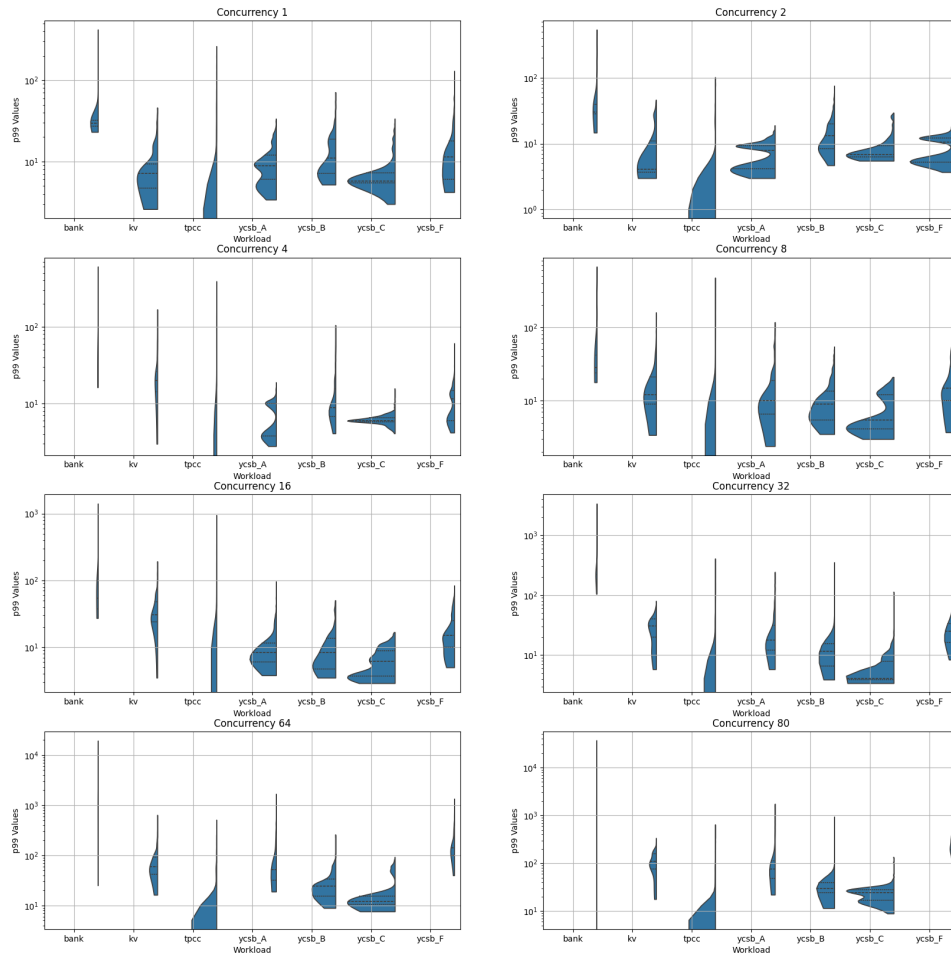


Figure 5.1.2: P99 in DRAM execution

Latency (p99) in Optane Memory

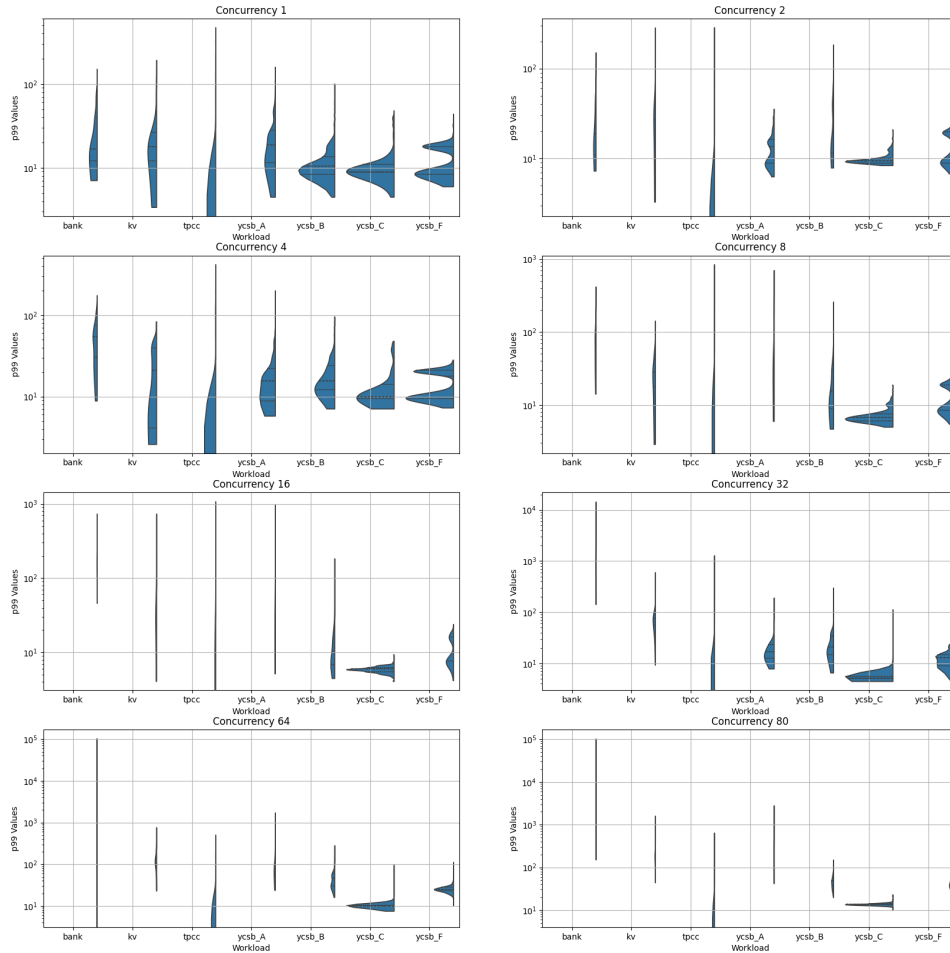


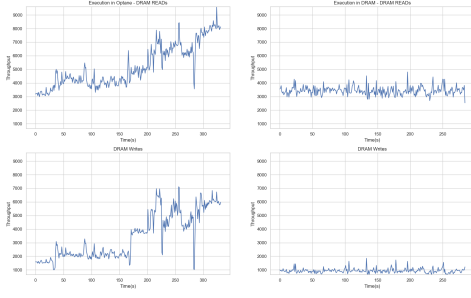
Figure 5.1.3: P99 in PMEM execution

We can observe that query latency has wide range and it is not clustered around a specific value for most workloads. The values are between 10 and 10^5 ms in DRAM and in Optane execution. Bank and Tpcck workloads introduce the greater latency into the system, and Ycsb introduces the smallest, especially type C, which is a read only version of the workload. As the number of threads increases, latency tends to increase and the range of the values becomes wider.

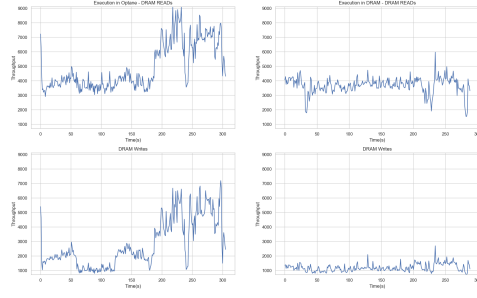
Finally, via PCM Memory tool, we investigate for each workload their memory throughput in DRAM and Optane DCPM for each concurrency value. The results for each workload are presented separately for DRAM Reads/Writes in both executions and PMM Reads/Writes in the following sections.

5.1.1 Bank

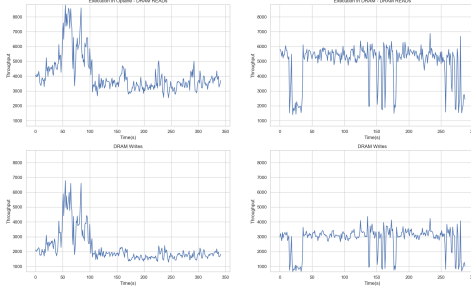
DRAM Throughput



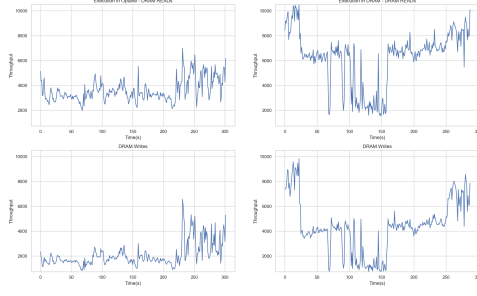
(a) Concurrency value: 1



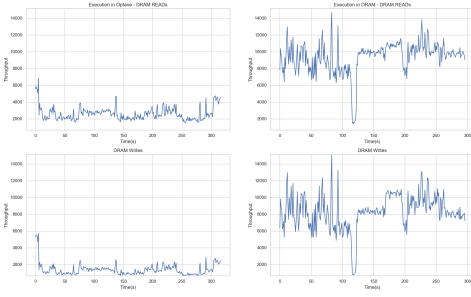
(b) Concurrency value: 2



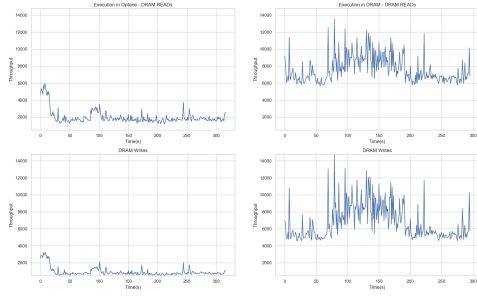
(c) Concurrency value: 4



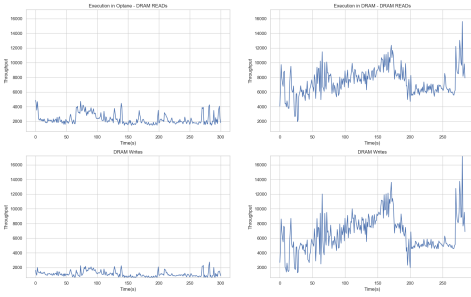
(d) Concurrency value: 8



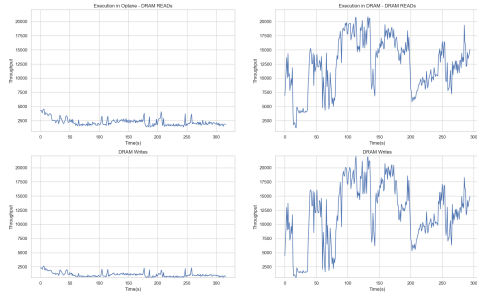
(e) Concurrency value: 16



(f) Concurrency value: 32



(g) Concurrency value: 64



(h) Concurrency value: 80

Figure 5.1.4: Memory DRAM Reads/Writes plots for different concurrency values - Bank Workload

We can observe that DRAM Reads are greater than DRAM writes for all the concurrency values in

both executions (DRAM and Optane). Up to concurrency value 8, DRAM throughput is higher in Optane execution. The values range from 3000 until 9000 for reads and from 1000 up to 7000 for writes. As the concurrency value increases, the throughput increases. However, it can not be noticed a specific pattern or a symmetry neither at reads nor at writes throughput. From concurrency value 8, it seems that DRAM execution reaches higher read and write throughput. The throughput increases as the value of concurrency increases and the throughput reaches up to 20000. At the same time, the throughput in Optane execution remains almost stable, after concurrency value 8.

PMM Throughput

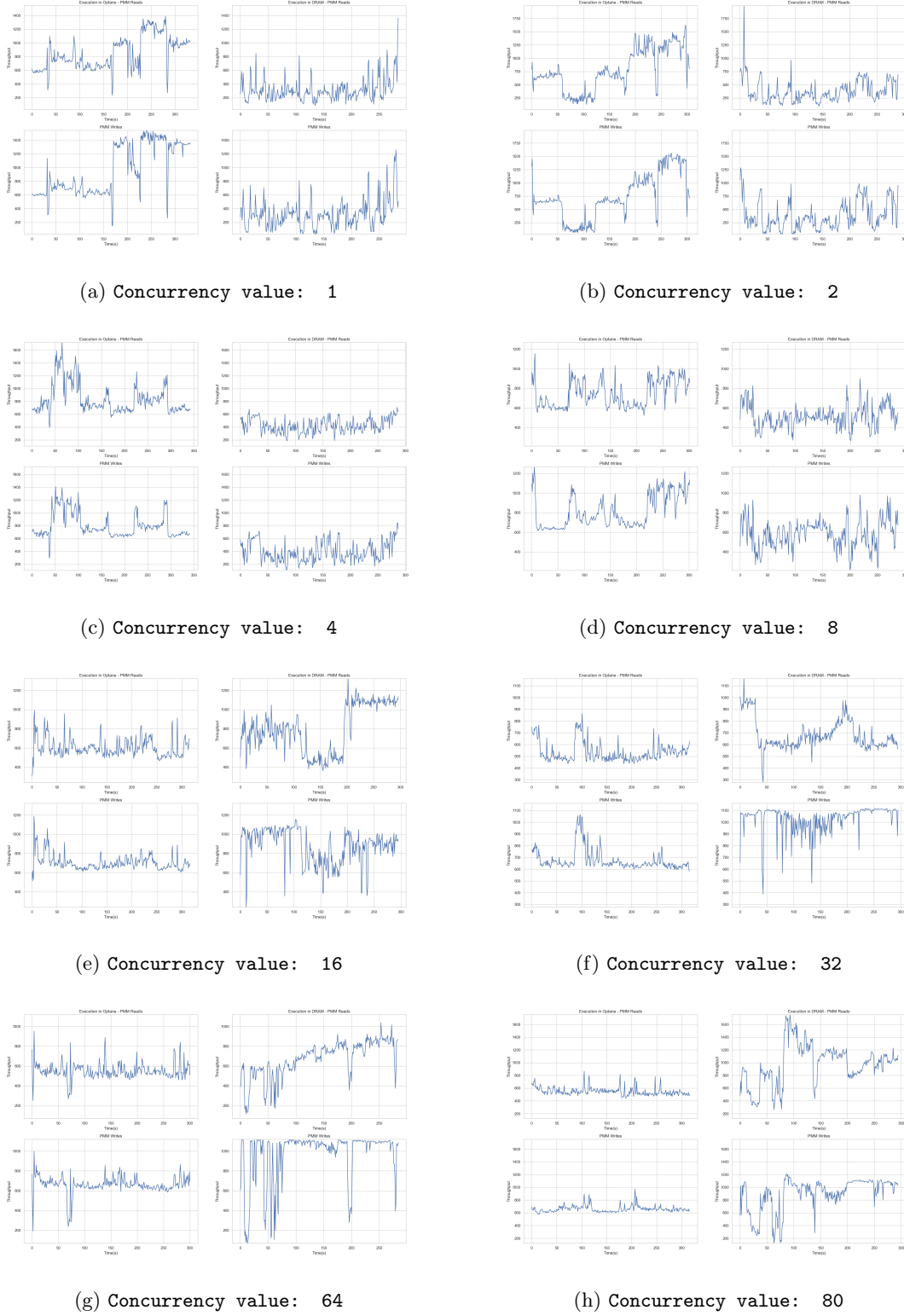
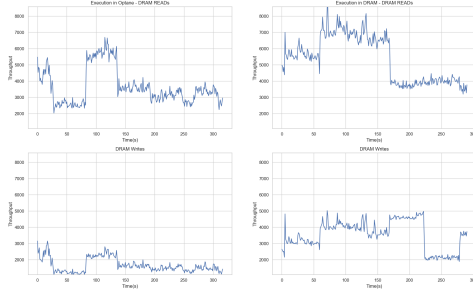


Figure 5.1.5: Memory Optane plots for different concurrency values - Bank Workload

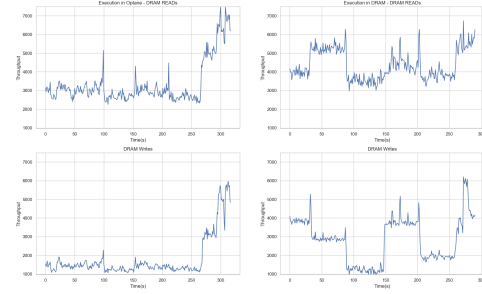
Persistent memory throughput is clearly lower than the DRAM throughput at both executions. The values range from 200 to 1700 for reads and from 200 to 1200 for writes. As the concurrency value increase the throughput increase. PMM Reads and PMM Writes are higher in Optane execution until the concurrency value equal to 8, then the throughput is similar in both executions and sometimes higher in DRAM execution.

5.1.2 Kv

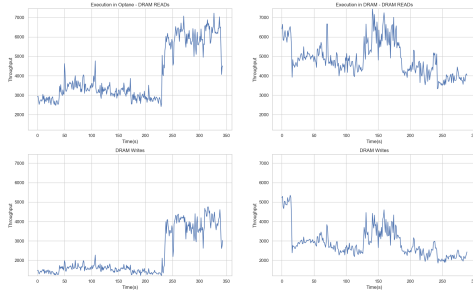
DRAM Throughput



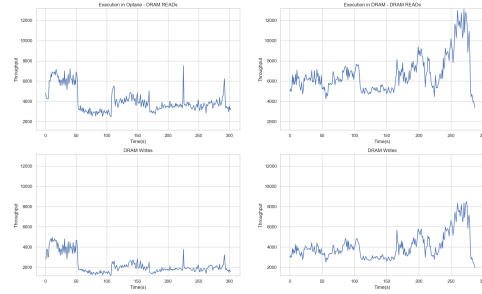
(a) Concurrency value: 1



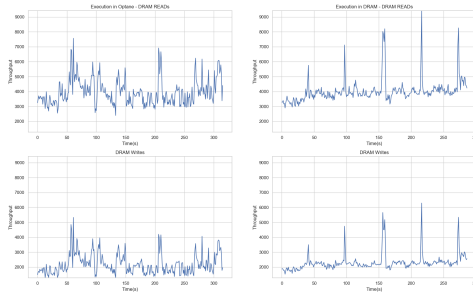
(b) Concurrency value: 2



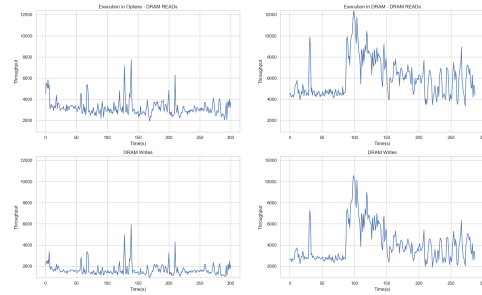
(c) Concurrency value: 4



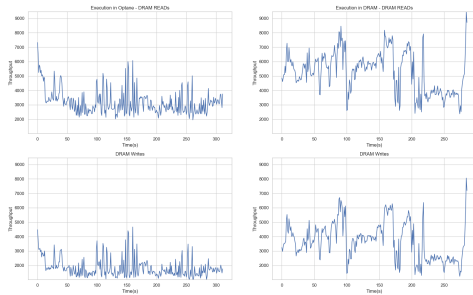
(d) Concurrency value: 8



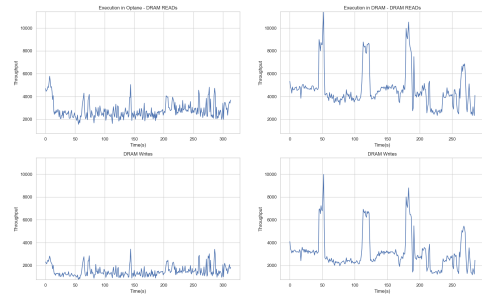
(e) Concurrency value: 16



(f) Concurrency value: 32



(g) Concurrency value: 64



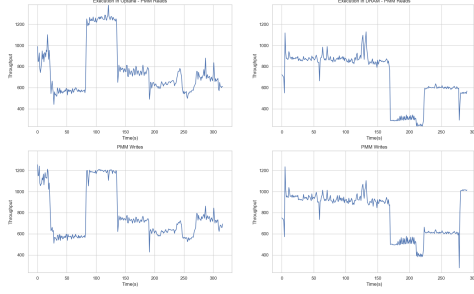
(h) Concurrency value: 80

Figure 5.1.6: DRAM throughput plots for different concurrency values - kv Workload

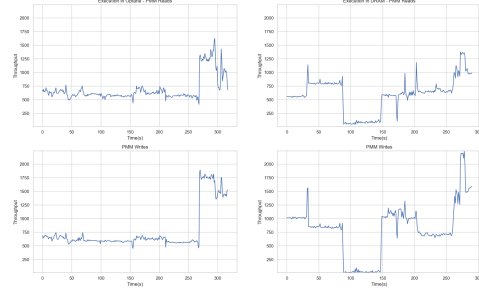
From the above diagrams, it is noticed that DRAM throughput is higher in DRAM execution for

all the concurrency values. DRAM reads are higher than DRAM writes with values that range from 4000 to 12000 for reads in DRAM and from 2000 to 7000 in Optane execution. DRAM writes are between 1000 and 5000 in Optane execution and between 2000 and 10000 in DRAM execution. As the concurrency value increases the memory read and write throughput increases in both executions.

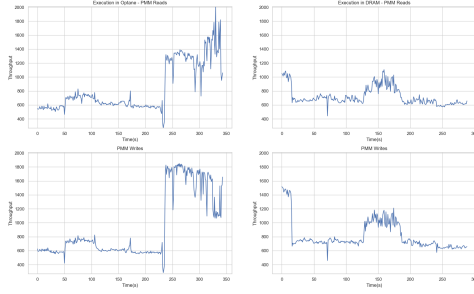
PMM Throughput



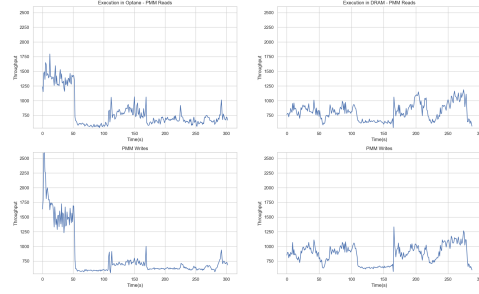
(a) Concurrency value: 1



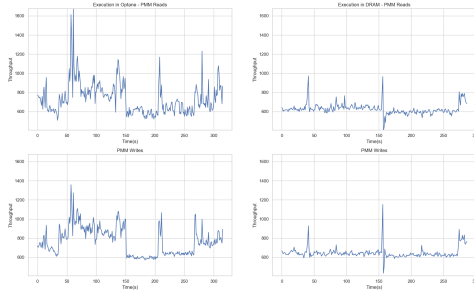
(b) Concurrency value: 2



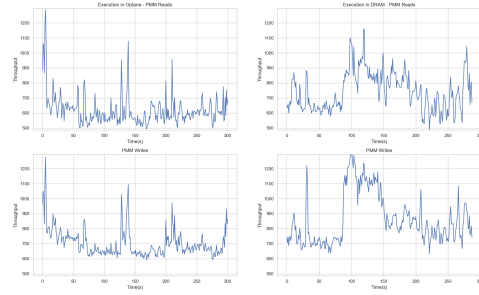
(c) Concurrency value: 4



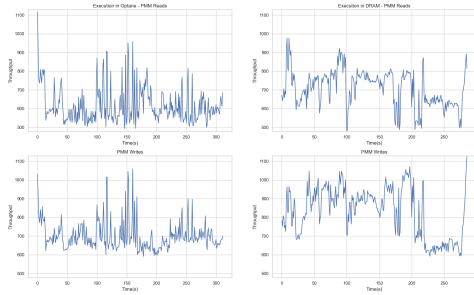
(d) Concurrency value: 8



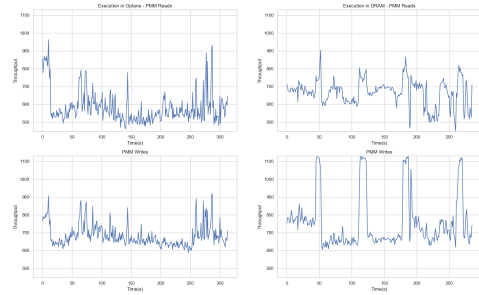
(e) Concurrency value: 16



(f) Concurrency value: 32



(g) Concurrency value: 64



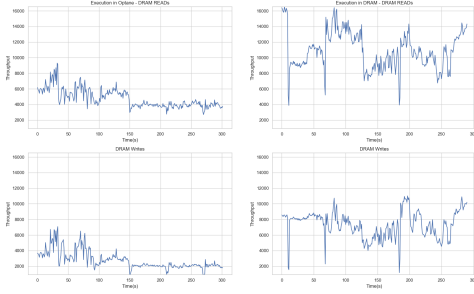
(h) Concurrency value: 80

Figure 5.1.7: PMM throughput plots for different concurrency values - kv Workload

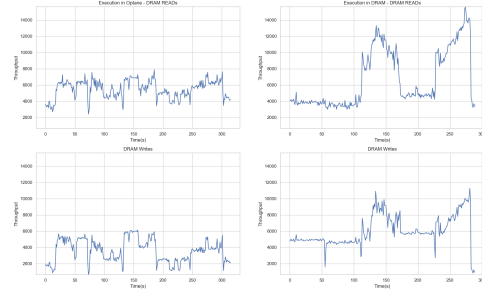
In the Optane execution, read and write memory throughput is lower. PMM reads and writes range from 600 to 1200. There is not great difference between memory reads and memory writes and memory writes are often proved to be more than memory reads. As the value of concurrency increases, the memory throughput tends to increases, without observing abrupt increases. However, the memory throughput remains low in all the different executions and parameters.

5.1.3 TPCC

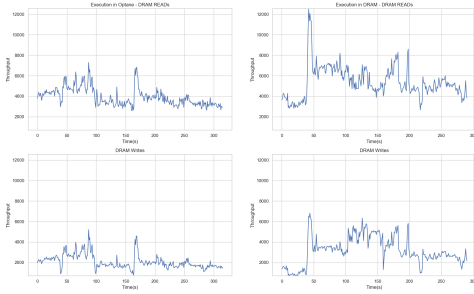
DRAM throughput



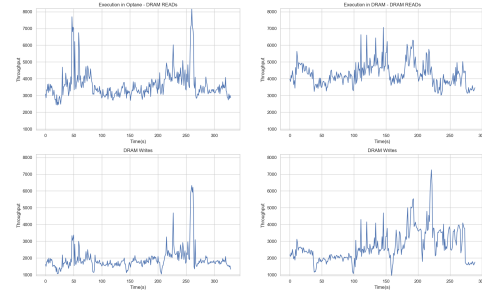
(a) Concurrency value: 1



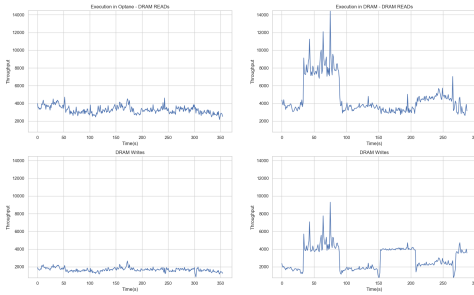
(b) Concurrency value: 2



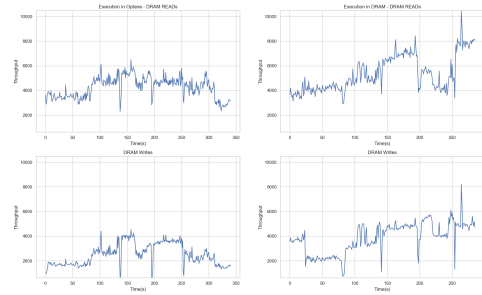
(c) Concurrency value: 4



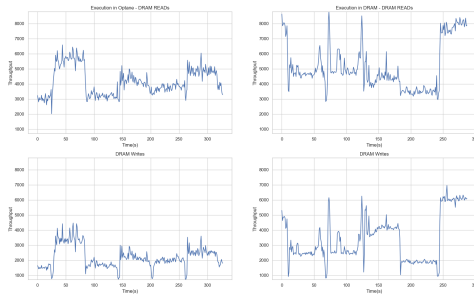
(d) Concurrency value: 8



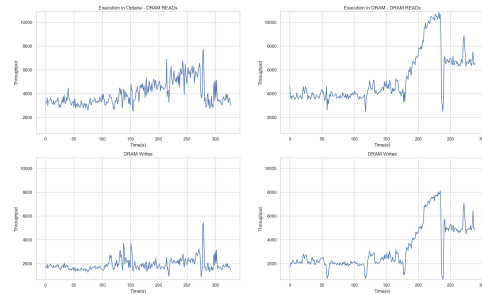
(e) Concurrency value: 16



(f) Concurrency value: 32



(g) Concurrency value: 64

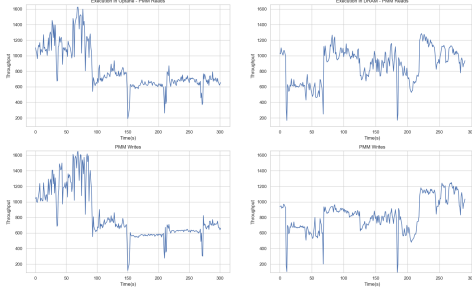


(h) Concurrency value: 80

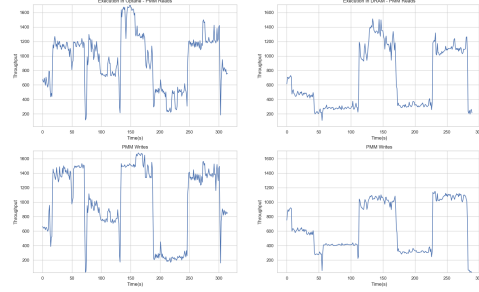
Figure 5.1.8: DRAM throughput plot for different concurrency values - tpcc Workload

In Tpcx workload, memory read throughput is higher than memory write throughput. Moreover, Dram throughput is higher in DRAM executions for all the concurrency values. We can observe that memory throughput is greater in the small values of concurrency. From concurrency value 4, the throughput decreases in both executions. For memory reads, the throughput range is 3000- 1200, with peak values between 14000-16000 in DRAM execution and 2000-8000 in Optane execution. As for memory writes, the values range from 1000 to 10000 in DRAM execution and from 1000 to 7000 in Optane execution.

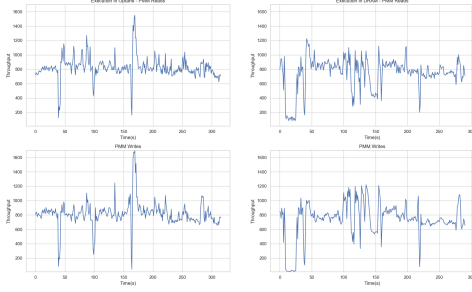
PMM Throughput



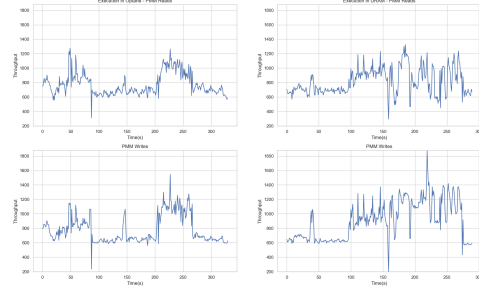
(a) Concurrency value: 1



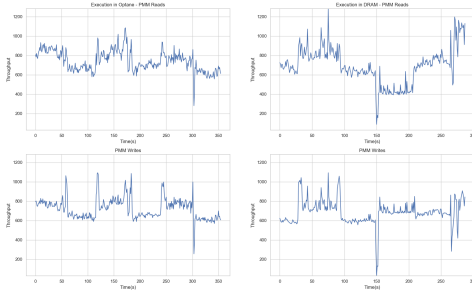
(b) Concurrency value: 2



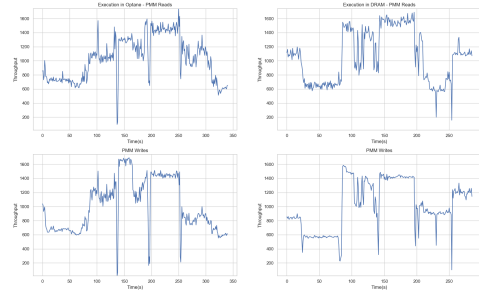
(c) Concurrency value: 4



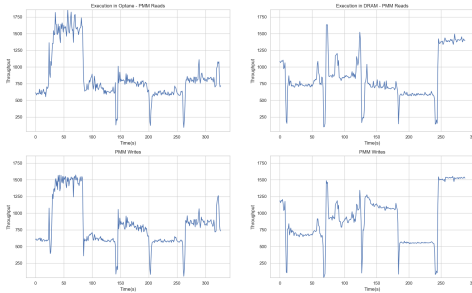
(d) Concurrency value: 8



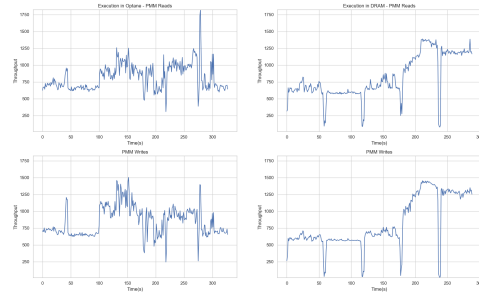
(e) Concurrency value: 16



(f) Concurrency value: 32



(g) Concurrency value: 64



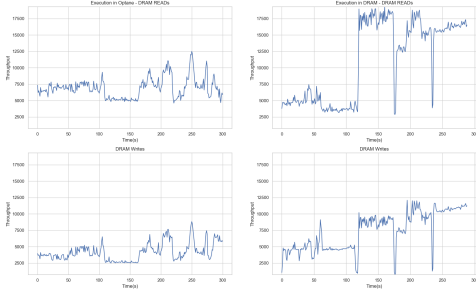
(h) Concurrency value: 80

Figure 5.1.9: PMM throughput plot for different concurrency values - tpcc Workload

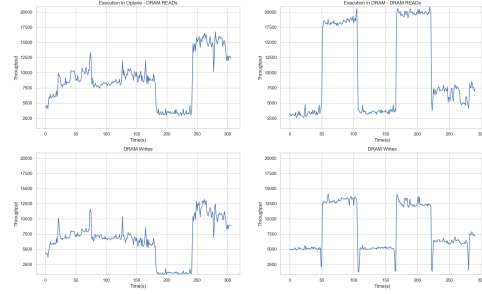
The PMM throughput is similar in DRAM execution compared to Optane execution, where it shows a small superiority, but it remains relatively low for all the concurrency values. In fact, the memory throughput increases as the number of threads increases, but for the concurrency values 4 and 8 a throughput decrease is observed. In DRAM execution, the values of memory reads are from 200 to 1500 and memory writes from 100 to 1400. In Optane, for reads the values range from 200 to 1750 and for writes range from 100 to 1600.

5.1.4 YCSB

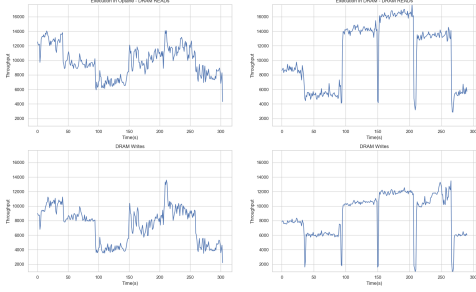
Type A: DRAM throughput



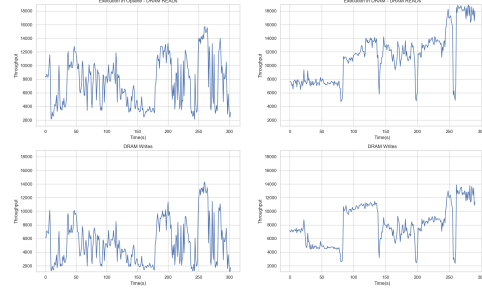
(a) Concurrency value: 1



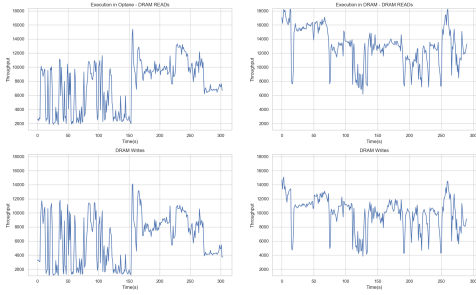
(b) Concurrency value: 2



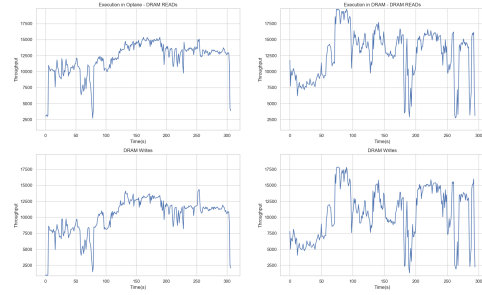
(c) Concurrency value: 4



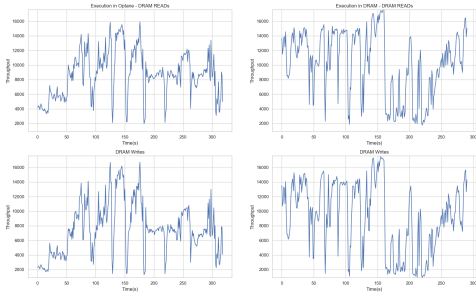
(d) Concurrency value: 8



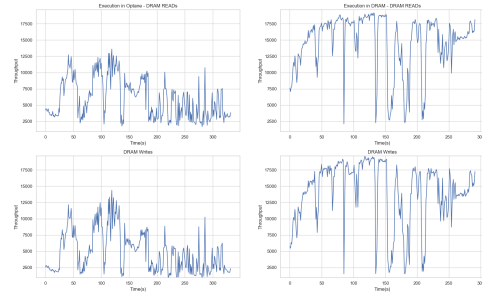
(e) Concurrency value: 16



(f) Concurrency value: 32



(g) Concurrency value: 64



(h) Concurrency value: 80

Figure 5.1.10: DRAM throughput plots for different concurrency values - ycsb A Workload

Type A: PMM Throughput

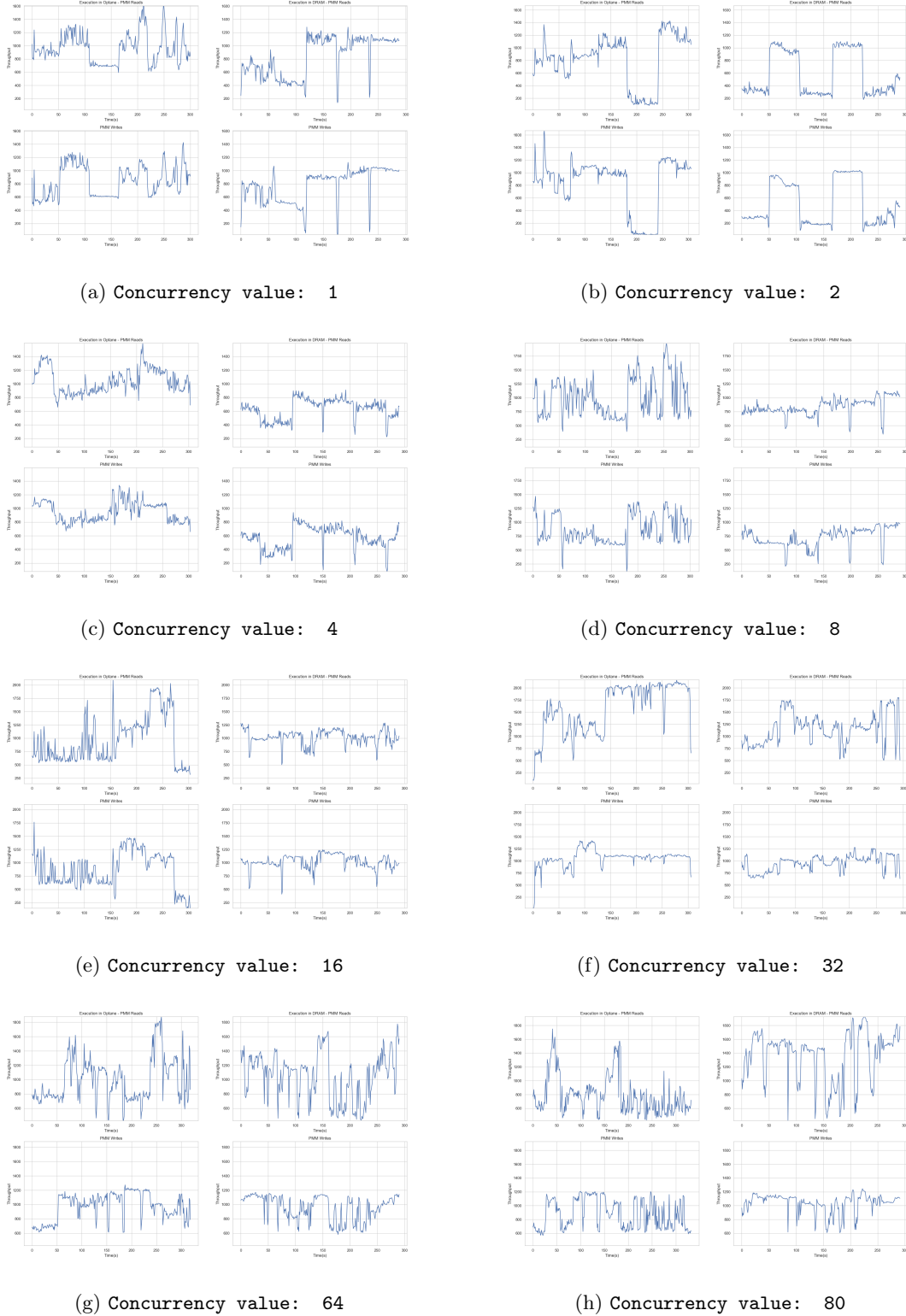
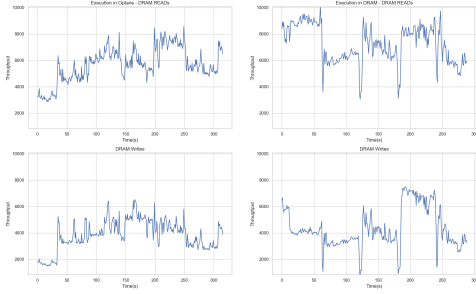
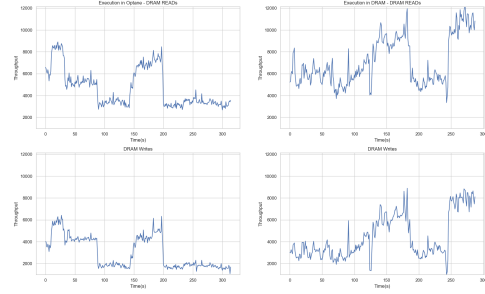


Figure 5.1.11: PMM throughput plots for different concurrency values - ycsb A Workload

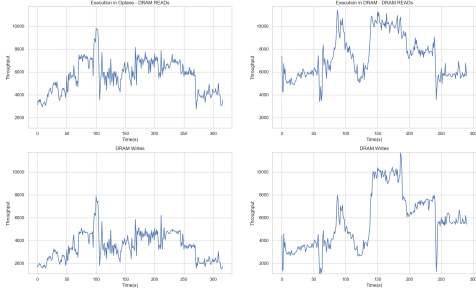
Type B: DRAM throughput



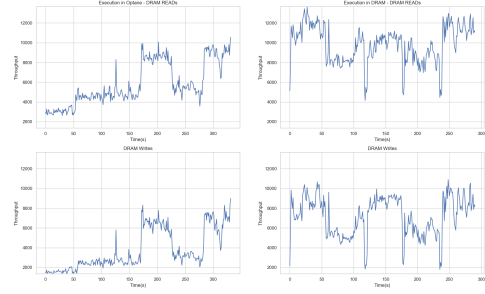
(a) Concurrency value: 1



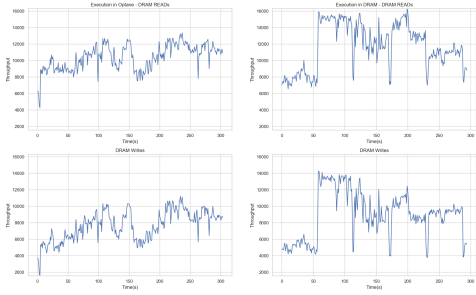
(b) Concurrency value: 2



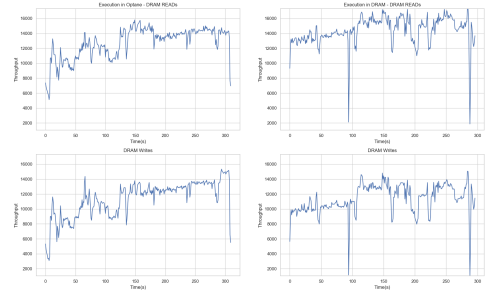
(c) Concurrency value: 4



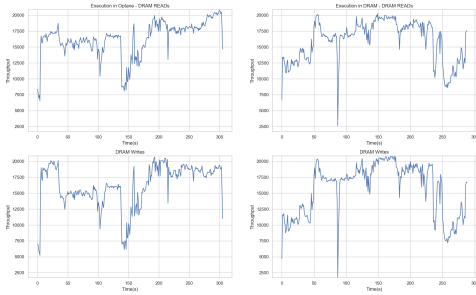
(d) Concurrency value: 8



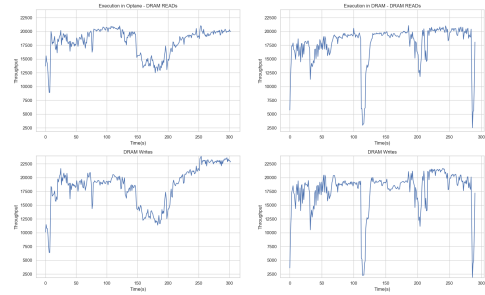
(e) Concurrency value: 16



(f) Concurrency value: 32



(g) Concurrency value: 64



(h) Concurrency value: 80

Figure 5.1.12: DRAM throughput plots for different concurrency values - ycsb B Workload

Type B: PMM Throughput

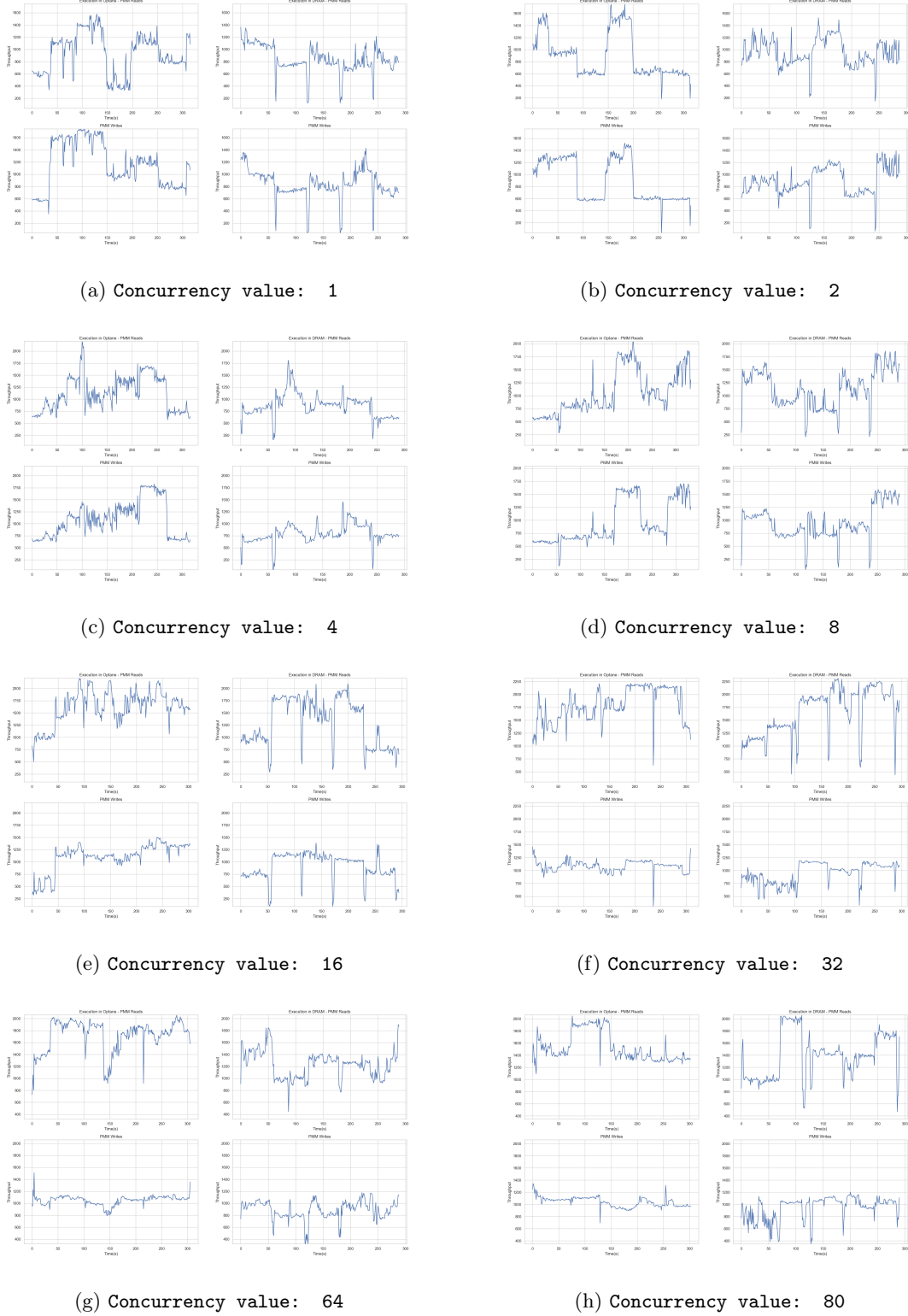
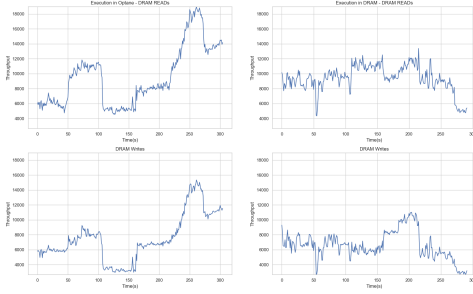
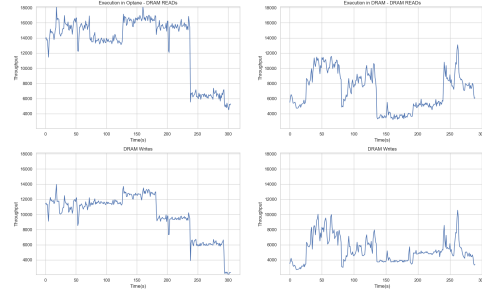


Figure 5.1.13: PMM throughput plots for different concurrency values - ycsb B Workload

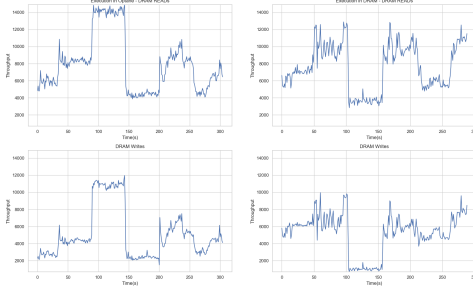
Type C: DRAM Throughput



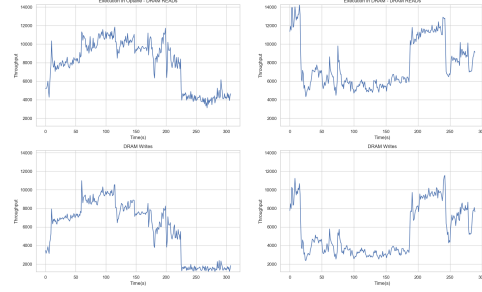
(a) Concurrency value: 1



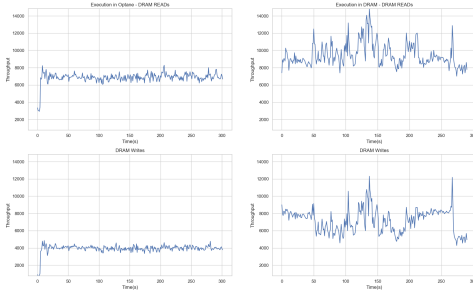
(b) Concurrency value: 2



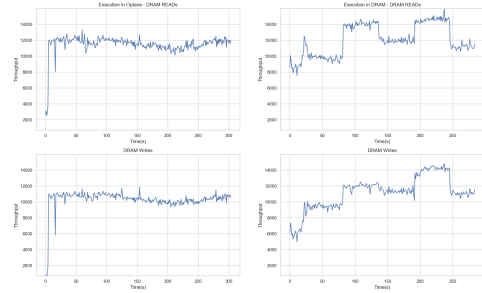
(c) Concurrency value: 4



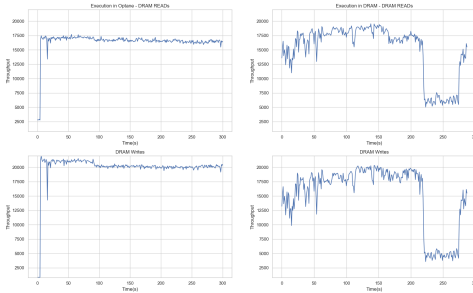
(d) Concurrency value: 8



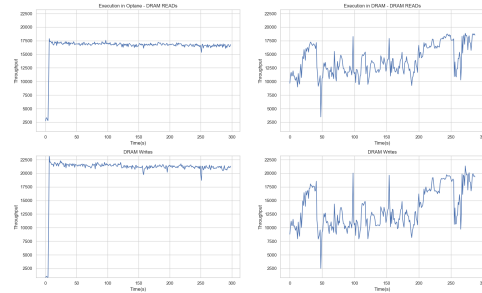
(e) Concurrency value: 16



(f) Concurrency value: 32



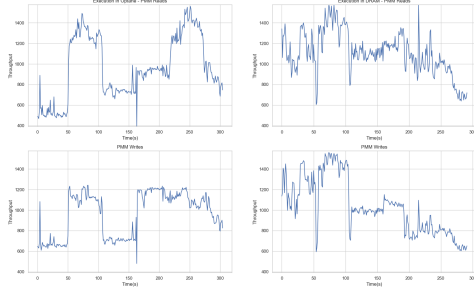
(g) Concurrency value: 64



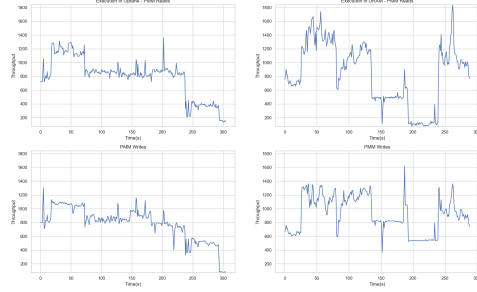
(h) Concurrency value: 80

Figure 5.1.14: Memory DRAM plots for different concurrency values - ycsb C Workload

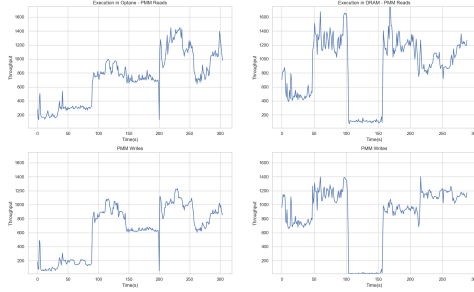
Type C: PMM Throughput



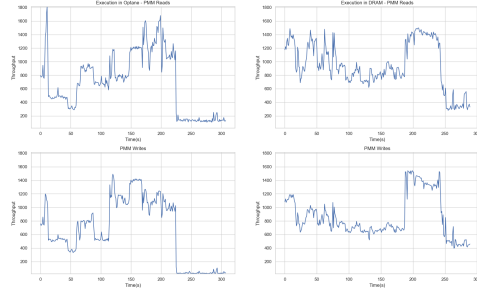
(a) Concurrency value: 1



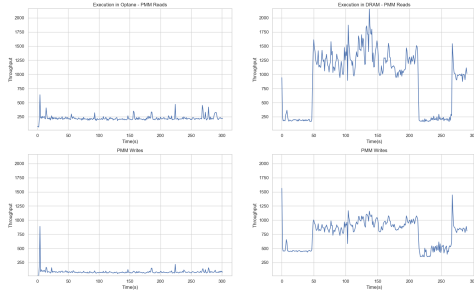
(b) Concurrency value: 2



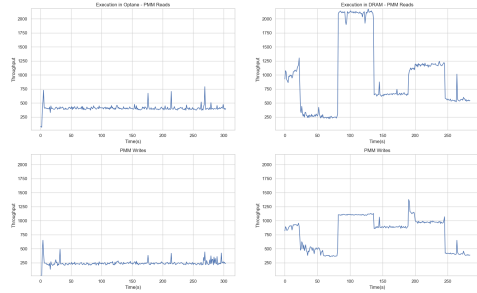
(c) Concurrency value: 4



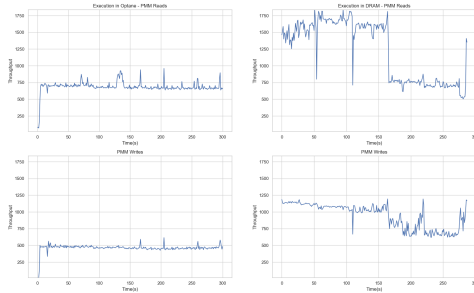
(d) Concurrency value: 8



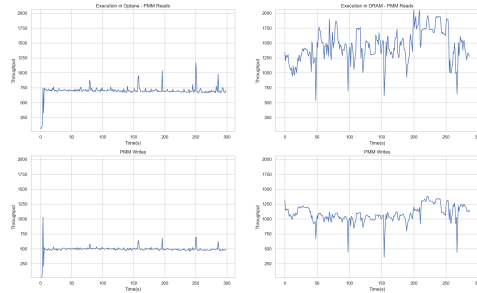
(e) Concurrency value: 16



(f) Concurrency value: 32



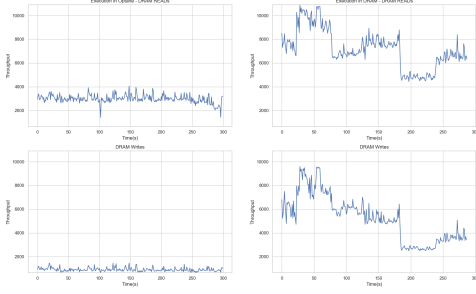
(g) Concurrency value: 64



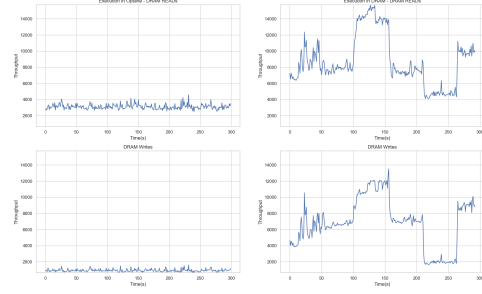
(h) Concurrency value: 80

Figure 5.1.15: PMM Throughput plots for different concurrency values - ycsb C Workload

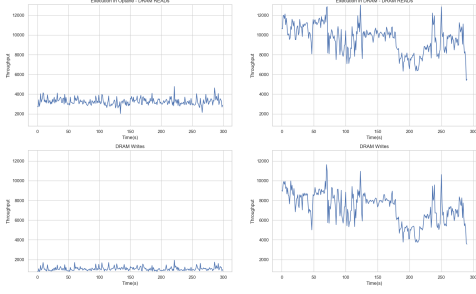
Type F: DRAM throughput



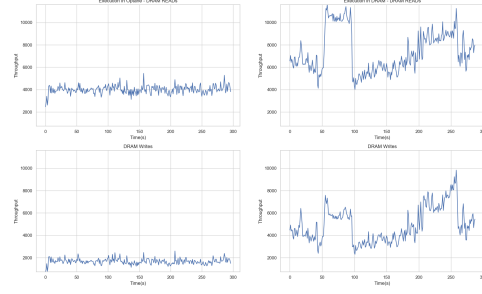
(a) Concurrency value: 1



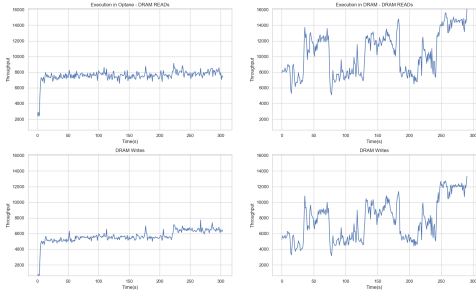
(b) Concurrency value: 2



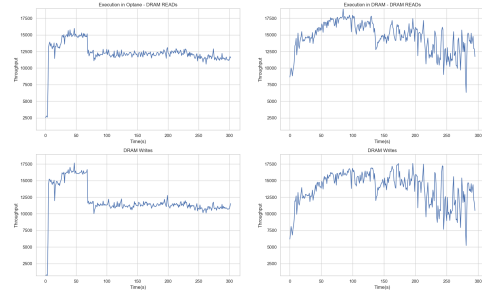
(c) Concurrency value: 4



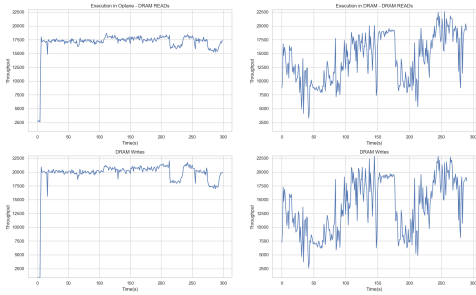
(d) Concurrency value: 8



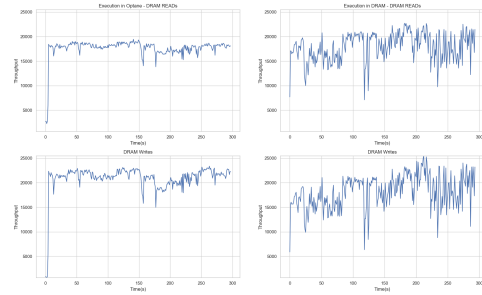
(e) Concurrency value: 16



(f) Concurrency value: 32



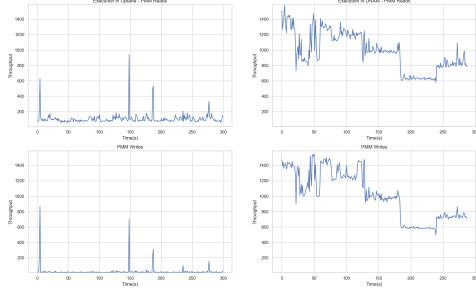
(g) Concurrency value: 64



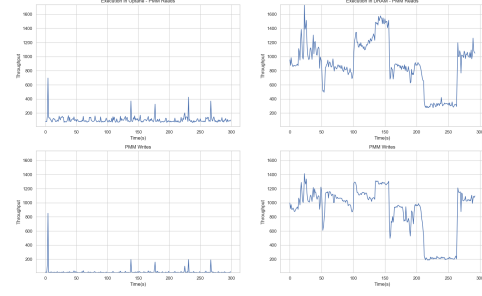
(h) Concurrency value: 80

Figure 5.1.16: DRAM throughput plots for different concurrency values - ycsb F Workload

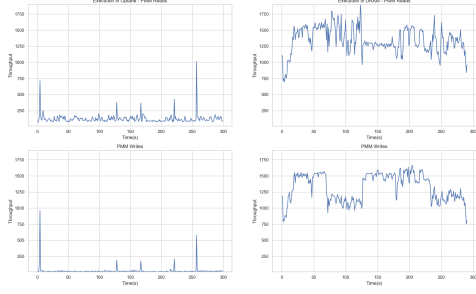
Type F: PMM Throughput



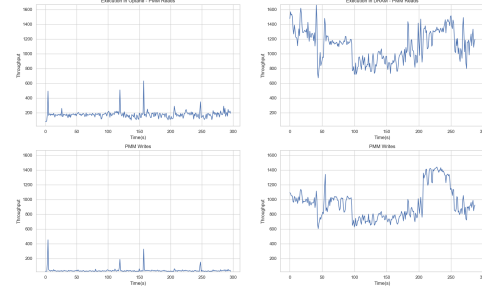
(a) Concurrency value: 1



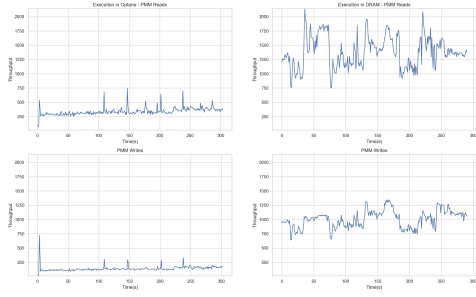
(b) Concurrency value: 2



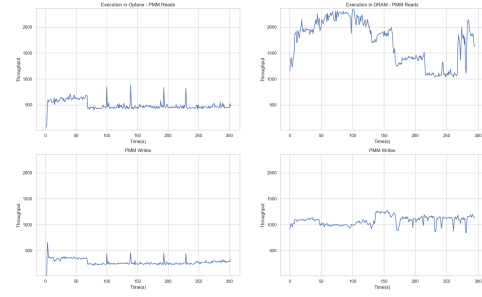
(c) Concurrency value: 4



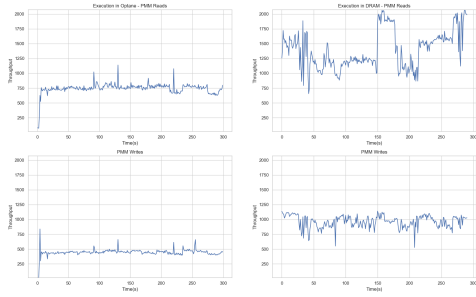
(d) Concurrency value: 8



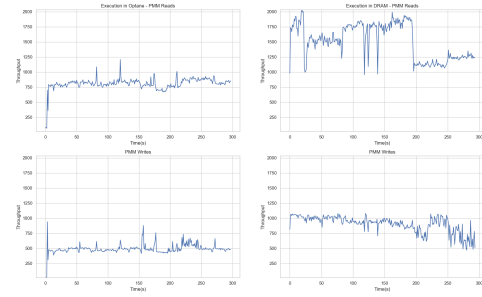
(e) Concurrency value: 16



(f) Concurrency value: 32



(g) Concurrency value: 64



(h) Concurrency value: 80

Figure 5.1.17: PMM Throughput plots for different concurrency values - ycsb F Workload

The YCSB workloads achieve relatively high DRAM throughput during their execution on both types of memory. Specifically, in forms A and F, the read throughput reaches values as high as 18000–20000. The lowest throughput is observed in form C. The memory read throughput is, in most cases, higher than the write throughput. During DRAM execution, the memory throughput achieves higher values compared to Optane execution. Additionally, during the execution of the YCSB workload, an increase in the concurrency value leads to an increase in throughput, though without extreme fluctuations in the values, expect value 64, where the increase of throughput is noticeable in all the types of workload. The values rich the value 20000 for read and write DRAM throughput. In the following table, we can observe the ranges of throughput for the different types of YCSB Workload.

Type	Read Throughput in DRAM execution	Write Throughput in DRAM execution
Type A	2000-18000	1000-17000
Type B	4000-20000	2000-20000
Type C	3000-17500	2000-20000
Type F	4000-22500	2000-25000

Table 5.1: YCSB Memory DRAM Throughput-DRAM Execution

Type	Read Throughput in Optane execution	Read Throughput in Optane execution
Type A	2000-16000	1000-15000
Type B	3000-20000	2000-16000
Type C	4000-18000	3000-22000
Type F	2000-20000	1000-22500

Table 5.2: YCSB Memory PMM Throughput-DRAM Execution

Regarding the throughput of persistent memory, it is observed to be significantly lower than the throughput of DRAM. The highest values are noted in forms A and B, ranging between 1800 and 2200. In contrast, in form F, the throughput is considerably lower, particularly during execution in Optane, with values not exceeding 1000. The memory throughput for reads proves to be higher than the memory throughput for writes. The throughput of persistent memory is higher during execution in Optane for all YCSB forms, except for form F, where the values are generally very low. In the following table, we can observe the ranges of throughput for the different types of YCSB Workload.

Type	Read Throughput in DRAM execution	Write Throughput in DRAM execution
Type A	2000-1800	200-1250
Type B	600-2000	500-1500
Type C	100-2000	300-1300
Type F	200-2500	200-1500

Table 5.3: YCSB Memory DRAM Throughput-Optane Execution

Type	Read Throughput in Optane execution	Read Throughput in Optane execution
Type A	200-2000	600-1300
Type B	400-2000	500-1750
Type C	200-1750	100-1400
Type F	100-1000	100-750

Table 5.4: YCSB Memory PMM Throughput-Optane Execution

In general, we conclude that all applications exhibit asymmetric behavior regarding accesses to DRAM and Optane DCPM, with considerable fluctuations per second. In all applications, both during execution in DRAM and execution in Optane, the throughput of persistent memory is lower than the throughput of DRAM. However, as throughput values decrease, this difference becomes smaller, and similar performance is observed. In addition, when the concurrency value increases, the DRAM throughput increases too, but PMM throughput remains similar depending on the concurrency. So, it can be noticed that workloads do not scale well in PMM and maybe a bottleneck is caused by NVM. Memory reads seem to outperform memory writes in almost all cases. The application with the lowest throughput values could be TPCC. A possible cause is that during its execution transactions occur at regular intervals rather than throughout the entire execution period. The highest throughput is observed during the execution of the YCSB application, specifically in type A.

5.2 Experiments in Interference-aware System

We introduce the profiling analysis for the same workloads and the same parameters, but this time we add interference in the system. In a real computer system, many workloads have to be done in parallel, CPUs are working at the same time executing different procedures and memory bandwidth is often stressed by requests from other applications. In order to simulate a more realistic system, we execute iBench benchmarks, that increase the intensity in resources, in parallel with the workloads execution. Then, we examine the results.

5.2.1 iBench Benchmarks

iBench [32] consists of a set of carefully-crafted benchmarks that induce interference of increasing intensity in resources that span the CPU, cache hierarchy memory storage and networking system. There are seven benchmarks and every benchmark stresses one resource at time, there is not overlapping. Each benchmark progressively puts more pressure on a specific shared resource until it reaches the requested limit. More specifically, the benchmarks [39] are 'cpu', 'l1d', 'l1i', 'l2', 'l3', 'memCap' and 'memBw'. In this work, we take advantage of the three of them, l3, memBw and cpu. L3 benchmark maps a block in the cache size and while it is executing it accesses memory block, via memcpy instruction. MemBw defines an array of size 2N and performs reads on one half of the array and writes on the other half of the array in parallel. Finally, cpu gradually engages the threads of the system.

5.2.2 Results

First, with the use of ipmctl command, we measure the memory accesses in the 6 Optane DC DIMMs when each benchmark runs in parallel. The results are presented below.

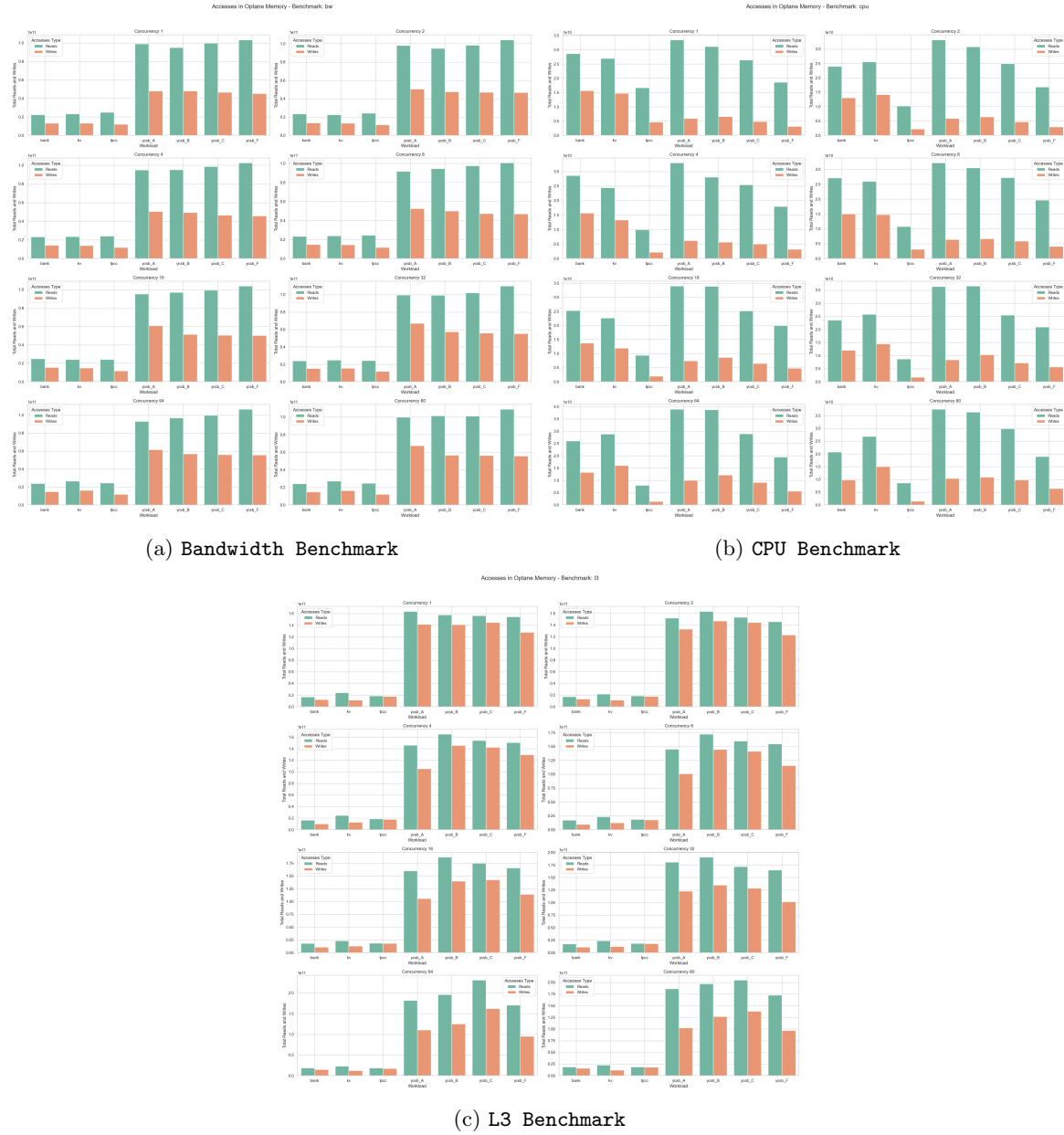


Figure 5.2.1: Memory Optane accesses plots for different concurrency values and benchmarks

We can observe that when the benchmarks of bandwidth and L3 cache run in parallel with the workloads, the total Optane accesses increase, leading to a performance degradation. However, with the cpu benchmark the memory accesses vary within a similar range of values. In all the cases, memory reads exceed memory writes. Moreover, the total memory accesses increase while the number of threads increases without extreme variations. More specifically, in Bandwidth benchmark the most accesses are observed in Ycsb_F workload and the least accesses are observed in Bank, Kv and Tpc. The values of Ycsb accesses range in $(0.9 - 1.1) \times (10^{11})$ for reads and in $(0.4 - 0.7) \times (10^{11})$ for writes, while the values of the other workloads range in $(0.2 - 0.3) \times (10^{11})$ for reads and in $(0.1 - 0.2) \times (10^{11})$ for writes. Similar behavior is appeared with the L3 benchmark. The values of Ycsb accesses range in $(1.5 - 2.3) \times (10^{11})$ for reads and in $(0.9 - 1.5) \times (10^{11})$ for writes, while the values of the other workloads range in $(0.1 - 0.25) \times (10^{11})$ for reads and in $(0.1 - 0.2) \times (10^{11})$ for writes. However, with the CPU benchmarks the accesses are reduced. The least accesses are noticed in Tpc workload

with values $(0.7 - 1.5) \times (10^{10})$ in range of for reads and in range of $(0.1 - 0.5) \times (10^{10})$ for writes. Ycsb workload reaches the highest accesses values, except ycsb F which appears similar behavior with the Tpcsc workload. For reads the values are between $(3 - 4) \times (10^{10})$ and for writes are between $(0.6 - 1.1) \times (10^{10})$. For Bank and Kv workloads, the values range between $(2 - 3) \times (10^{10})$ for reads and between $(1 - 1.5) \times (10^{10})$ for writes.

From the outputs of the workloads, we collect the latency's that the queries add to the system and plot them for each workload and each benchmark in violin plots.

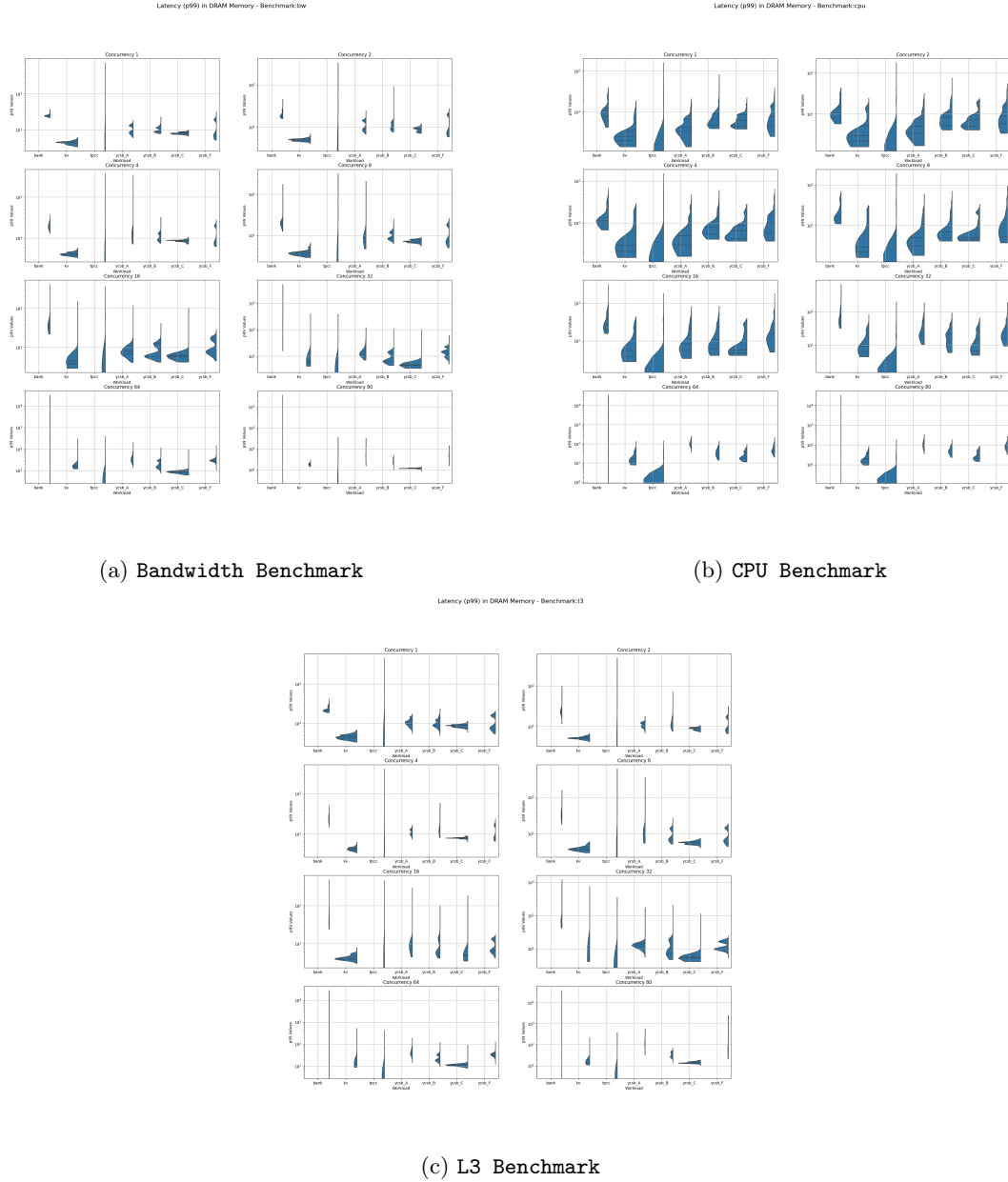


Figure 5.2.2: Query latency's for different benchmarks in DRAM execution



Figure 5.2.3: Query latency's for different benchmarks in Optane execution

It can be noticed that, compared to the isolated system, the latency's in the system with interference stay almost the same. The values vary in a range of $(10 - 10^5)$ ms in both executions. In Optane execution, the latency tend to be a little bit higher. Also, the values appeared to have a relative big variance in each workload. In TpcC workload, for both executions and every concurrency value, the latency exhibit a big range and high values. Furthermore, in Bank workload when the concurrency value is equal to 32 or greater than that, the values increase up to $(10^3 - 10^5)$ and start to disperse in the whole range. However, in Kv and Ycsb workload, the query latency remain within a specific and low range. In fact, Kv workload introduces the lowest latency into the system.

As a next step, we examine the DRAM throughput and the PMM throughput in both executions, in DRAM and in Optane DCPM. We measure the memory reads and writes for each workload separately,

under the operation of each benchmark independently. From the results, we can observe that the benchmarks achieve to stress the resources of the system. Also, while the workloads are running in parallel with the benchmarks, the systems metrics and memory throughput are affected depending on the executed benchmark. So, we have decided to analyse the particular outputs of the workloads and consequently, analyze the impact of benchmarks on their performance. For this purpose we evaluate the operations per second through runtime for every workload under the operation for every benchmark.

In the following plots, we present the impact of ibench benchmarks in the operations per second, that each workload achieve.

5.2.3 Bank

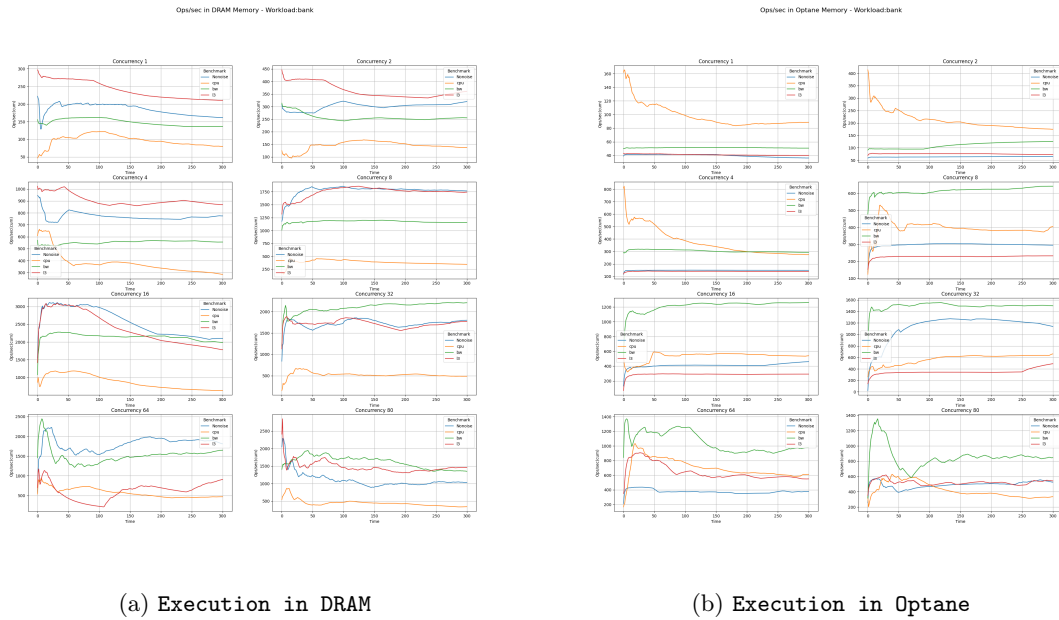
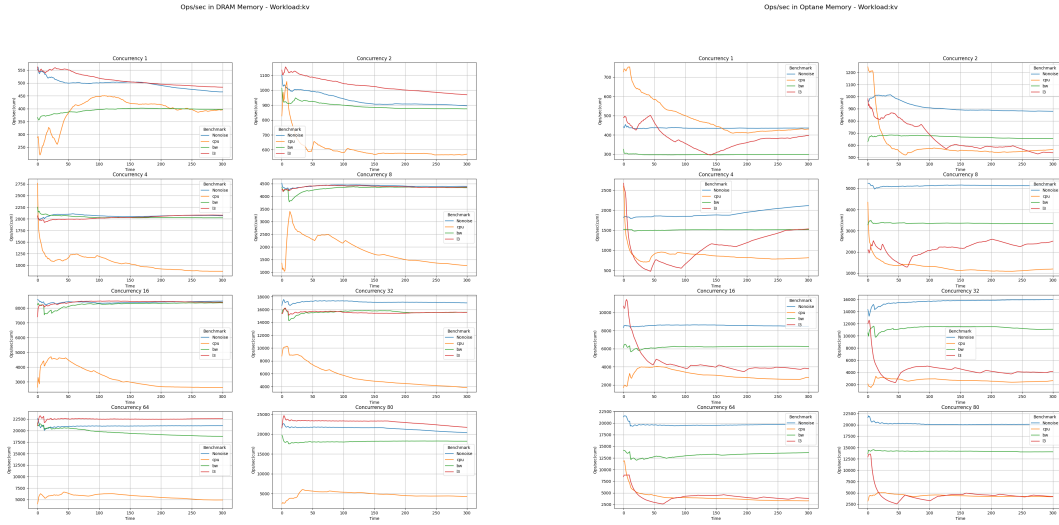


Figure 5.2.4: Ops/sec with benches for Bank

5.2.4 Kv



(a) Execution in DRAM

(b) Execution in Optane

Figure 5.2.5: Ops/sec with benches for kv

5.2.5 TPCC



(a) Execution in DRAM

(b) Execution in Optane

Figure 5.2.6: Ops/sec with benches for tpcc

5.2.6 YCSB

Type A

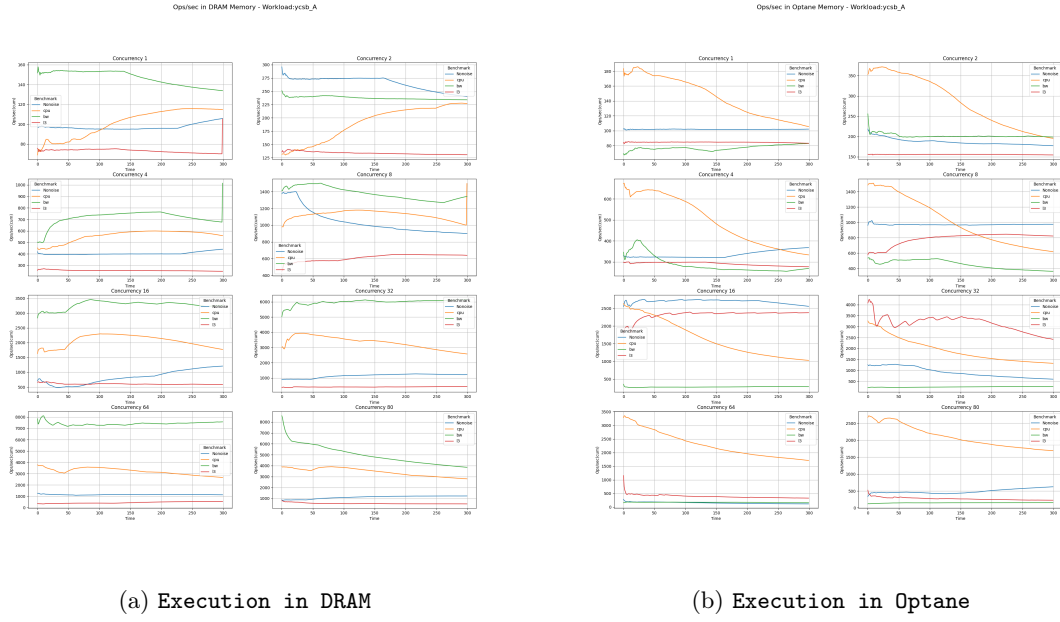


Figure 5.2.7: Ops/sec with benches for ycsb A

Type B

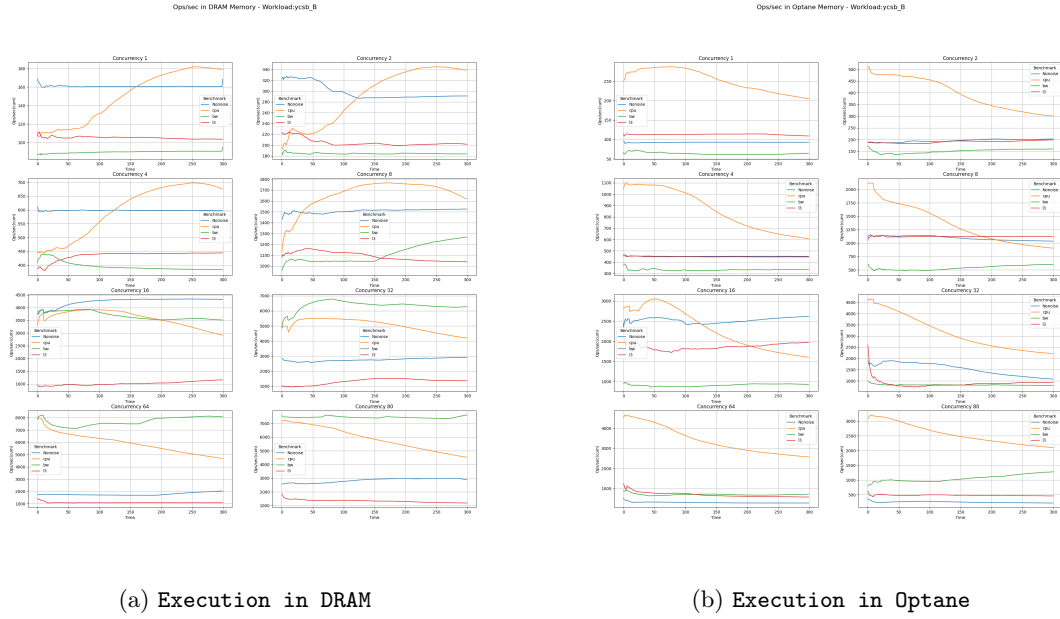


Figure 5.2.8: Ops/sec with benches for ycsb B

Type C

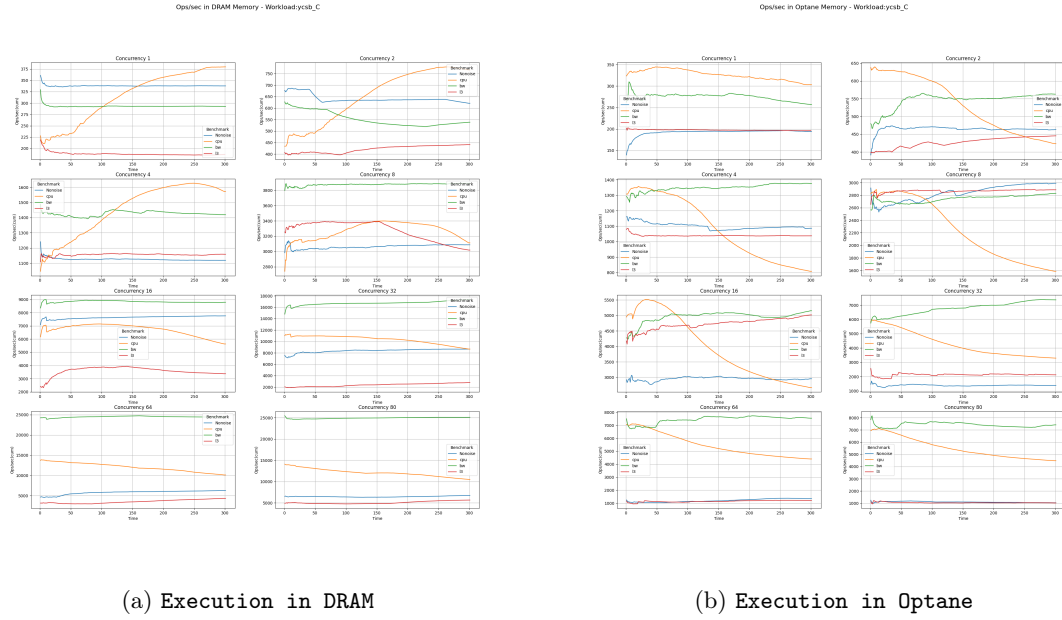


Figure 5.2.9: Ops/sec with benches for ycsb C

Type F

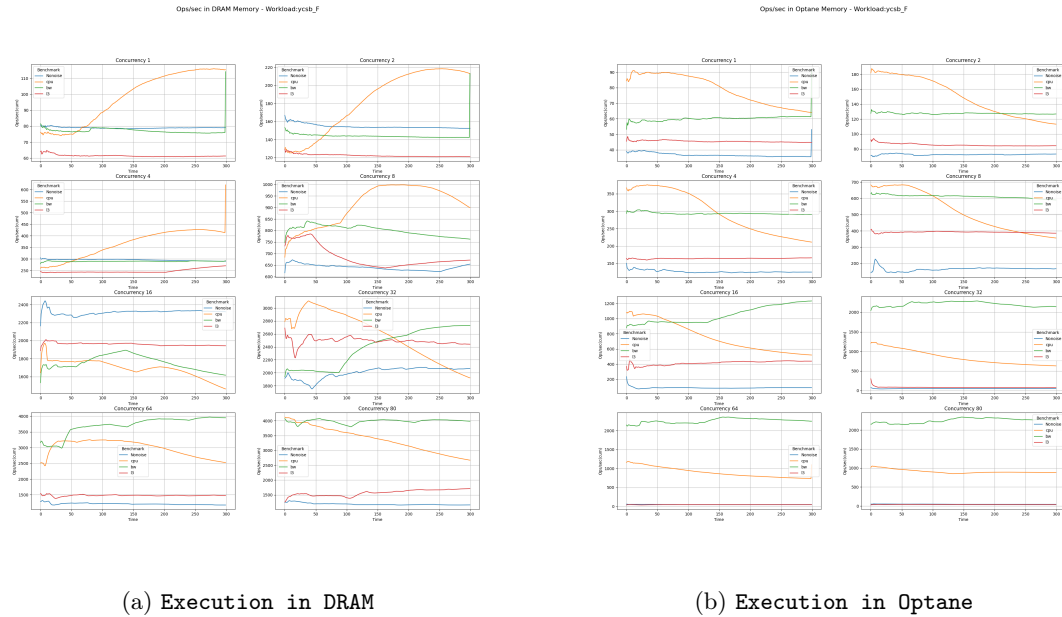


Figure 5.2.10: Ops/sec with benches for Ycsb F

The above diagrams show the impact of each benchmark in the performance of every workload. In particular, they picture the operations per second which fluctuate depending on the benchmark and over time. First of all, it can be noticed that for all the workloads, the number of operations per second increases while concurrency values increase. The impact on performance depends on the type of interference and the memory that we execute the workload. In Optane memory, it is always noticed a bigger degradation. In DRAM memory, sometimes interference do not cause degradation.

The behavior of each workload differs from each other, so it is preferable to analyze each workload separately.

In 5.2.4a and 5.2.4b we can observe the impact of benchmarks on the performance of Bank workload. In the Optane execution 5.2.4b, the values of ops/sec start from 160 and reach the value of 1500. Bandwidth benchmark leads to an increase of the ops/sec and exceeds all the other benchmarks, especially for concurrency value equal to or greater than 8. L3 benchmark presents similar performance with the execution without interference, except concurrency value 64 or 80, where the 'no-noise' execution exhibits the lowest operations per second. Finally, CPU benchmark shows high value of ops/sec for all the concurrencies and the highest value for concurrency 1,2 and 4. In fact, as a function of time, for each concurrency value, the initial values of ops/sec are higher, due to the fact that the threads are occupied gradually and through time the occupied threads increase. In DRAM execution 5.2.4a, the values of ops/sec starts from 300 and reach the value of 3000. This time, CPU benchmark shows the lowest operations per second for all the concurrency values. L3 benchmark shows similar behavior with 'no noise' execution, with a little bit higher values, so it has mediate impact on the performance of Bank workload. Bw benchmark affects the performance of the workload, it degrades the operations per second for all the concurrency values, except the value of 32.

In 5.2.5a and 5.2.5b we can observe the impact of benchmarks on the performance of Kv workload. In both executions, we can notice that all the benchmarks lead to a decrease in the number of operations per second. In the Optane execution 5.2.5b, the values of ops/sec start from 500 and reach the value of 22.500. Bandwidth benchmark reaches the highest values compared to the other benchmarks and cpu the lowest. In DRAM execution 5.2.5a, the values start from 500 and reach 25000. Cpu benchmark presents again the lowest values of operations per second. In fact, it can be noticed that for all the concurrency values, while the cpu benchmark is executing, the operations per second is decreasing over time, due to the gradual increase of the occupied threads. L3 and Bandwidth benchmark and 'no noise' executions present similar behavior, so they do not have a great impact on the operation of Kv workload.

In 5.2.6a and 5.2.6b we can observe the impact of benchmarks on the performance of Tpcck workload. Tpcck workload has the peculiarity that queries simulate a transaction processing among different tables and they are not performed every second but at more frequent intervals, like 5 seconds. So the number of operations per second are low in both executions and for every concurrency values. The values vary in a range of (0.2-1.75) ops/sec. Benchmarks do not show great impact on the performance of the workload. Although, L3 and Bandwidth benchmarks seem to have a mediate impact leading to an increase of the operations per second at some cases.

In 5.2.10, it is shown the impact of the benchmarks on the performance of Ycsb workload in every type (A,B,C,F) separately. More specifically, for type A, where the reads and the writes operations are equal, all the benchmarks affect the performance. In Optane execution 5.2.7b, the values start from 80 and reach 2500. Cpu benchmark leads to the highest increase of operations per second and the bandwidth benchmarks leads to the biggest decrease in the performance, especially for the concurrency values 8 and 16. L3 shows a mediate impact compared with the execution without interference, except at the concurrency value 32, where it shows the highest increase in the operations per second. In DRAM execution 5.2.7a, the values range from 80 to 8000. Bandwidth benchmark presents the highest increase in the number of operations for all the concurrency values. However, Cpu benchmark also appears to have great impact on the performance and leads to an increase of operations. L3 degrades the performance of workload for all the concurrency values, especially for the initial values of 1,2,4 and 8 threads.

For type B, which is a read mostly approach, the affect of the benchmarks is critical. In DRAM execution 5.2.8a, the values of operations per second range between 300 and 8000. For the initial concurrency values, until 16, bandwidth and L3 benchmarks lead to a great decrease of the operations

per second. After that, the bandwidth benchmark increase the number of operations, which maybe suggests a memory bottleneck, while L3 continue to decrease them. Cpu benchmark behavior change over time and introduce higher operations into the workload as the concurrency value increases. In Optane execution [5.2.8b](#), the values of operations start from 300 and reach 3000. Cpu benchmark increase the number of operations for all the concurrency values. At the same time, the affect of bandwidth and L3 benchmarks is obviously less than in DRAM execution. Especially, L3 benchmark shows a similar behavior with the 'no-noise' execution, except for the concurrency values 16 and 32, where it presents a degrade in operations. Bandwidth benchmark leads to a decrease of operations until the concurrency value of 32, especially for the values 8, 16 and 32.

The type C of Ycsb workload is a reads only form of this workload. In DRAM execution [5.2.9a](#), the values range between 100 and 25000 operations per second. Bandwidth benchmark shows a great impact on the workload's performance and it presents the biggest increase in the ops/sec after the concurrency value of 8 threads. Cpu benchmark affects the operation of the workload and its affect changes through time. In most cases it leads to an increase of operations. L3 has the lowest number of operations, except for the 8 threads. In Optane execution [5.2.9b](#), we can notice that all of the benchmarks affect the behavior of the workload. Cpu and Bandwidth benchmarks lead to an increase of the operations per second for all the concurrency values. However, L3 benchmark presents similar performance with the execution without interference, except when the concurrency value is equal to 16 and L3 has obviously higher number of operations. The general values of ops/sec start from 150 and reach 8000 ops/sec.

Type F is a read-modify-write version of ycsb workload. In DRAM execution [5.2.10a](#), the values are between 60 and 4000 operations per second. L3 benchmark has a mediate impact on workloads performance, except for concurrency values 16 and 32, where it leads to a decrease and a abrupt increase of operations. Bandwidth benchmark gets higher performance as concurrency increase and exceed the number of operations in the no-interference execution after the concurrency value 8. Cpu benchmark has a high impact on the performance for all the concurrency values and increases the number of operations per second. In Optane execution [5.2.9b](#), the values of operations are 30-2500. In this execution, it can be noticed again that bandwidth and cpu benchmarks have a great impact on performance and lead to an increase of operation for all the concurrency values. L3 has a mediate impact and presents similar behavior, except from values of threads equal to 8 and 16, where the ops/sec are higher than the no-interference execution.

Chapter 6

Proposed Methodology

In this chapter, we present our proposed methodology in order to determine the placement of application data and indexes through the heterogeneous memory system. More specifically, we introduce the idea of training a machine learning model in order to classify the applications into five main classes, depending on their performance and their memory behavior. The classes discriminate the applications into Memory Bound, DRAM-Bandwidth Bound, PMM-Read Bandwidth Bound, PMM-Write Bandwidth Bound, CPU-Bound and undefined. Based on that classification, decisions can be made regarding the allocation of applications to the memories of the DRAM/NVM Heterogeneous system.

The first step in order to train the ML model is creating the appropriate input dataset. We collect the data from the Performance Counter Monitor (PCM) and the Performance Counter Monitor specified for Memory (PCM-Memory), that generate data every second while the workloads are executing. The data obtain information about the system, the processor, the memory accesses and the memory footprint. We choose to utilize the data obtained from the profiling analysis in the interference aware system by using iBench Benchmarks. From these data, we exclude the data referred to the metrics of the system and we combine in one dataset the data from the PCM monitor and the PCM-Memory monitor. The next goal is to determine the class of each element in the dataset.

In order to classify our input data and then train the model to predict this classification, it is necessary to pose specific thresholds in the memory and the processor metrics. The thresholds are based on the values of the IPC, in order to decide if an application is CPU-bound or not. In order to decide if the application is Memory bound or not, we examine the miss rate of the last level cache (L3) during the execution in DRAM and the execution in Optane. The miss rate is calculated as the percentage of misses relative to the total accesses in the L3 cache. Finally, we compute the maximum bandwidth that it can be achieved in DRAM and in Optane for reads and writes and compare them with the bandwidth that the workloads achieve, in order to decide if the workloads are DRAM bandwidth bound, PMM Read bandwidth bound, PMM Write bandwidth bound or not.

For the computation of the maximum bandwidth of the memories we execute specific benchmarks that stress the memory bandwidth. In particular, we take advantage of the Intel® Memory Latency Checker and via the command `'mlc -max_bandwidth'` the maximum bandwidth of the memories is known. This command prints maximum memory b/w (by automatically varying load injection rates) for various read-write ratios with all local accesses. So the values of the maximum bandwidth are the following:

DRAM Max Bw	42324.47 MB/s
Optane Max Read Bw	1700.55 MB/s
Optane Max Write Bw	930.76 MB/s

Table 6.1: Max Bandwidth Values

In the following table, the declaration of the classes are presented.

Class 1	Memory Bound	L3 miss rate > 0.6
Class 2	RAM Bw Bound	DRAM Bw $> 0.6 \times$ DRAM Max Bw (Read or Write)
Class 3	PMM Read Bw Bound	PMM Read Bw $> 0.6 \times$ PMM Max Bw (Read)
Class 4	PMM Write Bw Bound	PMM Write Bw $> 0.6 \times$ PMM Max Bw (Write)
Class 5	CPU Bound	IPC > 1
Class 0	Undefined	

Table 6.2: User-Level Thresholds for Classification

The distribution of the classes are presented in the diagram below:

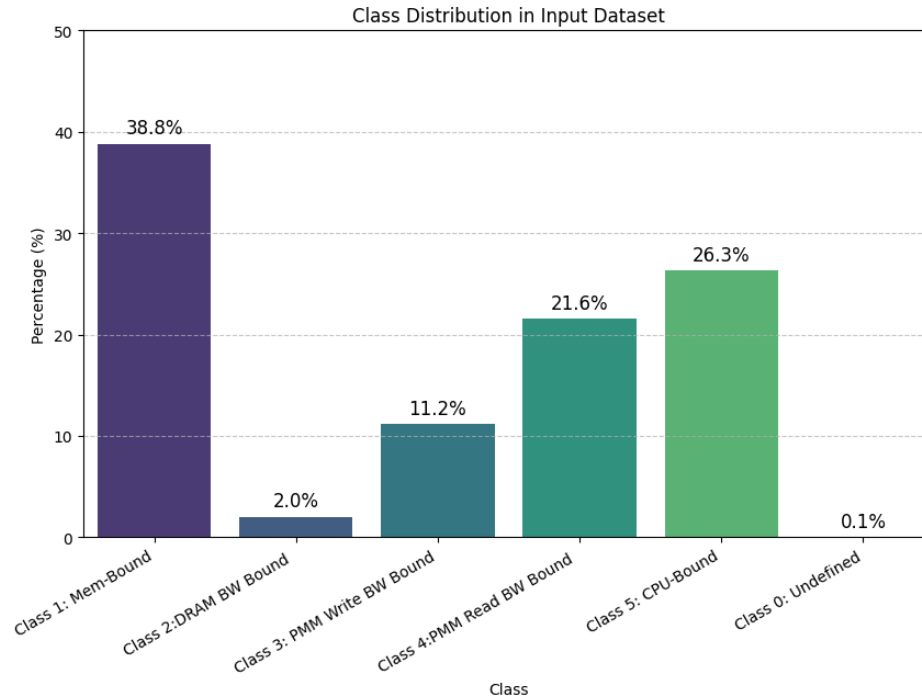


Figure 6.0.1: Distribution of Input Samples into the Classes

The majority of the applications are classified as memory-bound. The rest of the samples are distributed among PMM Read or Write Bound and CPU-bound classes. The least samples are included into DRAM BW Bound class and the undefined class, which indicates a more basic operation of workloads.

Then, we apply the percentage split method to divide our input dataset into two sets, the train and test set, with a random factor equal to 42. We use 70% of the data to train the model and the 30% of

the data to test the trained model. The comparison of the predicted labels and the real labels of the data are useful in order to evaluate the accuracy of the model afterwards. The data is conducted by Bank and TpcC Workload. Furthermore, an important step is the pre-processing of numerical data to facilitate the training of the model, especially those based on propagation and iterations according to a loss function, such as MLP and Logistic Regression. For this purpose, we use the following pipeline for the preprocessing:

```
numeric_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='mean')),
    ('scaler', StandardScaler())])
```

`SimpleImputer()` function from sklearn library with strategy 'mean' fills in the missing values with the mean value of the specific system characteristic and `StandardScaler()` function standardize the numerical features of the input dataset by subtracting the mean and dividing by the standard deviation of each system characteristic separately. In this way, errors due to missing values and dominance of features with larger magnitudes are prevented during the training of the model.

We choose to examine four supervised machine learning algorithms: MLP, Random Forests, SVM and Logistics Regression. These algorithms are based on different machine learning algorithms, so it is interesting to compare their accuracy and their performance. Also, all of them support the process of big datasets during the training and can make predictions into multiple classes and not necessarily for linear problems.

Finally, before starting to train the models, it is necessary to define the appropriate hyper parameters of the models to increase their performance and select the best configuration. More specifically, for Random Forests we execute `GridSearch`, testing the values [50,100,150,200,250] for estimators and the values [10,15,20,25,30] for the max depth. For `SVM()`, we examine the kernels ['rbf', 'sigmoid', 'poly'] and the values [0.1,1,10,100] for C, which is the regularization factor. For MLP, we test its performance by using the activation functions: ['tanh', 'relu', 'logistic'] and the solvers ['adam', 'lbfgs', 'sgd']. For Logistic Regression, we use `OneVsRestClassifier` for multiple classes and we try two different solvers ['newton-cg', 'saga', 'lbfgs'] and the values [0.1,1,10,100] for C factor.

The overall proposed methodology is summarized in the following diagram:

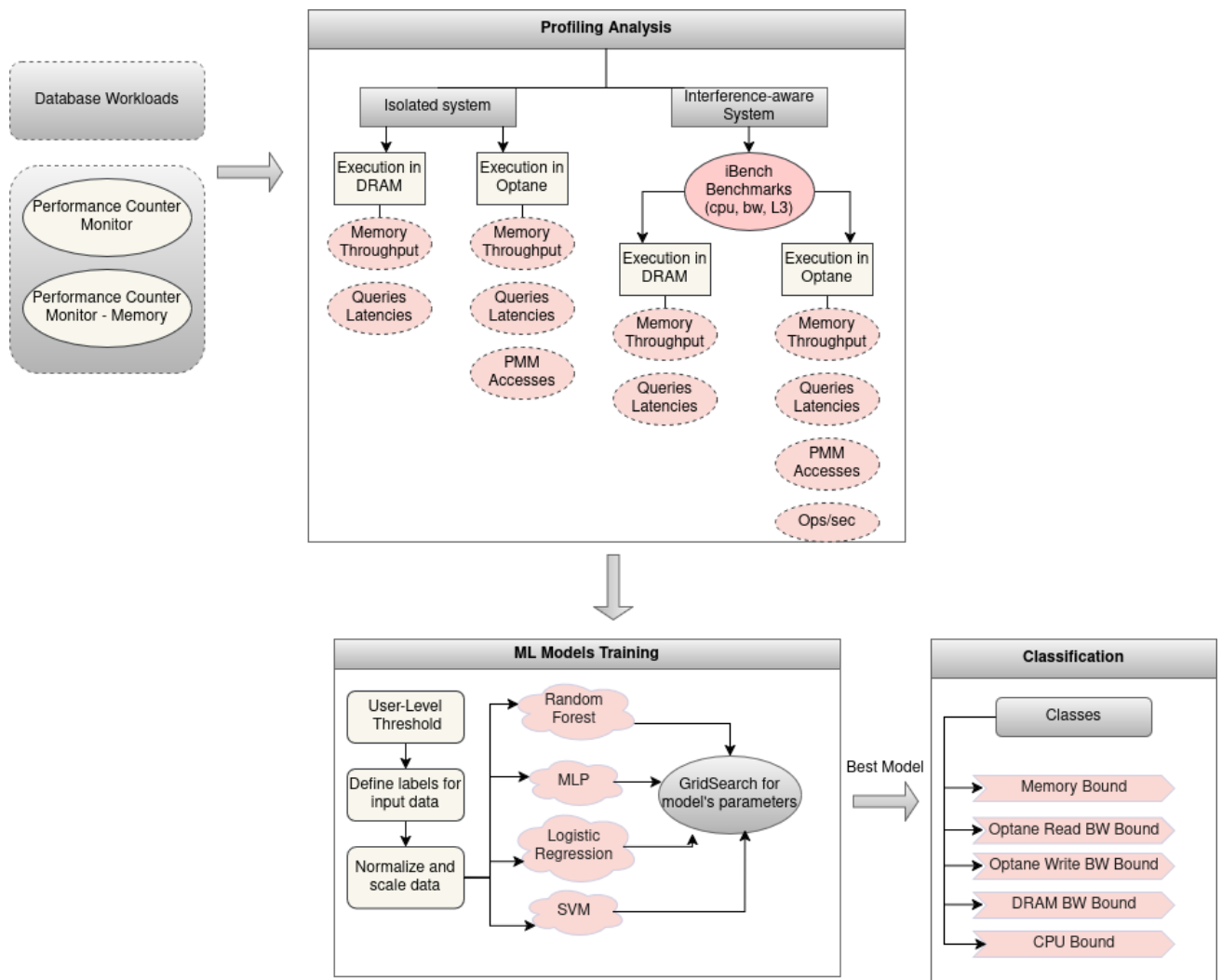


Figure 6.0.2: Proposed Methodology

Chapter 7

Experimental Setup and Evaluation

In this chapter, we present the evaluation of the models after the training process. In order to evaluate the different machine learning models and conclude in the final model's choice, we take into account some specific metrics. First of all, we examine their performance depending on the different values of hyper parameters. For this purpose, it is important to consider the accuracy score and the mean fit time that is needed for the training. Besides, the final choice of a ML model is a trade off between accuracy and performance. Then, we evaluate the models by the accuracy of their predictions, their precision and their recall in the test input dataset. Accuracy computes how many labels are predicted right, precision refers to the number of instances predicted as a class which actually belong to that class and recall measures how many actual instances of a class were correctly predicted. As a final step, we evaluate the accuracy of models in datasets that have not been trained for.

First of all, in Chapter 6 we describe the way that we choose the best parameters for the model's configurations via GridSearch. In this section, we present heatmaps that shows the way that accuracy and mean fit times change depending on the parameters for all the ML models.

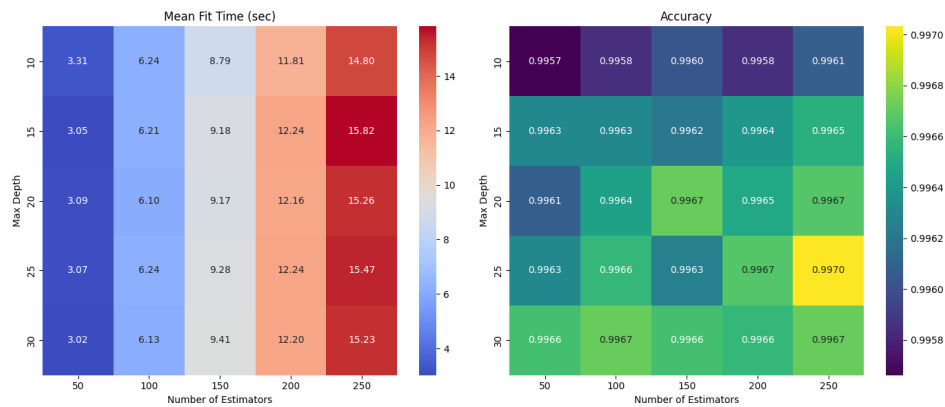


Figure 7.0.1: Heatmaps of Mean Fit time and Accuracy for Random Forest

It can be noticed that Random Forest achieves high accuracy scores during GridSearch of parameters for all the values of max_depth and n_estimators. However, the mean fit time increases depending the number of estimators. As the estimators increase, mean fit time increases. Given a specific number of estimators, the parameter of maximum depth does not introduce additional fit time to the system, but as max depth increases there is a relative increase in accuracy score.

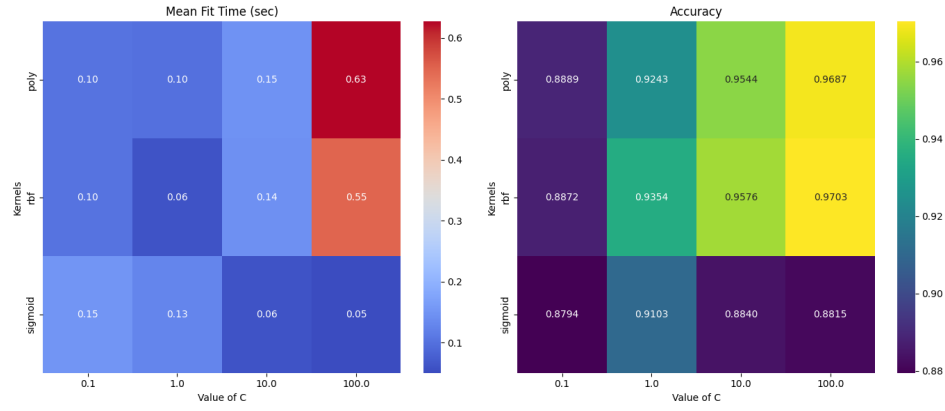


Figure 7.0.2: Heatmaps of Mean Fit time and Accuracy for SVM

SVM model presents quite low mean fit time for all the parameters and accuracy scores over 87%. In particular, the kernel of 'rbf' achieve the highest accuracy scores and this scores increases as the regularization factor of C increases. The increase of factor C improve accuracy score for all the kernels. The mean fit time stays low for all the kernels, especially for the 'sigmoid'. As the factor C reach the value 100 there is an increase in the mean fit time for kernels 'rbf' and 'poly', but time remains low, below 1 second.

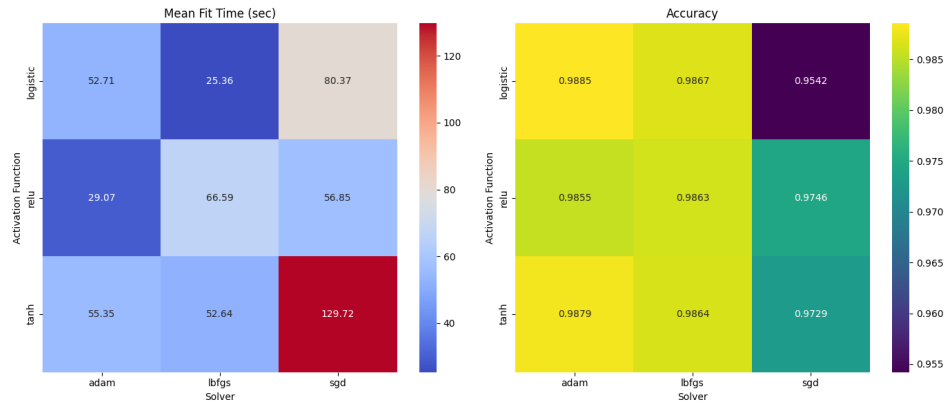


Figure 7.0.3: Heatmaps of Mean Fit time and Accuracy for MLP

From the above heatmaps we can observe that MLP achieves high accuracy scores, but mean fit time is quite high for all the parameters. Especially, for optimization solver 'sgd' and for 'tanh' activation function, fit time presents the highest values. Also, for solver 'sgd' the accuracy score is the lowest. Different activation functions have mediate impact on the models accuracy scores.

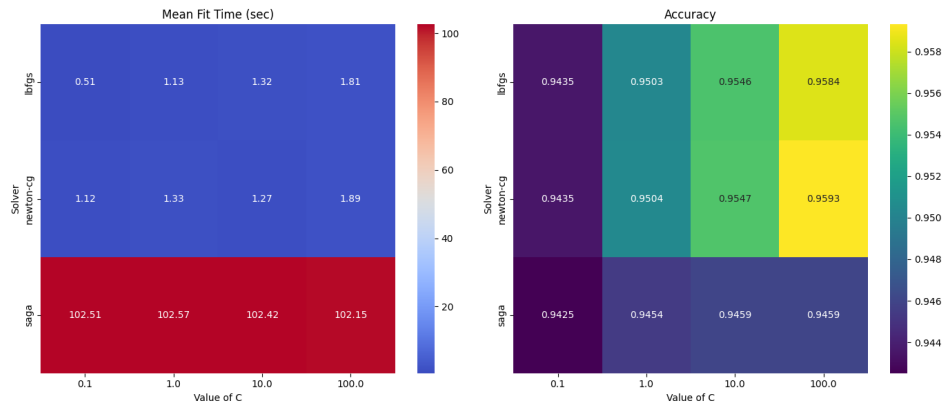


Figure 7.0.4: Heatmaps of Mean Fit time and Accuracy for Logistic Regression

Logistic regression achieves the lowest accuracy scores compared to the other models. Additionally, for the 'saga' solver mean fit time turns out to be relatively high. The best accuracy scores and the lowest fit time are done with 'lbfgs' and 'newton-cg' solvers. In fact, as factor C increases the accuracy score also increases.

The final configurations of the best models are the followings:

Random Forest	n_estimators = 250	max_depth = 25
SVM	kernel = 'rbf'	C = 100
MLP	activation = 'logistic'	solver = 'adam'
Logistic Regression	solver = 'newton-cg'	C = 100

Table 7.1: ML Model's Best Parameters

After the test of the ML models into the input test dataset, the following results are obtained.

Model	Accuracy	Precision	Recall
Random Forest	0.9968	0.994	0.9274
SVM	0.9852	0.9849	0.9013
MLP	0.9898	0.953	0.8867
Logistic Regression	0.9593	0.9682	0.8066

Table 7.2: Accuracy, Precision and Recall for each model

Finally, we examine the accuracy of the ML models at Kv and Ycsb workloads. The models are not trained in these workloads, so we can derive more objective measurements of accuracy and avoid over-fitting phenomena. The results are presented in the following plot:

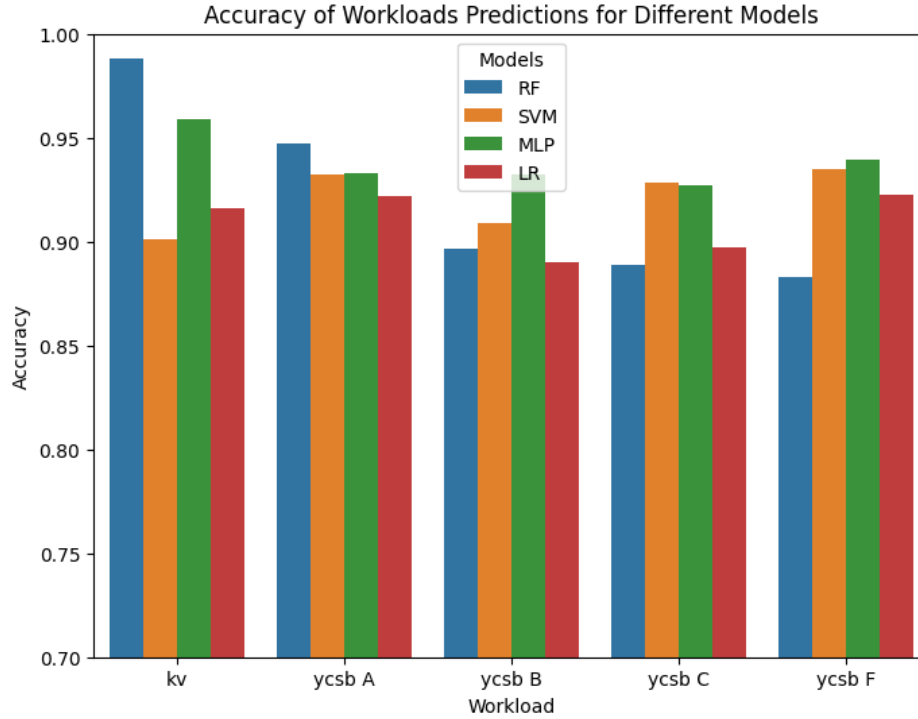


Figure 7.0.5: Accuracy of Workloads Predictions for Different Models

	Random Forest	SVM	MLP	Logistic Regression
Kv	0.9886	0.9235	0.9576	0.9133
Ycsb A	0.9456	0.9411	0.9424	0.9249
Ycsb B	0.8977	0.9321	0.9416	0.8946
Ycsb C	0.8853	0.9396	0.9424	0.9070
Ycsb F	0.8799	0.948	0.9435	0.9325

Table 7.3: Accuracy of Workloads predictions for the different models

From the above accuracy scores, we can observe that all the models achieve a high accuracy score, over 88%. In fact, MLP and SVM achieve the highest values of accuracy in all the datasets that are tested in. So, they could be appropriate solutions to classify data and predict the performance of the workloads. However, as we can notice at the heatmaps 7.0.2 and 7.0.3, the mean fit time of MLP is much higher than the mean fit time of SVM. In a trade off performance and accuracy, SVM model would be a more sufficient choice.

Based on the final classification of the data, we could decide the way the workloads could be placed into the heterogeneous memory system. Memory-bound applications, which are not so latency-sensitive, could be placed into Optane DCPM, due to its increased capacity compared to DRAM. However, applications which are CPU-bound, it would be preferable to be placed into DRAM for lower latency accesses. DRAM and PMM bandwidth bound applications could be placed into DRAM and migrate to Optane during runtime based on a possible priority function.

Chapter 8

Conclusion and Future Work

8.1 Conclusion

In this work, the deployment, the profiling analysis and the characterization of Database workloads in a Heterogeneous memory system are presented. The experiments were conducted into an isolated system and in an interference-aware system by using specific benchmarks that stress the resources of the system. Each workload was executed separately in both DRAM and in Optane DCPM. We can observe that each workload shows differences in their performance depending on the memory that are executed at. All the workloads present an asymmetry in their memory accesses and their latency that introduce into the system. Their behavior can not be easily predicted. In fact, inference leads to a further performance degradation. For this purpose, machine learning models are trained to classify these workloads based on the data collected from the experimental analysis through runtime. The workloads are classified as memory-bound, cpu-bound, RAM-Bandwidth bound, PMM Read Bandwidth bound and PMM Write Bandwidth bound. So, it is proven that models can achieve accuracy of predictions over 90%. In fact, SVM model can achieve the best accuracy scores in the least mean fit time. Based on that classification, specific guidelines could be given for the dynamic placement of workloads in the Heterogeneous Memory System.

8.2 Future Work

This work could be further optimized with the integration of the trained ML models in the runtime decision making placement policy through memories in the Heterogeneous memory system, DRAM and Optane DCPM. Moreover, a possible improvement would be the training of an advance neural network or a LLM to predict the performance of the workloads and take automatically decisions of the final placement. As far as profiling analysis is concerned, the workloads could be evaluated also by their energy consumption. Also, their performance could be tested for different databases features, like the shape of tables, the number of attributes, the key hash seed etc. Finally, other applications could be examined, that have great memory demands, such as Machine learning applications.

Bibliography

- [1] G. Gill, R. Dathathri, L. Hoang, R. Peri, and K. Pingali, “Single machine graph analytics on massive datasets using intel optane dc persistent memory,” 2020. [Online]. Available:
- [2] B. Mahesh, “Machine learning algorithms - a review,” *International Journal of Science and Research (IJSR)*, vol. 9, pp. 381–386, 1 2020.
- [3] (2024) Support vector machine algorithm. [Online]. Available:
- [4] (2024) Neural networks (supervised). [Online]. Available:
- [5] M. Katsaragakis, C. Baloukas, L. Papadopoulos, and F. Catthoor, “Performance, energy and nvm lifetime-aware data structure refinement and placement for heterogeneous memory systems,” 7 2023. [Online]. Available:
- [6] S. Shiratake, “Scaling and performance challenges of future dram,” in *2020 IEEE International Memory Workshop (IMW)*, 2020, pp. 1–3.
- [7] J. Boukhobza, S. Rubini, R. Chen, and Z. Shao, “Emerging nvm: A survey on architectural integration and research challenges,” *ACM Trans. Des. Autom. Electron. Syst.*, vol. 23, no. 2, Nov. 2017. [Online]. Available:
- [8] J. Yang, B. Li, and D. J. Lilja, “Exploring performance characteristics of the optane 3d xpoint storage technology,” *ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS)*, vol. 5, no. 1, pp. 1–28, 2020.
- [9] V. Gupta, S. Kapur, S. Saurabh, and A. Grover, “Resistive random access memory: a review of device challenges,” *IETE Technical Review*, vol. 37, no. 4, pp. 377–390, 2020.
- [10] E. Kültürsay, M. Kandemir, A. Sivasubramaniam, and O. Mutlu, “Evaluating stt-ram as an energy-efficient main memory alternative,” in *2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. IEEE, 2013, pp. 256–267.
- [11] Y. Guo, W. Xiao, Q. Liu, and X. He, “A cost-effective and energy-efficient architecture for die-stacked dram/nvm memory systems,” in *2018 IEEE 37th International Performance Computing and Communications Conference (IPCCC)*. IEEE, 2018, pp. 1–2.
- [12] A. Narayan, T. Zhang, S. Aga, S. Narayanasamy, and A. Coskun, “Moca: Memory object classification and allocation in heterogeneous memory systems,” in *Proceedings - 2018 IEEE 32nd International Parallel and Distributed Processing Symposium, IPDPS 2018*. Institute of Electrical and Electronics Engineers Inc., 8 2018, pp. 326–335.
- [13] B. Kammerdiener, J. Z. McMichael, M. R. Jantz, K. A. Doshi, and T. Jones, “Flexible and effective object tiering for heterogeneous memory systems,” in *International Symposium on Memory Management, ISMM*. Association for Computing Machinery, 6 2023, pp. 163–175.
- [14] M. Laghari, N. Ahmad, and D. Unat, “Phase-based data placement scheme for heterogeneous memory systems.”

- [15] M. Jorda, S. Rai, E. Ayguade, J. Labarta, and A. J. Pena, “ecohmem: Improving object placement methodology for hybrid memory systems in hpc,” in *Proceedings - IEEE International Conference on Cluster Computing, ICC*, vol. 2022-September. Institute of Electrical and Electronics Engineers Inc., 2022, pp. 278–288.
- [16] M. Katsaragakis, K. Stavrakakis, D. Masouros, L. Papadopoulos, and D. Soudris, “Adjacent lstm-based page scheduling for hybrid dram/nvm memory systems,” in *OpenAccess Series in Informatics*, vol. 107. Schloss Dagstuhl- Leibniz-Zentrum für Informatik GmbH, Dagstuhl Publishing, 3 2023.
- [17] A. Hazarika, S. Poddar, and H. Rahaman, “Survey on memory management techniques in heterogeneous computing systems,” pp. 47–60, 3 2020.
- [18] I. Corporation, “Pmdk: Persistent memory development kit,” .
- [19] J. Izraelevitz, J. Yang, L. Zhang, J. Kim, X. Liu, A. Memaripour, Y. J. Soh, Z. Wang, Y. Xu, S. R. Dulloor, J. Zhao, and S. Swanson, “Basic performance measurements of the intel optane dc persistent memory module,” 3 2019. [Online]. Available:
- [20] T. Hirofuchi and R. Takano, “A prompt report on the performance of intel optane DC persistent memory module,” *CoRR*, vol. abs/2002.06018, 2020. [Online]. Available:
- [21] A. Pavlo and M. Aslett, “What’s really new with newsql?” *SIGMOD Rec.*, vol. 45, no. 2, p. 45–55, Sep. 2016. [Online]. Available:
- [22] C. Labs, “Start a local cluster,” .
- [23] R. Taft, I. Sharif, A. Matei, N. Vanbenschoten, J. Lewis, T. Grieger, K. Niemi, A. Woods, A. Birzin, R. Poss, P. Bardea, A. Ranade, B. Darnell, B. Gruneir, J. Jaffray, L. Zhang, and P. Mattis, “Cockroachdb: The resilient geo-distributed sql database,” in *Proceedings of the ACM SIGMOD International Conference on Management of Data*. Association for Computing Machinery, 6 2020, pp. 1493–1509.
- [24] V. Nasteski, “An overview of the supervised machine learning methods,” *Horizons. b*, vol. 4, no. 51-62, p. 56, 2017.
- [25] A. Mankodi, A. Bhatt, B. Chaudhury, R. Kumar, and A. Amrutiya, “Evaluating machine learning models for disparate computer systems performance prediction,” in *2020 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, 2020, pp. 1–6.
- [26] L. Breiman, “Random forests,” *Machine learning*, vol. 45, pp. 5–32, 2001.
- [27] J. E. T. Akinsola, “Supervised machine learning algorithms: Classification and comparison,” *International Journal of Computer Trends and Technology (IJCTT)*, vol. 48, pp. 128 – 138, 06 2017.
- [28] R. Battiti, M. Brunato, and F. Mascia, *Reactive search and intelligent optimization*. Springer Science & Business Media, 2008, vol. 45.
- [29] K. . Kirasich, T. . Smith, and B. Sadler, “Random forest vs logistic regression: Binary classification for heterogeneous datasets,” Tech. Rep. [Online]. Available:
- [30] I. Corporation, “Pcm: Performance counter monitor,” .
- [31] —, “ipmctl: Nvm dimm utility,” .
- [32] C. Delimitrou and C. Kozyrakis, “ibench: Quantifying interference for datacenter applications,” 2023.
- [33] T. Vogelsang, “Understanding the energy consumption of dynamic random access memories,” in *2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture*, 2010, pp. 363–374.

- [34] I. B. Peng, M. B. Gokhale, and E. W. Green, “System evaluation of the intel optane byte-addressable nvm,” in *ACM International Conference Proceeding Series*. Association for Computing Machinery, 9 2019, pp. 304–315.
- [35] T. K. Ho, “Random decision forests,” in *Proceedings of the 3rd International Conference on Document Analysis and Recognition*, 1995, pp. 278–282.
- [36] M. Awad and R. Khanna, *Efficient learning machines: theories, concepts, and applications for engineers and system designers*. Springer nature, 2015.
- [37] C. Cortes and V. Vapnik, “Support-vector networks,” *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.
- [38] V. Nasteski, “An overview of the supervised machine learning methods,” *HORIZONS.B*, vol. 4, pp. 51–62, 12 2017.
- [39] S. M. Group, “ibench: A micro-benchmarking suite for storage systems,” .