ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ
http://www.cslab.ece.ntua.gr

Διπλωματικές Εργασίες
Ακαδημαϊκό έτος 2023-2024

I. Αρχιτεκτονική Υπολογιστών

1 Αρχιτεκτονική Υπολογιστών και Μηχανική Μάθηση

Αλγόριθμοι μηχανικής μάθησης ταξινόμησης (classification) και πρόβλεψης (prediction) εφαρμόζονται κατά κανόνα σε τομείς όπως η όραση υπολογιστών, η επεξεργασία φυσικής γλώσσας κ.α, πετυχαίνοντας εντυπωσιακά αποτελέσματα. Πλέον, έχουν αρχίσει και αυξάνονται οι περιπτώσεις εφαρμογής/χρήσης τους για τη βελτίωση της ίδιας της επίδοσης ενός υπολογιστικού συστήματος.

Οι παρακάτω εργασίες εστιάζουν τόσο στην χρήση τεχνικών μηχανικής μάθησης στην αρχιτεκτονική υπολογιστών όσο και στην αποδοτική υλοποίηση των ίδιων των αλγορίθμων.

1.1 Εφαρμογή αλγορίθμων μηχανικής μάθησης στην αρχιτεκτονική υπολογιστών

Οι σύγχρονες αρχιτεκτονικές συχνά εμπλέκουν ευριστικές μεθόδους, μεθόδους πρόβλεψης/υποθετικής εκτέλεσης για τη μεγιστοποίηση της επίδοσης ενός συστήματος. Παράδειγμα μπορεί να θεωρηθεί η χρήση προανάκλησης (prefetching), που χρησιμοποιείται για την αντιμετώπιση ενός σημαντικού σημείου συμφόρησης (bottleneck) επίδοσης των σύγχρονων αρχιτεκτονικών, του κόστους προσπέλασης της κύριας μνήμης. Σκοπός της συγκεκριμένης διπλωματικής είναι η διερεύνηση της δυνατότητας εφαρμογής αλγορίθμων μηχανικής μάθησης για τη βελτιστοποίηση της επίδοσης με στόχο βελτιστοποιήσεις στη χρήση των κρυφών μνημών (caches, prefetching), στο μηχανισμό εικονικής μνήμης (TLBs), στο μηχανισμό πρόβλεψης διακλαδώσεων (branch prediction) κ.α.

Στόχος είναι αρχικά να χρησιμοποιηθεί λογισμικό μηχανικής μάθησης (π.χ pytorch) με πραγματικά δεδομένα από σύγχρονα μηχανήματα για τη μελέτη διαφορετικών μοντέλων (π.χ LSTMs, transformers,

κ.α.). Στη συνέχεια, ανάλογα με τα συμπεράσματα του πρώτου βήματος, θα επιχειρήσουμε να αξιολογήσουμε τη δυνατότητα εφαρμογής τους σε επίπεδο μικροαρχιτεκτονικής λαμβάνοντας υπόψη την πολυπλοκότητα, το χρόνο απόκρισης και την κατανάλωση χώρου και ενέργειας.

Σχετικά Μαθήματα: Προηγμένα Θέματα Αρχιτεκτονικής Υπολογιστών, Νευρωνικά Δίκτυα και Ευφυή Υπολογιστικά Συστήματα

Σχετική Βιβλιογραφία:

1. Learning Memory Access Patterns
2. Dynamic Branch Prediction with Perceptrons
3. BranchNet: A Convolutional Neural Network to Predict Hard-to-Predict Branches
4. Virtual Address Translation via Learned Page Table Indexes
5. SmartChoices: Hybridizing Programming and Machine Learning
6. Applying Deep Learning to the Cache Replacement Problem
7. Branch Prediction as a Reinforcement Learning Problem: Why, How and Case Studies
8. TransforMAP: Transformer for Memory Access Prediction

Επικοινωνία: Κωστής Νίκας, knikas@cslab.ece.ntua.gr, 210-772-4159

1.2 Πρόβλεψη συμπεριφοράς κρυφών μνημών με τεχνικές επεξεργασίας φυσικής γλώσσας

Λόγω της αυξανόμενης διαφοράς ταχύτητας μεταξύ του επεξεργαστή και της κύριας μνήμης, οι κρυφές μνήμες (CPU caches) είναι κρίσιμες για την επίτευξη υψηλής ταχύτητας επεξεργασίας. Κάθε προσπέλαση μνήμης που δεν βρίσκει τα δεδομένα της στις κρυφές μνήμες στοιχίζει δεκάδες έως εκατοντάδες κύκλους μηχανής. Η ικανότητα να προβλέψουμε πως θα συμπεριφερθεί ένα πρόγραμμα για διαφορετικές ιεραρχίες κρυφής μνήμης είναι λοιπόν τρομερά χρήσιμη. Μας επιτρέπει να διαλέξουμε σε ποιον επεξεργαστή να τρέξουμε το πρόγραμμά μας ή πως να μοιράσουμε την κρυφή μνήμη ανάμεσα σε προγράμματα που τρέχουν παράλληλα στον ίδιο επεξεργαστή, ώστε να πετύχουμε την μέγιστη δυνατή ταχύτητα.

Σχεδόν όλες οι επιτυχημένες τεχνικές πρόβλεψης για κρυφές μνήμες επεξεργαστών περιορίζονται σε LRU και τυχαίες πολιτικές αντικατάστασης [1, 2, 3]. Η μόνη πρόσφατη τεχνική [4] που υποστηρίζει άλλες πολιτικές αντικατάστασης απαιτεί προηγούμενη γνώση της πολιτικής και αναλυτική περιγραφή των χαρακτηριστικών της. Το πρόβλημα όμως είναι ότι οι μοντέρνοι επεξεργαστές δεν χρησιμοποιούν LRU ή τυχαίες πολιτικές στο τελευταίο επίπεδο κρυφών μνημών και κατά κανόνα δεν έχουμε πληροφορίες για το τι χρησιμοποιούν.

Ο στόχος αυτής της διπλωματικής είναι να χρησιμοποιήσει τεχνικές μηχανικής μάθησης και επεξεργασίας φυσικής γλώσσας, παρόμοια με τα [5, 6], για να μοντελοποιήσει αυτόματα πως τα προγράμματα χρησιμοποιούν την κρυφή μνήμη. Με αυτό το μοντέλο, θα μπορούμε να προβλέψουμε το ποσοστό αποτυχιών κρυφής μνήμης ενός προγράμματος για κρυφές μνήμες διαφόρων μεγεθων (cache miss ratio curve) χωρίς καμία άλλη πληροφορία για τις παραμέτρους της μνήμης. Η πρόβλεψη θα βασίζεται αποκλειστικά και μόνο:

1. στο μοντέλο της κρυφής μνήμης,

2. στον κώδικα του προγράμματος, και

3. στην κατανομή των αποστάσεων επαναχρησιμοποίησης (reuse-distance histograms [3]), η οποία δείχνει την τοπικότητα των προσπελάσεων του προγράμματος.

Πιο συγκεκριμένα, ένα νευρωνικό δίκτυο τύπου transformer θα εκπαιδευτεί να εξαγάγει από τον κώδικα τις ιδιότητές του εκείνες που επηρεάζουν την συμπεριφορά του σχετικά με την μνήμη. Μετά, ενα δεύτερο νευρωνικό δίκτυο (CNN ή attention-based) θα μάθει να συσχετίζει τις ιδιότητες του κώδικα και τις αποστάσεις επαναχρησιμοποίησης με το ποσοστό αποτυχιών κρυφής μνήμης. Με αυτά τα δύο δίκτυα θα μπορούμε να προβλέπουμε την συμπεριφορά οποιουδήποτε προγράμματος.

Η διπλωματική μπορεί να υλοποιηθεί είτε σε πραγματικό επεξεργαστή είτε σε εξομοιωτή κρυφής μνήμης για να συγκεντρωθούν τα απαραίτητα δεδομένα. Η πρώτη επιλογή είναι πιο γρήγορη αλλά απαιτεί κώδικα χαμηλού επιπέδου για τον έλεγχο της κρυφής μνήμης και την συλλογή των στατιστικών. Η δεύτερη επιλογή είναι λίγο πιο χρονοβόρα άλλα πιο εύκολο να υλοποιηθεί και πιο ευέλικτη. Μετά την συλλογή των δεδομένων, ο σχεδιασμός και η εκπαίδευση των δύο νευρωνικών δικτύων θα είναι ο ίδιος και θα βασίζεται στα λογισμικά keras και tensorflow.

Η διπλωματική θα εκπονηθεί σε συνεργασία με τον Δρ. Παύλο Πετούμενο από το University of Manchester.

Σχετικά Μαθήματα: Προηγμένα Θέματα Αρχιτεκτονικής Υπολογιστών, Νευρωνικά Δίκτυα και Ευφυή Υπολογιστικά Συστήματα

Σχετική Βιβλιογραφία:

1. M. Sasongko, M.Chabbi, M. Bagheri Marzijarani, and D. Unat, ”ReuseTracker: Fast Yet Accurate Multicore Reuse Distance Analyzer”.

2. Q. Wang, X. Liu and M. Chabbi, ”Featherlight Reuse-Distance Measurement.

3. E. Berg and E. Hagersten, ”StatCache: a probabilistic approach to efficient and accurate data locality analysis”.

4. N. Beckmann and D. Sanchez, ”Modeling cache performance beyond LRU”

5. C. Cummins, P. Petoumenos, Z. Wang and H. Leather, ”End-to-End Deep Learning of Optimization Heuristics”.

6. M. Allamanis, E. T. Barr, P. Devanbu, and C. Sutton, ”A Survey of Machine Learning for Big Code and Naturalness”.

Επικοινωνία: Κωστής Νίκας, knikas@cslab.ece.ntua.gr, 210-772-4159

2 Επεκτάσεις Αρχιτεκτονικής

2.1 Αρχιτεκτονική υποστήριξη για βελτίωση του χρόνου εκκίνησης και εκτέλεσης containers

Πολλοί πάροχοι υποδομών υπολογιστικού νέφους (cloud computing) παρέχουν την δυνατότητα εκτέλεσης εφαρμογών χρησιμοποιώντας το περιβάλλον των containers. Ωστόσο, σε αυτό το περιβάλλον

εκτέλεσης υπάρχουν δύο αιτίες που επηρρεάζουν την απόδοση των εφαρμογών: (α) η αργή αρχικοποίηση (boot time due to cold starts) των containers (για παράδειγμα, serverless functions [2,3,4,5,1]), και (β) η αργή εκτέλεση των κλήσεων συστήματος (system calls) λόγω των επιπλέον ελέγχων που γίνονται [6] (για παράδειγμα, I/O intensive applications).

Σε αυτή τη διπλωματική εργασία, θα αναλύσουμε την εκτέλεση υπάρχοντων περιβάλλοντων εκτέλεσης containers (για παράδειγμα, Docker, gVisor, Firecracker) για να καταλάβουμε καλύτερα τις συνέπειες της εκτέλεσης εφρμογών σε docker, και θα αναγωνρίσουμε λειτουργίες που έχουν την δυνατότητα να επιταχυνθούν μέσω ειδικής υποστήριξης στο υλικό. Πιο συγκεκριμένα, θα επικεντρωθούμε σε εκείνα τα επίπεδα εικονικοποίησησς που επιτρέπουν την απομόνωση εφαρμογών (e.g. SecComp [6], Namespaces). Αυτά τα επίπεδα εικονικοποίησης χρησιμοποιουν διάφορους πίνακες που χρειάζονται να δημιουργούνται, να ενημερώνονται, και να χρησιμοποιούνται από τον πυρήνα του λειτουργικού συστήματος, για να παρέχεται απομόνωση των εφσρμογών. Μετά την ανάλυση, θα επικεντρωθούμε στην ανάπτυξη ειδικής υποστήριξης σε επίπεδο υλικού και αρχιτεκτονικής με σκοπό να μειώσουμε τον χρόνο αρχικοποίησης των containers και το κόστος εκτέλεσης των system calls.

Σχετικά Μαθήματα: Προηγμένα Θέματα Αρχιτεκτονικής Υπολογιστών, Εργαστήριο Λειτουργικών Συστημάτων

Σχετική Βιβλιογραφία:

1. Architectural Implications of Function-as-a-Service Computing, MICRO 2019
2. Catalyzer: Sub-millisecond Startup for Serverless Computing with Initialization-less Booting, ASPLOS 2020
3. SOCK: Rapid Task Provisioning with Serverless-Optimized Containers
4. SAND: Towards High-Performance Serverless Computing
5. Serverless in the Wild: Characterizing and Optimizing the Serverless Workload at a Large Cloud Provider, ATC 2020
6. Draco: Architectural and Operating System Support for System Call, Security MICRO 2020
7. BabelFish: Fusing Address Translations for Containers, ISCA 2020

Επικοινωνία: Κωστής Νίκας, knikas@cslab.ece.ntua.gr, 210-772-4159

3 Αρχιτεκτονική RISC-V

Η RISC-V αρχιτεκτονική είναι μια ανοικτή και επεκτάσιμη αρχιτεκτονική συνόλου εντολών που ξεκίνησε να αναπτύσσεται στο Πανεπιστήμιο του Berkeley το 2010 και από το 2016 λαμβάνει διεθνή προσοχή τόσο από τον ακαδημαϊκό χώρο όσο και από τον χώρο της βιομηχανίας, με κατάλληλη υποστήριξη σε όλα τα επίπεδα της υπολογιστικής στοίβας (υλικό, λειτουργικό σύστημα, βιβλιοθήκες, μεταγλωττιστές, κτλ). Ως ανοικτή και επεκτάσιμη αρχιτεκτονική προσφέρεται για την έρευνα σε λειτουργικές επεκτάσεις, ενώ πολλές υλοποιήσεις ανοικτού κώδικα είναι άμεσα διαθέσιμες, άλλες απλούστερες με γραμμική in-order pipeline και άλλες μεγαλύτερων επιδόσεων με πυρήνα εκτέλεσης εντολών εκτός σειράς (out-of-order).

3.1 Μελέτη περιβάλλοντος ανάπτυξης και υλοποίηση επιταχυντών σε RISC-V αρχιτεκτονική

Στόχος της παρούσας διπλωματικής εργασίας είναι η μελέτη του περιβάλλοντος ανάπτυξης υλικού του Rocket Chip Generator που αναπτύσσεται από το Πανεπιστήμιο του Berkeley και θα εστιάσουμε στην ανάπτυξη επιταχυντών σε RISC-V αρχιτεκτονικές χρησιμοποιώντας το framework του Rocket Custom Coprocessor (RoCC).

Σχετικά Μαθήματα: Προηγμένα Θέματα Αρχιτεκτονικής Υπολογιστών

Σχετική Βιβλιογραφία:

1. A Hardware Accelerator for Protocol Buffers, MICRO 2021
2. A Hardware Accelerator for Tracing Garbage Collection
3. `https://en.wikipedia.org/wiki/RISC-V`
4. `https://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-17.pdf`

Επικοινωνία: Κωστής Νίκας, knikas@cslab.ece.ntua.gr, 210-772-4159

3.2 Μελέτη Fuzzing Τεχνικών για την αρχιτεκτονική RISC-V

Χρησιμοποιώντας Fuzzing πρακτικά εκτελούμε ένα τυχαίο instruction stream σε κάποιον επεξεργαστή, με σκοπό να ανακαλύψουμε πιθανά μικροαρχιτεκτονικά σφάλματα. Στην παρούσα διπλωματική θα μελετήσετε/υλοποιήσετε διαφορετικές μεθόδους fuzzing για την αρχιτετονική RISC-V με σκοπό την βελτίωση είτε της αρκίβειας της μεθόδου, είτε την μείωση της χρονικής διάρκειας που απαιτείται για την εύρεση σφαλμάτων την μικροαρχιτεκτονική.

Σχετικά Μαθήματα: Προηγμένα Θέματα Αρχιτεκτονικής Υπολογιστών, Εργαστήριο Λειτουργικών Συστημάτων

Σχετική Βιβλιογραφία:

[1] Cascade: CPU Fuzzing via Intricate Program Generation,
`https://comsec.ethz.ch/research/hardware-design-security/cascade-cpu-fuzzing-via-intricate-prog`

[2] Revizor - a fuzzer to search for microarchitectural leaks in CPUs,
`https://github.com/microsoft/sca-fuzzer`

Επικοινωνία: Νίκος Χ. Παπαδόπουλος, ncpapad@cslab.ece.ntua.gr
Φοίβος Ηλιάδης, filiadis@cslab.ece.ntua.gr

3.3 Αξιόπιστα Περιβάλλοντα Εκτέλεσης

Ένα αξιόπιστο περιβάλλον εκτέλεσης (Trusted Execution Environment - TEE) είναι μία εναλλακτική λειτουργία του επεξεργαστή η οποία προσφέρει υψηλότερα επίπεδα ασφάλειας. Όταν ένας επεξεργαστής βρίσκεται σε λειτουργία ασφαλούς εκτέλεσης εγγυάται ότι ο κώδικας και τα δεδομένα μίας εφαρμογής που τρέχει εντός του TEE διατηρούνται απόρρητα (confidentiality), και προστατεύεται η ακεραιότητα (integrity) τους. Ένα TEE είναι ένα απομονωμένο περιβάλλον εκτέλεσης το οποίο παρέχει

επιπλέον δικλείδες ασφαλείας σε σχέση με την "απλή" εκτέλεση του επεξεργαστή. Ο χώρος εκτέλεσης ενός TEE -ονομάζεται enclave- προσφέρει υψηλότερη ασφάλεια στις εφαρμογές από την τυπική ασφάλεια που προσφέρει ένα λειτουργικό σύστημα, χωρίς όμως να χάνει τα προσόντα του λειτουργικού συστήματος όπως τα system calls, I/O, διαδιεργασιακή επικοινωνία κλπ.

Σχετικά Μαθήματα: Προηγμένα Θέματα Αρχιτεκτονικής Υπολογιστών, Εργαστήριο Λειτουργικών Συστημάτων

Σχετική Βιβλιογραφία:

[1] Trusted Execution Environment (Wikipedia Page),
`https://en.wikipedia.org/wiki/Trusted_execution_environment`

[2] Confidential VM Extension for Confidential Computing on RISC-V Platforms,
`https://github.com/riscv-non-isa/riscv-ap-tee/blob/main/specification/riscv-cove.pdf`

[3] Keystone: An Open Framework for Architecting Trusted Execution Environments,
`http://n.ethz.ch/~sshivaji/publications/keystone_eurosys20.pdf`

Επικοινωνία: Νίκος Χ. Παπαδόπουλος, ncpapad@cslab.ece.ntua.gr
Φοίβος Ηλιάδης, filiadis@cslab.ece.ntua.gr

3.4 Μελέτη προηγμένων σχημάτων διαχείρισης εικονικής μνήμης στη αρχιτεκτονική RISC-V

Σε αυτή την ενότητα διπλωματικών εργασιών μπορείτε να επιλέξετε είτε την υλοποίηση των σχημάτων σε πραγματικό hardware (Rocket Chip Generator, CVA6), είτε σε κάποιον simulator (gem5).

- Ο Rocket Chip Generator [2] είναι ένας ανοιχτού κώδικα System-on-Chip (SoC) Generator που παράγει παραμετροποιήσιμα RISC-V SoCs. Είναι υλοποιημένος στην γλώσσα Chisel [3], η οποία καθιστά εύκολη την περιγραφή πολύπλοκων και παραμετροποιήσιμων γεννητριών για επεξεργαστικούς πυρήνες, κρυφές μνήμες και δικτύων διασύνδεσης εντός του SoC
- Η CVA6 [3] είναι μία υλοποίηση του RISC-V ISA σε SystemVerilog. Η ανάπτυξη της CVA6 ξεκίνησε από το ETH Zurich, και τώρα γίνεται maintain από το OpenHW Group
- ο gem5 [4] είναι ένας simulator που χρησιμοποιείται στην έρευνα Αρχιτεκτονικής Υπολογιστών. Υποστηρίζει διαφορετικά ISAs, ένα από αυτά αποτελεί ο RISC-V

Στα παρακάτω θέματα θα χρησιμοποιήσουμε εργαλεία ανοικτού κώδικα για την ανάπτυξη υλικού, επιβεβαίωσης ορθής λειτουργίας του, καθώς και FPGAs (Field Programmable Gate Arrays) για την μελέτη επίδοσης του υλικού που σχεδιάσαμε.

Σχετική Βιβλιογραφία:

[1] RISC-V Technical Specifications,
`https://riscv.org/technical/specifications/`

[2] Rocket Chip Generator,
`https://github.com/chipsalliance/rocket-chip/`

[3] CVA6,
`https://github.com/openhwgroup/cva6`

[4] gem5,
https://www.gem5.org/

Επικοινωνία: Νίκος Χ. Παπαδόπουλος, ncpapad@cslab.ece.ntua.gr

3.4.1 Advanced MMU Caching Techniques

Σε πολλά σύγχρονα benchmarks/workloads, η μετάφραση των εικονικών διευθύνσεων μπορεί να επιβαρύνει αισθητά την επίδοση και την ενεργειακή απόδοση του υπολογιστικού συστήματος, λόγω των αστοχιών TLB. Αναλόγως της αρχιτεκτονικής του πίνακα σελίδων, απαιτούνται 3-4 προσβάσεις στην μνήμη για την μετάφραση της εικονικής-σε-φυσική διεύθυνση. Συγκεκριμένα, σε περιβάλλοντα εικονοποίησης ο αριθμός προσβάσεων στην μνήμη μπορεί να φτάσει τις 24. Μία πρόταση στο παραπάνω πρόβλημα είναι η χρήση μεγαλύτερης χωρητικότητας -ή μεγαλύτερης συσχετιστικότητας- TLB. Όμως, το TLB βρίσκεται στο critical path του επεξεργαστή με αποτέλεσμα να επηρεάζει τον χρονισμό του: δημιουργείται ένα trade-off μεταξύ μεγέθους TLB (χαμηλότερος χρονισμός CPU) και αστοχιών TLB (χαμηλότερη επίδοση). Στην σύγχρονη βιβλιογραφία προτείνονται διάφορες αρχιτεκτονικές για caching εικονικής μνήμης, όπως τα Coalesced TLBs [1], Clustered TLBs [2], Hybrid TLB Coalescing [3], Direct Segments [4], Redundant Memory Mappings [5] και άλλα.

Σχετικά Μαθήματα: Προηγμένα Θέματα Αρχιτεκτονικής Υπολογιστών

Σχετική Βιβλιογραφία:

[1] CoLT: Coalesced Large-Reach TLBs,
https://ieeexplore.ieee.org/document/6493625

[2] Increasing TLB Reach by Exploiting Clustering in Page Translations,
http://www.cs.yale.edu/homes/abhishek/binhpham-hpca14.pdf

[3] Hybrid TLB Coalescing: Improving TLB Translation Coverage under Diverse Fragmented Memory Allocations,
https://iamchanghyunpark.github.io/papers/htc-isca2017.pdf

[4] Efficient Virtual Memory for Big Memory Servers,
https://research.cs.wisc.edu/multifacet/papers/isca13_direct_segment.pdf

[5] Redundant Memory Mappings for Fast Access to Large Memories,
http://www.cslab.ece.ntua.gr/~vkarakos/papers/isca15_redundant_memory_mappings.pdf

[6] A Configurable TLB Hierarchy for the RISC-V Architecture,
https://cgi.di.uoa.gr/~vkarakos/papers/fpl20_configurable_tlb_riscv.pdf

Επικοινωνία: Νίκος Χ. Παπαδόπουλος, ncpapad@cslab.ece.ntua.gr

3.4.2 Enabling TLB Prefetching for RISC-V processors

Μία άλλη τεχνική για την βελτίωση της επίδοσης του συστήματος εικονικής μνήμης είναι το prefetching [1]. Η τεχνική αυτή βασίζεται στην εικασία ότι φέρνοντας δεδομένα από την κύρια μνήμη στην κρυφή μνήμη πριν χρειαστούν, θα μηδενιστεί η αναμονή σε μελλοντική πρόσβαση στα δεδομένα αυτά. Στην διπλωματική αυτή, θα μελετηθεί η ανάπτυξη prefetcher για το TLB [2, 3, 4], και θα εξεταστεί η επίδοση του χρησιμοποιώντας benchmarking suites όπως το SPEC2006/SPEC2017.

Σχετικά Μαθήματα: Προηγμένα Θέματα Αρχιτεκτονικής Υπολογιστών

Σχετική Βιβλιογραφία:

[1] Cache Prefetching (Wikipedia Page)
https://en.wikipedia.org/wiki/Cache_prefetching

[2] Exploiting Page Table Locality for Agile TLB Prefetching,
https://ieeexplore.ieee.org/document/9499825

[3] Recency-Based TLB Preloading,
https://courses.cs.washington.edu/courses/cse590g/00au/p117-saulsbury.pdf

[4] Going the Distance for TLB Prefetching: An Application-driven Study,
http://www.cse.psu.edu/~axs53/csl/papers/isca02.pdf

[5] Inter-core cooperative TLB for chip multiprocessors,
https://dl.acm.org/doi/abs/10.1145/1735970.1736060

Επικοινωνία: Νίκος Χ. Παπαδόπουλος, ncpapad@cslab.ece.ntua.gr

3.4.3 Enabling Configurable Page Table Walk Caches for the Rocket Chip Generator

Η μονάδα που αναλαμβάνει την μετάφραση μιας εικονικής διεύθυνσης σε φυσική ονομάζεται Page Table Walker (PTW) και είναι συνήθως υλοποιημένη στο υλικό για την ταχύτερη μετάφραση των εικονικών διευθύνσεων. Η δομή δεδομένων που χρησιμοποιείται για το mapping των εικονικών σε φυσικές διευθύνσεις ονομάζεται πίνακας σελίδων (page table) [1] και αναλόγως την αρχιτεκτονική, αποτελείται από 3 ή 4 επίπεδα. Στον Rocket Chip Generator (RCG) ο πίνακας σελίδων αποτελείται από 3 επιπεδα για το σχήμα εικονικής μνήμης Sv39 [2]. Σε περίπτωση αστοχίας TLB, η μονάδα PTW πρέπει να κάνει επομένως 3 προσβάσεις στον πίνακα σελίδων ώστε να μεταφράσει την ζητούμενη εικονική διεύθυνση σε φυσική. Για να αποφευχθούν οι κοστοβόρες προσβάσεις στην μνήμη, στον RCG έχει υλοποιηθεί μία μικρή PTW Cache [3] η οποία αποθηκεύει το mapping των πρώτων 2 επιπέδων (το mapping του 3ου επιπέδου αποθηκεύεται στο TLB). Αντικείμενο της διπλωματικής αυτής είναι η παραμετροποίηση της PTW Cache του RCG, η μελέτη υλοποίησης πιο προηγμένων σχημάτων PTW Cache [3], και η εξέταση της επίδοσης τους χρησιμοποιώντας benchmarking suites όπως το SPEC2006/SPEC2017.

Σχετικά Μαθήματα: Προηγμένα Θέματα Αρχιτεκτονικής Υπολογιστών

Σχετική Βιβλιογραφία:

[1] Page Table (Wikipedia Page),
https://en.wikipedia.org/wiki/Page_table

[2] RISC-V Page-Based 39-bit Virtual-Memory System, pages 62-64,
https://riscv.org//wp-content/uploads/2017/05/riscv-privileged-v1.10.pdf

[3] Translation caching: skip, don't walk (the page table),
https://dl.acm.org/doi/10.1145/1816038.1815970

Επικοινωνία: Νίκος Χ. Παπαδόπουλος, ncpapad@cslab.ece.ntua.gr

3.5 Επεκτάσεις του RISC-V για near/in memory accelerators

Η εργασία αυτή αφορά αφενός την δημιουργία ενός μικρού πυρήνα RISC-V ο οποίος θα είναι ο δομικός λίθος για επεξεργασία κοντά στην μνήμη για επεξεργασία μεγάλων δεδομένων (η αρχιτεκτονική αναφοράς είναι το Modrian Data Engine). Έτσι οι βασικές συναρτήσεις θα είναι ερωτήσεις βάσεων δεδομένων στις οποίες τα δεδομένα βρίσκονται στην μνήμη. Το σύστημα μνήμης θα αποτελείται από ένα (η περισσότερα) HMC modules. Συγκεκριμένα, τα βήματα της εργασίας είναι (α) η επιλογή και επέκταση ενός υπάρχοντος πυρήνα RISC-V με εντολές vector (β) ο προγραμματισμός των βασικών λειτουργιών και η επιβεβαίωση ορθής λειτουργίας με προσομοιώσεις, (γ) η ολοκλήρωσή του επεξεργαστή στο περιβάλλον FPGA+HMC της Micron και (δ) η αξιολόγηση του συνολικού συστήματος.

Σχετικά Μαθήματα: Προηγμένα Θέματα Αρχιτεκτονικής Υπολογιστών

Σχετική Βιβλιογραφία:

1. `https://en.wikipedia.org/wiki/RISC-V`
2. `https://en.wikipedia.org/wiki/Vector_processor`
3. `https://en.wikipedia.org/wiki/Hybrid_Memory_Cube`
4. `https://pure.tue.nl/ws/files/100178113/gagan2018dsd.pdf`
5. `https://arxiv.org/pdf/1908.02640.pdf`
6. The Mondrian Data Engine: `https://dl.acm.org/citation.cfm?id=3080233`
7. `https://www.sigarch.org/simd-instructions-considered-harmful/`
8. `https://www.youtube.com/watch?v=GzZ-8bHsD5s`

Επικοινωνία: Διονύσιος Πνευματικάτος, pnevmati@cslab.ece.ntua.gr, 6944763171
Μηλιάδης Παναγιώτης, pmiliad@cslab.ece.ntua.gr

4 Αποδοτική απεικόνιση σε FPGAs

Στις μέρες μας, η χρήση των επαναπρογραμματιζόμενων αρχιτεκτονικών (FPGAs) αποτελεί σημαντική εναλλακτική πρόταση, καθώς προσφέρει τη σχεδίαση υλικού για την εκτέλεση συγκεκριμένων εφαρμογών (application-specific), με σκοπό τη βελτιστοποίηση και την επιτάχυνση του χρόνου εκτέλεσης. Αν και μπορούν να απεικονίσουν όμως οποιαδήποτε σχεδίαση υλικού, οι FPGAs έχουν ιδιαιτερότητες και «προτιμήσεις».

4.1 Αποδοτική απεικόνιση επεξεργαστών RISC-V σε FPGA

Η εργασία αυτή αφορά αφενός την συγκριτική μελέτη βασικών υπαρχόντων υλοποιήσεων RISC-V ως προς το κόστος και τις επιδόσεις τους όταν υλοποιούνται με διαφορετικές FPGA, και αφετέρου την παραμετροποίηση των εσωτερικών δομών (η την αντικατάστασή τους με άλλες ισοδύναμες) ώστε η συνολική σχεδίαση να είναι περισσότεραο «φιλική» προς τις FPGA. Η εργασία συνδυάζει προσομοιώσεις για την μέτρηση επιδόσεων σε αρχιτεκτονικό επίπεδο και απεικόνιση των αρχιτεκτονικών σε FPGA με εργαλεία CAD.

Σχετικά Μαθήματα: Προηγμένα Θέματα Αρχιτεκτονικής Υπολογιστών

Σχετική Βιβλιογραφία:

1. https://en.wikipedia.org/wiki/RISC-V
2. https://github.com/pulp-platform/riscv
3. https://tspace.library.utoronto.ca/handle/1807/80713
4. https://github.com/riscv-boom/riscv-boom

Επικοινωνία: Διονύσιος Πνευματικάτος, pnevmati@cslab.ece.ntua.gr, 6944763171

4.2 Υλοποίηση Compiler για τη διάσπαση σχεδίασης σε πολλαπλά μικρότερα bitstreams και την αποδοτική χαρτογράφιση τους σε FPGAs

Στη σύγχρονη εποχή οι FPGAs ενσωματώνονται σε συστήματα cloud, με πιο χαρακτηριστικά παραδείγματα το F1 instance της Amazon και της Alibaba. Κάθε χρήστης μπορεί να νοικιάσει ένα σύστημα στο οποίο ενσωματόνονται FPGAs, προκειμένου να ενσωματώσει και να τρέξει τη σχεδίαση του ή τον υπολογιστικό του πυρήνα (Platform as a Service). Όμως, σπανίως ένας χρήστης αξιοποιεί πλήρως την έκταση μιας FPGA, αφήνοντας ένα μεγάλο μέρος των πόρων ανεκμετάλλευτο. Σκοπός στο cloud, είναι η πλήρης εκμετάλλευση της έκτασης ενός FPGA, δηλαδή την υπολογιστική του δύναμη, μοιράζοντας τους διαθέσιμους πόρους μεταξύ πολλών χρηστών (multi-tenant) ταυτόχρονα. Έτσι, προτείνεται η δημιουργία μικρότερων fixed-slots εντός της FPGA, όπου η σχεδίαση κάθε χρήστη θα χαρτογραφείται στο νέο περιορισμένο χώρο. Δυστυχώς, πολλές σχεδιάσεις δεν ταιριάζουν στο νέο περιορισμένο χώρο που διατίθεται από τα νέα συστήματα των παρόχων, με αποτέλεσμα την ανάγκη διάσπασης της σχεδίασης σε μικρότερα κομμάτια, όπου κάθε κομμάτι θα χαρτογραφείται σε ένα fixed-slot, ικανοποιώντας τους χωρικούς περιορισμούς που επιβάλλονται. Σκοπός της διπλωματικής εργασίας είναι η ανάπτυξη ενός compiler όπου θα δέχεται σαν είσοδο την σχεδίαση ενός χρήστη σε επίπεδο HLS ή HDL, και θα διασπά τη σχεδίαση σε μικρότερα bitstreams, με σκοπό την αποδοτική χαρτογράφιση των επιμέρους κομματιών στο νέο περιορισμένο χώρο που διατίθεται.

Σχετικά Μαθήματα: Προηγμένα Θέματα Αρχιτεκτονικής Υπολογιστών

Σχετική Βιβλιογραφία:

1. Yue Zha and Jing Li. 2020. Virtualizing FPGAs in the Cloud. In Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '20). Association for Computing Machinery, New York, NY, USA, 845–858. DOI:https://doi.org/10.1145/3373376.3378491
2. Y. Zha and J. Li, "Hetero-ViTAL: A Virtualization Stack for Heterogeneous FPGA Clusters," 2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA), 2021, pp. 470-483, doi: 10.1109/ISCA52012.2021.00044.
3. https://www.rapidwright.io/
4. https://github.com/Xilinx/RapidWright

Επικοινωνία: Μηλιάδης Παναγιώτης, pmiliad@cslab.ece.ntua.gr
Διονύσιος Πνευματικάτος, pnevmati@cslab.ece.ntua.gr, 6944763171

II. Λειτουργικά Συστήματα - Εικονικές Μηχανές

5 Μελέτη της αλληλεπίδρασης των μηχανισμών δέσμευσης (εικονικής) μνήμης χώρου χρήστη με τους μηχανισμού δέσμευσης (φυσικής) μνήμης χώρου πυρήνα

Η δυναμική δέσμευση μνήμης (dynamic memory allocation) αποτελεί μια από τις βασικές λειτουργίες που προσφέρει μια γλώσσα προγραμματισμού στους χρήστες και μπορεί να επηρεάσει σε μεγάλο βαθμό την απόδοση των προγραμμάτων που γράφονται στην συγκεκριμένη γλώσσα. Για προγράμματα σε C / C++, η διεπαφή για την δέσμευση και αποδέσμευση μνήμης περιλαμβάνει της συναρτήσεις malloc() και free(). Οι βιβλιοθήκες δέσμευσης μνήμης (memory allocators) που υλοποιούν αυτή την διεπαφή [1], χρησιμοποιούν τους μηχανισμούς του λειτουργικού συστήματος για να δεσμεύσουν και να αποδεσμεύσουν μνήμη (brk, mmap σε συστήματα GNU / Linux) και βασικός στόχους τους είναι η αποδοτική αξιοποίηση του δεσμευμένου χώρου και η ελάχιστη δυνατή επιβάρυνση στον χρόνο εκτέλεσης.

Ο τρόπος δέσμευσης της εικονικής μνήμης από το λειτουργικό σύστημα, π.χ. τα μεγέθη και η ευθυγράμμιση (alignment) της μνήμης ενδέχεται να επηρεάσουν την απόδοση του προγράμματος. Για παράδειγμα, η βιβλιοθήκη δέσμευσης μνήμης μπορεί να ζητήσει από το λειτουργικό σύστημα τη χρησιμοποίηση μεγάλων σελίδων (huge pages) [2][3][7], με σκοπό την ελαχιστοποίηση του κόστους της μετάφρασης εικονικών διευθύνσεων ή να συνεργαστεί με το λειτουργικό σύστημα για την πιο αποδοτική δέσμευση μνήμης σε μηχανήματα με ανομοιόμορφο κόστος πρόσβασης στη μνήμη (NUMA) [8].

Σκοπός αυτής της διπλωματικής είναι να διερευνηθεί, για συστήματα GNU / Linux, αν και σε τι βαθμό υποστηρίζουν τη δέσμευση μνήμης με μεγάλες σελίδες (transparent hugepages [4], hugetlbfs [5]), σε αρχιτεκτονικές x86 και ARMv8, οι πιο δημοφιλείς βιβλιοθήκες δυναμικής δέσμευσης μνήμης C και να αξιολογηθεί η επίδρασή τους στην απόδοση των allocators και των τελικών προγραμμάτων. Τέλος, θα διερευνηθεί η δυνατότητα συνεργασίας των userspace allocators, ειδικά όσων έχουν υποστήριξη για huge page allocations, με το μηχανισμό δέσμευσης φυσικά συνεχόμενης μνήμης σε χώρο πυρήνα [6] και πώς αυτό επηρεάζει το κόστος της μετάφρασης εικονικών διευθύνσεων των τελικών προγραμμάτων.

Σχετικά Μαθήματα: Λειτουργικά Συστήματα, Εργαστήριο Λειτουργικών Συστημάτων, Προηγμένα Θέματα Αρχιτεκτονικής Υπολογιστών

Σχετική Βιβλιογραφία:

1. https://en.wikipedia.org/wiki/C_dynamic_memory_allocation#Implementations
2. https://github.com/libhugetlbfs/libhugetlbfs/issues/52#issuecomment-995203319
3. https://google.github.io/tcmalloc/temeraire.html
4. https://www.kernel.org/doc/Documentation/mm/transhuge.rst
5. https://www.kernel.org/doc/Documentation/vm/hugetlbpage.txt
6. https://github.com/cslab-ntua/contiguity-isca2020
7. https://mosalloc.cs.technion.ac.il/
8. https://www.kernel.org/doc/Documentation/vm/numa.rst

Επικοινωνία: Στράτος Ψωμαδάκης, psomas@cslab.ece.ntua.gr

6 Ανάλυση και βελτιστοποίηση του μηχανισμού και των πολιτικών διαχείρισης μνήμης Automatic NUMA balancing για αρχιτεκτονικές με ανομοιόμορφη πρόσβαση μνήμης

Ένας από τους τρόπους αύξησης της κλιμακωσίμοτητας των υπολογιστικών συστημάτων που έχουν πολλαπλούς επεξεργαστές στο ίδιο σύστημα είναι μέσω των Αρχιτεκτονικών με Ανομοιόμορφη Πρόσβαση Μνήμης ή αλλιώς Non Uniform Memory Access (NUMA) systems. Η NUMA αρχιτεκτονική αποτελεί έναν σχεδιασμό συστήματος μνήμης πολυεπεξεργαστικών υπολογιστικών συστημάτων, στα οποία ο χρόνος πρόσβασης της κύριας μνήμης (RAM) εξαρτάται από την απόσταση της θέσης μνήμης που προσπελαύνει ο επεξεργαστής. Σε μία NUMA αρχιτεκτονική, ένας επεξεργαστής έχει γρηγορότερη πρόσβαση σε μία τοπική μνήμη από ότι σε μία απομακρυσμένη, η οποία όμως είναι τοπική για κάποιον άλλον επεξεργαστή. Το λειτουργικό σύστημα Linux παρέχει τον μηχανισμό Automatic NUMA Balancing ο οποίος μεταφέρει δυναμικά δεδομένα ανάμεσα σε κόμβους μνήμης για να μειώσει τον χρόνο πρόσβασης στην μνήμη. Στόχος της παρούσας διπλωματικής είναι η μελέτη, η ανάλυση, και η βελτιστοποίηση του μηχανισμού Automatic NUMA balancing του Linux, καθώς επίσης και η υλοποίηση αποδοτικών πολιτικών χρήσης του.

Σχετικά Μαθήματα: Εργαστήριο Λειτουργικών Συστημάτων

Σχετική Βιβλιογραφία:

1. Automatic Non-Uniform Memory Access (NUMA) Balancing, SUSE
2. Automatic NUMA Balancing, Red Hat
3. NUMA Memory Architectures and the Linux Memory System, Red Hat

Επικοινωνία: Στράτος Ψωμαδάκης, psomas@cslab.ece.ntua.gr

III. Παράλληλα Συστήματα

7 Πολλαπλασιασμός αραιού πίνακα με διάνυσμα (SpMV)

Ο υπολογιστικός πυρήνας του πολλαπλασιασμού αραιού πίνακα με διάνυσμα (SpMV) χρησιμοποιείται ευρέως σε παράλληλες εφαρμογές μεγάλης κλίμακας. Ωστόσο, λόγω της αλγοριθμικής του φύσης, δεν αξιοποιεί επαρκώς την υπολογιστική ισχύ των σύγχρονων επεξεργαστών. Οι παρακάτω εργασίες εστιάζουν στην βελτιστοποίηση του με σε διαφορετικές αρχιτεκτονικές με τη χρήση των κατάλληλων προγραμματιστικών μοντέλων.

7.1 Βελτιστοποίηση του υπολογιστικού πυρήνα πολλαπλασιασμού αραιού πίνακα με διάνυσμα (SpMV) σε FPGAs

Στην παρούσα διπλωματική εργασία, θα μελετηθεί η υλοποίηση και η βελτιστοποίηση του συγκεκριμένου υπολογιστικού πυρήνα σε επαναδιαμορφούμενες αρχιτεκτονικές (FPGAs), που επιτρέπουν στον προγραμματιστή τη δημιουργία υλικού εξειδικευμένου στην εφαρμογή (application-specific). Συγκεκριμένα, θα μελετηθεί η επίδοση βασικών υλοποιήσεων του SpMV για FPGAs με τη χρήση του προγραμματιστικού μοντέλου της OpenCL ή άλλων προγραμματιστικών μοντέλων υψηλού επιπέδου, καθώς και εναλλακτικά σχήματα αποθήκευσης αραιών πινάκων, και θα εφαρμοστούν τεχνικές βελτιστοποίησης του υπολογιστικού πυρήνα με στόχο την επίτευξη της μέγιστης δυνατής επίδοσης στις συγκεκριμένες αρχιτεκτονικές.

Σχετικά Μαθήματα: Συστήματα Παράλληλης Επεξεργασίας, Ψηφιακά Συστήματα VLSI

Επικοινωνία: Παναγιώτης Μπάκος, pmpakos@cslab.ece.ntua.gr
Γεώργιος Γκούμας, goumas@cslab.ece.ntua.gr, 210-772-2402

7.2 Βελτιστοποίηση του υπολογιστικού πυρήνα πολλαπλασιασμού αραιού πίνακα με διάνυσμα (SpMV)

Μελέτη παράλληλων υλοποιήσεων του SpMV πυρήνα σε αρχιτεκτονικές CPU (Intel, AMD, ARM) ή GPU (Nvidia). Τα προγραμματιστικά μοντέλα που μπορούν να χρησιμοποιηθούν είναι το OpenMP για τις CPUs και η CUDA για τις GPUs, αλλά και όποια άλλη προτίμηση υπάρχει. Η πραγματοποίηση της διπλωματικής μπορεί να έχει διάφορες προσεγγίσεις:

- βελτίωση ήδη υπάρχουσας υλοποίησης του πυρήνα
- μελέτη της αναπαράστασης της δομής του αραιού πίνακα (συντεταγμένες των μη μηδενικών τιμών)
- συμπίεση των τιμών του πίνακα και μελέτη της επίδρασης lossy μεθόδων συμπίεσης

Σχετικά Μαθήματα: Συστήματα Παράλληλης Επεξεργασίας

Επικοινωνία: Δημήτριος Γαλανόπουλος, dgal@cslab.ece.ntua.gr
Γεώργιος Γκούμας, goumas@cslab.ece.ntua.gr, 210-772-2402

8 Αποδοτική χρήση και προγραμματισμός GPGPU

Οι σύγχρονες μονάδες επεξεργασίας γραφικών ή κάρτες γραφικών (GPUs) έχουν εξελιχθεί από το να είναι χρήσιμες μόνο για συγκεκριμένες λειτουργίες, σε ισχυρά εργαλεία γενικής χρήσης (GPGPUs) ικανά να υποστηρίξουν μια πολύ μεγαλύτερη ποικιλία προβλημάτων, παρέχοντας μια ταχύτερη και πιο ενεργειακά αποδοτική εναλλακτική λύση σε σχέση με τους κανονικούς επεξεργαστές (CPUs).

8.1 Βελτιστοποίηση πυρήνων γραμμικής άλγεβρας για συστήματα πολλαπλων GPU

Οι BLAS (Basic Linear algebra subprograms) αποτελούν μια συγκεκριμένη ομάδα απο ρουτίνες γραμικής άλγεβρας, που χρησιμοποιούνται σε πολλά επιστημονικά προβληματά. Σε συστήματα με πολλάπλες GPUs μια ολοκληρωμένη βιβλιοθήκη BLAS πρέπει να περιλαμβάνει και μεταφορές δεδομένων, χρονοδρομολόγηση υποπροβλημάτων και λήψη αποφάσεων για την επίτευξη καλής απόδοσης. Ο σκοπός της διπλωματικής αυτης είναι η εξοικίωση με τέτοιες βιβλιοθήκες, και η επέκταση τους η μελέτη πάνω σε αυτές. Ορισμένα πιο συγκεκριμένα θέματα θα μπορούσαν να περιλαμβάνουν:

- Μελέτη και ανάπτυξη των μοντέλων πρόβλεψης επίδοσης που χρησιμοποιούν για αυτόματη βελτιστοποίηση.
- Επέκταση των υποστηριζόμενων ρουτίνων σε βιβλιοθήκη BLAS.
- Μελέτη και βελτιστοποίηση των βιβλιοθηκών σε περιπτώσεις με ετερογενή χαρακτηριστικά.

Σχετικά Μαθήματα: Συστήματα Παράλληλης Επεξεργασίας

Σχετική Βιβλιογραφία:

1. https://docs.nvidia.com/cuda/cublas/index.html
2. https://ieeexplore.ieee.org/document/9408195
3. https://dl.acm.org/doi/abs/10.1145/3624569
4. https://arxiv.org/abs/1510.05041

Επικοινωνία: Αναστασιάδης Πέτρος, panastas@cslab.ece.ntua.gr

9 Αποδοτική χρήση και προγραμματισμός FPGA

Τα τελευταία χρόνια, οι ανάγκες των εφαρμογών για υψηλή επίδοση δεν καλύπτονται μόνο από συμβατικούς επεξεργαστές, αλλά από συνδυασμό CPU με επιταχυντές ειδικού σκοπού, όπως είναι οι GPU και τα FPGA, σε ετερογενή συστήματα υψηλών επιδόσεων. Πιο συγκεκριμένα, τα FPGA προτείνονται για ανάπτυξη εφαρμογών υψηλών επιδόσεων, καθώς ο στόχος πλέον δεν είναι μόνο η βελτίωση της επίδοσης, αλλά και η ενεργειακή αποδοτικότητα των συστημάτων αυτών. Προς αυτήν την κατεύθυνση έχουν αναπτυχθεί εργαλεία σύνθεσης υψηλού επιπέδου (High Level Synthesis), με σκοπό την επιτάχυνση και αυτοματοποίηση της διαδικασίας σχεδιασμού και προγραμματισμού υπολογιστικών εφαρμογών.

9.1 Αξιολόγηση των εφαρμογών της σουίτας Rodinia σε accelerator FPGAs

Ο σκοπός της διπλωματικής αυτής είναι η εξοικείωση με το περιβάλλον επιτάχυνσης εφαρμογών της Xilinx (Vitis) και η μεταφορά/porting των συγκεκριμένων εφαρμογών για εκτέλεση σε FPGA επιταχυντές. Ορισμένα ενδεικτικά βήματα της πορείας της διπλωματικής είναι:

- Μελέτη τεχνικών για επιτάχυνση εφαρμογών σε HLS περιβάλλον
- Επέκταση του υπάρχοντος benchmark suite για πιο σύγχρονα FPGA
- Σύγκριση της επίδοσης και της ενεργειακής αποδοτικότητας διαφορετικών FPGAs μεταξύ τους, αλλά και με σύγχρονες GPUs.

Σχετικά Μαθήματα: Συστήματα Παράλληλης Επεξεργασίας

Σχετική Βιβλιογραφία:

1. S. Che et al., "Rodinia: A benchmark suite for heterogeneous computing", 2009 IEEE International Symposium on Workload Characterization (IISWC), 2009.
2. Rodinia benchmark suite for CPUs/GPUs
3. Xilinx Accelerator FPGAs
4. Xilinx Vitis application acceleration guide
5. J. Cong et al., "Understanding Performance Differences of FPGAs and GPUs", 2018 IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), 2018.
6. Rodinia benchmark suite for Xilinx FPGAs

Επικοινωνία: Παναγιώτης Μπάκος, pmpakos@cslab.ece.ntua.gr

10 Αξιολόγηση των state of the art αλγορίθμων επίλυσης του MST προβλήματος

Το πρόβλημα επίλυσης του Minimum Spanning Tree (MST) είναι ένα από τα πιο γνωστά προβλήματα στην επιστήμη των υπολογιστών. Τρεις είναι οι βασικοί αλγόριθμοι που επιλύουν αυτό το πρόβλημα. Οι αλγόριθμοι του Boruvka, του Prim και του Kruskal. Ο κάθε ένας από αυτούς έχει μελετηθεί αρκετά και σήμερα υπάρχουν διάφορες παράλληλες υλοποιήσεις των παραπάνω αλγορίθμων, που λύνουν το πρόβλημα του MST σε αρκετά πιο σύντομο χρόνο από ότι οι σειριακές εκδοχές τους. Σκοπός αυτής της εργασίας είναι να υλοποιηθούν οι state of the art παράλληλοι αλγόριθμοι, όπως προτείνονται στην σχετική βιβλιογραφία, στο περιβάλλον του εργαστηρίου, με διαφορετικά datasets ώστε να έχουμε μια ολοκληρωμένη αξιολόγηση τους, τόσο για shared memory όσο και για distributed αρχιτεκτονικές.

Σχετικά Μαθήματα: Συστήματα Παράλληλης Επεξεργασίας

Σχετική Βιβλιογραφία:

1. https://en.wikipedia.org/wiki/Kruskal's_algorithm
2. https://en.wikipedia.org/wiki/Prim's_algorithm
3. https://en.wikipedia.org/wiki/Borůvka's_algorithm
4. Engineering Massively Parallel MST Algorithms
   (https://arxiv.org/abs/2302.12199)
5. MASTIFF: Structure-Aware Minimum Spanning Tree-Forest
   (https://pureadmin.qub.ac.uk/ws/portalfiles/portal/330214109/MASTIFF_Authors_Copy.pdf)

Επικοινωνία: Κατσιγιάννης Τάσος, tkats@cslab.ece.ntua.gr
Σιακαβάρας Δημήτρης , jimsiak@cslab.ece.ntua.gr

11 Δρομολόγηση Εφαρμογών και Διαχείριση Πόρων σε Υπολογιστικά Κέντρα

Οι υπολογιστικές υποδομές των Υπολογιστικών Κέντρων (Datacenters) χρησιμοποιούνται για την ταυτόχρονη εκτέλεση εφαρμογών. Η κατάλληλη χρονοδρομολόγηση και η διαχείριση των κοινόχρηστων πόρων του συστήματων αποτελούν καθοριστικούς παράγοντες για την αποτελεσματική χρήση των υπολογιστικών πόρων και την εξοικονόμηση χρόνου και ενέργειας.

11.1 Χαρακτηρισμός Εφαρμογών με χρήση intrusive micro-benchmarks

Καθώς η εκτέλεση πολλών τύπων υπηρεσιών μεταφέρεται σε συστήματα μεγάλης κλίμακας, η πρόκληση της διατήρησης υψηλής ποιότητας υπηρεσίας συνεχώς μεγαλώνει. Η απουσία αποδοτικών λύσεων διαμοιρασμού των κοινόχρηστων πόρων οδηγεί τους Cloud Service Providers στην απομόνωση ολόκληρων servers για την εκτέλεση εφαρμογών με αυστηρούς περιορισμούς για την επίδοση τους. Αυτό όμως οδηγεί στην υποχρησιμοποίηση αυτών των πόρων και την αύξηση του λειτουργικού κόστους. Για την αντιμετώπιση των ζητημάτων αυτών προτείνονται τεχνικές διαχείρισης των κοινόχρηστων πόρων (Last Level Cache - Intel CMT CAT, Memory Bandwidth, Core isolation), τεχνικές χαρακτηρισμού των εφαρμογών ως προς τους κρίσιμους πόρους με σκοπό τη συνεκτέλεση εφαρμογών με συμπληρωματικές απαιτήσεις για πόρους. Σκοπός της διπλωματικής είναι η ανάπτυξη ενός μηχανισμού που θα προβλέπει τις ανάγκες των εφαρμογών από άποψης πόρων με χρήση micro-benchmarks που, στερώντας πόρους από την κάθε εφαρμογή, θα αποκαλύπτουν τις απαιτήσεις της.

Σχετικά Μαθήματα: Συστήματα Παράλληλης Επεξεργασίας

Σχετική Βιβλιογραφία:

1. Quasar: resource-efficient and QoS-aware cluster management
2. Paragon: QoS-aware scheduling for heterogeneous datacenters
3. Heracles: improving resource efficiency at scale

Επικοινωνία: Γιάννης Παπαδάκης, ypap@cslab.ece.ntua.gr

11.2 Χρονοδρομολόγηση εφαρμογών και ανάθεση υπολογιστικών πόρων σε υπολογιστικά συστήματα υψηλής επίδοσης

Τα υπολογιστικά συστήματα υψηλής επίδοσης (High Performance Computing clusters -HPC clusters) είναι ευρέως διαδεδομένα και συχνά χρησιμοποιούνται για την επίλυση πολύπλοκων προβλημάτων σε ποικίλες ερευνητικές περιοχές όπως η πρόγνωση και η μοντελοποίηση των καιρικών φαινομένων καθώς και η διερεύνηση της ακολουθίας του ανθρώπινου γονιδιώματος, αλλά και εν γένει προβλημάτων που απαιτούν μεγάλο υπολογιστικό κόστος σε επεξεργαστή, μνήμη, επικοινωνία μεταξύ των επιμέρους διεργασιών. Το βασικό λογισμικό που συνθέτει και ενορχηστρώνει μια τέτοια υπολογιστική υποδομή, ονομάζεται διαχειριστής πόρων (resource manager) και περιλαμβάνει έναν χρονοδρομολογητή εργασιών (job scheduler). Ο χρονοδρομολογητής εργασιών επικοινωνεί με τον διαχειριστή πόρων προκειμένου να πληροφορηθεί για τις ουρές (queues), τα φορτία των υπολογιστικών κόμβων (nodes) και την διαθεσιμότητα των πόρων, ώστε να πάρει αποφάσεις για τη χρονοδρομολόγηση εργασιών. Επιπλέον, μια αναπόσπαστη και άρρηκτα συνδεδεμένη μονάδα με τον χρονοδρομολογή, αποτελεί ο κατανεμητής πόρων (resource allocator), ο οποίος αναλαμβάνει να διαμοιράσει τους υπολογιστικούς πόρους στις αντίστοιχες εργασίες βάσει κάποιου γενικού ή ειδικού ανά εφαρμογή/περίσταση σχήματος.

11.2.1 Profiling MPI εφαρμογών και εκπαίδευση μοντέλου με χρήση τεχνικών μηχανικής μάθησης (Machine Learning)

Για τις εργασίες που απαιτούν αρκετές ανταλλαγές μηνυμάτων (communication intensive), η ανάθεση κόμβων με διάσπαρτη κατανομή, περιμένουμε να μειώσει την αποδοτικότητα της εκτέλεσής τους. Αντιθέτως, οι εργασίες που απαιτούν συχνή μεταφορά δεδομένων από και προς τη μνήμη (memory intensive), η διάσπαρτη ανάθεση κόμβων αναμένουμε να την αυξήσει, λόγω μικρότερης συμφόρησης στο δίαυλο μνήμης. Επιπλέον, σε εφαρμογές που απαιτούν κυρίως επεξεργαστική ισχύ (computation intensive) η αποδοτικότητα της εκτέλεσής αναμένουμε να είναι σταθερή ανεξάρτητα από το σχήμα ανάθεσης. Επιπλέον, η σειρά και ο τρόπος ανάθεσης των εργασιών στους αντίστοιχους πόρους του συστήματος μπορεί να αποτελέσει καθοριστικός παράγοντας για την απόδοση του χρόνου εκτέλεσής τους ή τη ρυθμαπόδοση ολόκληρου του υπολογιστικού συστήματος (system throughput). Ως εκ τούτου, έχει δοθεί ιδιαίτερη βαρύτητα σε διαδικασίες κατανόησης των χαρακτηριστικών των εφαρμογών (monitoring/profiling) σε συνδυασμό με σύγχρονες τεχνικές μηχανικής μάθησης (machine learning -ML-), μέσω στατικής ανάλυσης του πηγαίου κώδικα. Η γνώση των χαρακτηριστικών μια εφαρμογής αποτελεί κομβικό σημείο για την περαιτέρω εφαρμογή αλγορίθμων χρονοδρομολόγησής της και/ή διανομής υπολογιστικών πόρων.
Σκοπός της παρούσας διπλωματικής είναι η μελέτη στο πως διαφορετικές τοπολογίες ανάθεσης κόμβων για MPI εφαρμογές επηρεάζουν την επίδοσή τους. Με τη χρήση εργαλείων profiling MPI εφαρμογών όπως mpiP, intelVTUNE, scalasca, scoreP στόχος μας είναι να δημιουργήσουμε ένα μοντέλο το οποίο θα μπορεί να προβλέψει την επίδοση μιας εφαρμογής ανάλογα με το με την τοπολογία ανάθεσης κόμβων. Για την υλοποίηση του μοντέλου μπορούν να μελετηθούν τόσο αναλυτικές προσεγγίσεις όσο και οι πιο σύγχρονες προσεγγίσεις Machine Learning.

11.2.2 Ανάπτυξη εργαλείου για την εκτέλεση του διαχειριστή πόρων OAR3 εικονικά σε δουλειά του διαχειριστή πόρων SLURM με χρήση εργαλείων της Bash και containers (ατομική ή συνεργατική διπλωματική εργασία)

Στους σύγχρονους υπερυπολογιστές το σύστημα διαχείρισης πόρων είναι υπεύθυνο για τη χρονοδρομολόγηση των εφαρμογών και την κατανομή των αναγκαίων πόρων σε αυτές. Και οι δύο διαδικασίες γίνονται με κριτήρια καθορισμένα, που αφορούν τον αλγόριθμο χρονοδρομολόγησης και την πολιτική

διαμοιρασμού πόρων αντίστοιχα. Στις μέρες μας, υπάρχει ποικιλία εργαλείων διαχείρισης πόρων που μπορούν να εγκατασταθούν σε υπάρχοντα clusters και να αποτελέσουν το βασικό λογισμικό ενορχήστρωσης των υπολογιστικών δομών. Η επιλογή τους είναι στην ευχέρεια των διαχειριστών των συστημάτων αυτών. Αν και πολύτιμοι, οι διαχειριστές πόρων είναι συνήθως δύσκολα τροποποιήσιμοι, ενώ το σημαντικότερο ζήτημα είναι πως η λειτουργία τους μπορεί να τροποποιηθεί μόνο με παρέμβαση των διαχειριστών του συστήματος. Το γεγονός αυτό δυσχεραίνει τους χρήστες/ερευνητές που θέλουν να εκτελέσουν τις εφαρμογές τους με τρόπο διαφορετικό από τον προκαθορισμένο.
Σκοπός της παρούσας διπλωματικής εργασίας είναι η παράκαμψη αυτού του προβλήματος μέσω της εγκατάστασης ενός σύγχρονου διαχειριστή πόρων (OAR3) κατά την υποβολή εργασίας στο πραγματικό διαχειριστή πόρων SLURM. Το εργαλείο αυτό θα δώσει τη δυνατότητα στο χρήστη/ερευνητή να εγκαταστήσει και να εκτελέσει σε έναν πραγματικό υπερυπολογιστή δικούς του αλγοριθμούς και γενικότερα τροποποιήσεις στο λογισμικό OAR3 (διαχειριστής πόρων που θα εκτελείται στο παρασκήνιο).

11.2.3 Μελέτη συμπεριφοράς MPI εφαρμογών σε καθεστώς συνεκτέλεσης (co-execution) σε πραγματικό υπερυπολογιστή

Υπάρχουν διάφοροι αλγόριθμοι στην κατηγορία των space-sharing αλγορίθμων χρονοδρομολόγησης, όπως ο First Come First Served (FCFS), ο Shortest Job First (SJF), ο Longest Job First (LJF), ο Backfilling κ.α. Οι αλγόριθμοι αυτοί -στα υπολογιστικά σύστημα υψηλής επίδοσης- δεσμεύουν, συνήθως, πόρους στο επίπεδο του κόμβου. Εξ' ορισμού η επιλογή ανάθεσης πόρων στον επίπεδο κόμβου (δεδομένου ότι οι κόμβοι περιλαμβάνουν ολοένα περισσότερα και μεγαλύτερα εξαρτήματα υλικού πια) είναι αντιπαραγωγική όσον αφορά τη ρυθμαπόδοση (throughput) του συστήματος, την κατανάλωση ενέργειας και κόστους. Μελέτες δείχνουν πως το co-scheduling, δηλαδή η ανάθεση πόρων στο επίπεδο του πυρήνα (κι άρα η εκτέλεση διαφορετικών εφαρμογών ταυτόχρονα στους ίδιους κόμβους), οδηγεί σε αποτελεσματικότερη χρήση των υπολογιστικών πόρων.
Σκοπός της διπλωματικής αποτελεί η πειραματική μελέτη και αξιολόγηση της συμπεριφοράς MPI εφαρμογών σε καθεστώς συνεκτέλεσης (co-execution) σε πραγματικό υπερυπολογιστή.

Σχετικά Μαθήματα: Συστήματα Παράλληλης Επεξεργασίας

Σχετική Βιβλιογραφία:

1. https://en.wikipedia.org/wiki/Supercomputer
2. https://en.wikipedia.org/wiki/Message_Passing_Interface
3. https://doc.aris.grnet.gr/development/perf/#vtune-amplifier-xe
4. https://doc.aris.grnet.gr/development/perf/#scalasca
5. https://doc.aris.grnet.gr/development/perf/#mpip
6. https://en.wikipedia.org/wiki/Slurm_Workload_Manager
7. https://oar-3.readthedocs.io/en/latest/
8. https://github.com/cslab-ntua/oar3
9. http://artemis.cslab.ece.ntua.gr:8080/jspui/handle/123456789/18315

Επικοινωνία: Νικόλαος Τριανταφύλλης, ntriantafyl@cslab.ece.ntua.gr
Ευστράτιος Καραπαναγιώτης, skarapan@cslab.ece.ntua.gr
Γεώργιος Γκούμας, goumas@cslab.ece.ntua.gr, 210-772-2402

IV. Κατανεμημένα Συστήματα - Προχωρημένα θέματα βάσεων δεδομένων

12 Μελέτη και βελτιστοποίηση του sync των Blockchain clients

Οι κόμβοι του Blockchain (κυρίως οι full nodes) αποθηκεύουν μεγάλο όγκο δεδομένων που σχετίζονται με το state του, τα transactions που έχουν γίνει, τα δεδομένα των έξυπνων συμβολαίων [1]. Συνήθως τα δεδομένα αυτά φυλάσσονται σε δενδρικές δομές αποθήκευσης (tries) που προσφέρουν γρήγορη αναζήτηση και που υλοποιούνται με τη βοήθεια κάποιου key-value store (leveldb [2], rocksdb [3]). Τα δεδομένα αυτά ανανεώνονται με κάθε νέο block που μπαίνει στο blockchain: Κάθε κόμβος τρέχει τα transactions του block σειριακά και ανανεώνει το state κάνοντας put/get στο key value store. Ιδίως στη φάση του sync, όταν δηλαδή πρωτομπαίνει ένας client στο blockchain και έχει να τρέξει transactions από πολλά blocks συνεχόμενα, υπάρχουν μεγάλες καθυστερήσεις λόγω α) της σειριακής εκτέλεσης των transactions και β) λόγω disk I/O [4]. Μας ενδιαφέρει να μελετήσουμε λύσεις ώστε να επιταχύνουμε αυτές τις λειτουργίες. Πιθανές κατευθύνσεις είναι:

1) Να χρησιμοποιήσουμε κάποιο in-memory database αντί για disk-based key value store [5]. 2) Να χρησιμοποιήσουμε ένα επίπεδο caching σε κομμάτια του key value store. Η επιλογή των δεδομένων στην cache θα γίνεται μέτά από κάποια ανάλυση στο workload ή και forecasting (δηλ. χρησιμοποιώντας ml τεχνικές). 3) Να ενσωματώνουμε concurrency τεχνικές στην εκτέλεση των transactions ώστα αυτά να μη χρειάζεται να εκτελούνται σειριακά, αλλά παράλληλα [6].

Σχετικά Μαθήματα: Κατανεμημένα Συστήματα

Σχετική Βιβλιογραφία:

1. Getting Deep Into Ethereum: How Data Is Stored In Ethereum?
2. LevelDB
3. RocksDB
4. Why Syncing Ethereum Node Is Slow
5. Gorenflo, Christian, et al. ”Fastfabric: Scaling hyperledger fabric to 20,000 transactions per second.” 2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC). IEEE, 2019.
6. Amiri, Mohammad Javad, Divyakant Agrawal, and Amr El Abbadi. ”Parblockchain: Leveraging transaction parallelism in permissioned blockchain systems.” 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS). IEEE, 2019.

Επικοινωνία: Κατερίνα Δόκα, katerina@cslab.ece.ntua.gr, 210-772-1175

13 Σύστημα αυθεντικοποίησης φοιτητών και έκδοσης πιστοποιητικών πάνω στο δίκτυο Cardano

Το Cardano είναι μία από τις πιο γνωστές blochchain πλατφόρμες που χρησιμοποιεί το Ouroboros ως το consensus πρωτόκολλό του. Η χρήση του συγκεκριμένου proof-of-stake πρωτοκόλλου, επιτρέπει στο Cardano πολύ γρήγορές και χαμηλού κόστους validation των συναλλαγών. Επιπλέον υπάρχει

η δυνατότητα χρήσης smart contracts και εφαρμογών που τρέχουν πάνω στην πλατφόρμα. Η αξιοποίηση των smart contracts για την αυθεντικοποίηση των φοιτητών μιας Σχολής και για την έκδοση πιστοποιητικών βάση των μαθημάτων που έχουν περάσει κατά την διάρκεια της φοιτητικής τους σταδιοδρομίας, θα μπορούσε να είναι μια αρκετά χρήσιμη περίπτωση αξιοποίησης της συγκεκριμένης blockchain πλατφόρμας. Τα πλεονεκτήματα που θα είχε μια τέτοια προσέγγιση έναντι της παραδοσιακής centralized, είναι ότι θα επέτρεπαν την ασφαλή και άμεση καταχώρηση βαθμών όπως και την εύκολη ανάκτησή τους, από την στιγμή που όλη η διαθέσιμη πληροφορία θα είναι πάνω στο Cardano Blockchain.

Στο πλαίσιο της παρούσας διπλωματικής ζητείται η ανάπτυξη ενός τέτοιου συστήματος στο επίπεδο Τομέα της Σχολής.

Σχετικά Μαθήματα: Κατανεμημένα Συστήματα

Σχετική Βιβλιογραφία:

1. `https://docs.cardano.org/`
2. `https://yoroi-wallet.com/`
3. `https://learnyouahaskell.com/chapters`
4. `http://github.com/input-output-hk/plutus-pioneer-program`
5. `https://play.marlowe-finance.io/`
6. `https://atalaprism.io/`
7. `https://www.skepsispool.com/`

Επικοινωνία: Κατερίνα Δόκα, katerina@cslab.ece.ntua.gr, 210-772-1175
Τάσος Κατσιγιάννης tkats@cslab.ece.ntua.gr

14 Σχεδιασμός και υλοποίηση εργαλείου προσομοίωσης κατανεμημένων συστημάτων επεξεργασίας συνεχών ροών δεδομένων σε πραγματικό χρόνο.

Η επεξεργασία συνεχών ροών δεδομένων σε πραγματικό χρόνο (Stream Processing - SP) είναι μια εφαρμογή που τα τελευταία χρόνια έχει γίνει ιδιαίτερα δημοφιλής, χάρη στην ανάπτυξη των τεχνολογιών επεξεργασίας και αποθήκευσης δεδομένων μεγάλου όγκου (Big Data). Κατανεμημένα συστήματα SP όπως τα Storm, Flink, Spark και Kafka εν γένει χρησιμοποιούν πολλαπλές υλοποιήσεις φυσικών τελεστών σε παράλληλη διάταξη, κατανέμοντας τα δεδομένα προς επεξεργασία μεταξύ τους. Ωστόσο, στοιχεία όπως:

- η κατανομή των δεδομένων,
- η αρχιτεκτονική και ο τρόπος λειτουργίας των συγκεκριμένων συστημάτων (true streaming vs micro-batch),
- το είδος των τελεστών (stateful vs stateless), αλλά και
- οι τεχνικές κατάτμησής και διαμοιρασμού τους στους υπολογιστικούς πόρους του συστήματος,

επηρεάζουν σημαντικά τις επιδόσεις.

Στόχος αυτής της διπλωματικής εργασίας είναι να σχεδιαστεί και υλοποιηθεί ένα παραμετροποιήσιμο εργαλείο προσομοίωσης της λειτουργίας ενός συστήματος επεξεργασίας συνεχών ροών δεδομένων σε πραγματικό χρόνο προκειμένου να μελετηθεί η επίδραση των παραπάνω παραμέτρων στην επίδοσή της λειτουργίας του.

Σχετική Βιβλιογραφία:

1. Gedik, Buğra. "Partitioning functions for stateful data parallelism in stream processing." The VLDB Journal 23, no. 4 (2014).

2. Nasir, Muhammad Anis Uddin, et al. "The power of both choices: Practical load balancing for distributed stream processing engines." 2015 IEEE 31st International Conference on Data Engineering. IEEE, 2015.

3. Nasir, Muhammad Anis Uddin, et al. "When two choices are not enough: Balancing at scale in distributed stream processing." 2016 IEEE 32nd International Conference on Data Engineering (ICDE). IEEE, 2016.

4. Katsipoulakis, Nikos R., Alexandros Labrinidis, and Panos K. Chrysanthis. "A holistic view of stream partitioning costs." Proceedings of the VLDB Endowment 10.11 (2017).

5. Abdelhamid, Ahmed S., et al. "Prompt: Dynamic data-partitioning for distributed micro-batch stream processing systems." Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data. 2020.

6. Chen, Hanhua, Fan Zhang, and Hai Jin. "Pstream: a popularity-aware differentiated distributed stream processing system." IEEE Transactions on Computers 70.10 (2020).

Επικοινωνία: Νίκος Χαλβαντζής, nchalv@cslab.ece.ntua.gr

15 Ελαστική διαχείριση πόρων συστημάτων διαχείρισης δεδομένων μεγάλης κλίμακας με τεχνικές μηχανικής μάθησης

Τα κατανεμημένα συστήματα επεξεργασίας δεδομένων μεγάλης κλίμακας χρησιμοποιούνται ευρέως για την αποδοτική και κλιμακώσιμη εκτέλεση αλγορίθμων πάνω από δεδομένα μεγάλου όγκου. Συστήματα αναλυτικής επεξεργασίας δεδομένων όπως το Apache Hive και Spark SQL χρησιμοποιούνται για την εκτέλεση batch-SQL ερωτημάτων, βάσεις NoSQL χρησιμοποιούνται για τον αποδοτική εκτέλεση real-time ερωτημάτων ενώ συστήματα όπως το TensorFlow και το Spark MLLib για την εκτέλεση αλγορίθμων μηχανικής μάθησης. Τα συστήματα αυτά έχουν την δυνατότητα με την προσθήκη επιπλέον πόρων να βελτιώσουν την εκτέλεση των αλγορίθμων εκμεταλλευόμενα την παράλληλη φύση της επεξεργασίας, κάτι που τα κάνει ιδανικά για να τρέχουν σε περιβάλλοντα υπολογιστικών νεφών: με την εκμετάλλευση της ιδιότητας της "ελαστικότητας" μπορεί κάποιος να αυξομειώνει τους πόρους της υποδομής ανάλογα με τις ανάγκες των χρηστών.

Το εργαστήριο έχει εργαστεί στον τομέα αυτό με την γενική προσέγγιση της δημιουργίας μοντέλων μηχανικής μάθησης τα οποία προβλέπουν και/ή αποφασίζουν την κατάλληλη δράση (πχ. προσθαφαίρεση υπολογιστικών κόμβων, αλλαγή χαρακτηριστικών εφαρμογής, κλπ) για να ανταπεξέλθει η εφαρμογή σε ένα δοθέν φορτίο εργασίας με βάση ορισμένα κριτήρια επίδοσης που δίνει ο χρήστης.

Το εργαστήριο έχει μελετήσει τεχνικές ενισχυτικής μάθησης (reinforcement learning) για την διαχείριση πόρων συστοιχιών NoSQL (σύστημα Tiramola [1]). Κατόπιν, μελέτησε μια προσέγγιση με προσαρμοστική δημιουργία του χώρου καταστάσεων της μαρκοβιανής αλυσίδας σε περίπτωση που υπήρχαν πολλές διαστάσεις (σύστημα MDP_DT [2]) καθώς και μια προσέγγιση με βαθιά ενισχυτική μάθηση (deep reinforcement learning).

Πιθανές κατευθύνσεις:

1. Επέκταση του μοντέλου ενισχυτικής μάθησης για την εφαρμογή επιπλέον ενεργειών κατά την απόφαση αλλαγής των πόρων μιας εφαρμογής. Προς το παρόν οι αλγόριθμοι προσθαφαιρούν μόνο υπολογιστικούς κόμβους, ενώ υπάρχει η δυνατότητα αλλαγής και επιπλέον χαρακτηριστικών (μέγεθος RAM, αριθμό πυρήνων, κλπ).

2. Εφαρμογή ενός από τους αλγορίθμους των [bigdata, ccgrid] σε ένα νέο σύστημα, πχ στο TensorFlow ή στο Apache Spark με διαφορετικά είδη φορτίου.

3. Πειραματική αποτίμηση επίδοσης διαφορετικών αλγορίθμων ελαστικής διαχείρισης πόρων σε διαφορετικά συστήματα.

4. Εφαρμογή ενός από τους παραπάνω αλγορίθμους στο σύστημα Kubernetes της Google για την διαχείριση ενός κατανεμημένου framework όπως το TensorFlow (πχ Kubeflow) [citation]

5. Εφαρμογή διαφορετικών τεχνικών Deep Reinforcement Learning για την εκπαίδευση του μοντέλου. Προτάσεις:

    - Τα recurrent deep neural networks χρησιμοποιούνται ευρέως όταν έχουμε να αντιμετωπίσουμε incomplete datasets, όταν πχ κάποιες μετρήσεις που δομούν ένα state λείπουν ή περιέχουν θόρυβο, όπου δεδομένου της φύσης του προβλήματος (cloud environment), σε πραγματικές συνθήκες θα συναντούμε τέτοια datasets.
    - Στο deep reinforcement learning χρησιμοποιείται η εξίσωση του Belman:
      Q(s,a)=V(s)+A(a)
      όπου η αξία του Q δομείται από την αξία του να είσαι σε ένα συγκεκριμένο state, V(s) συν το κέρδος που θα αποκόμιζες αν έπαιρνες το action a. Στην έως τώρα υλοποίησή μας το δίκτυό μας υπολογίζει τα Q values συνολικά. Στα dueling deep neural networks στόχος είναι δύο διαφορετικά δίκτυα να υπολογίζουν ξεχωριστά τα V(s), A(a) και εν συνεχεία να δομείται η συνολική λύση. Αυτή η τεχνική βοηθάει στο decoupling του προβλήματος και στην καλύτερη εκπαίδευση του δικτύου, ενώ παράλληλα αποφεύγει τη σπατάλη πόρων σε ειδικές περιπτώσεις όπου ένα εκ των V(s), A(a) δεν έχουν μεγάλη σημασία για την πολιτική που έχει επιλέξει ο χρήστης και επομένως ο υπολογισμός τους μπορεί να παραλειφθεί.

Σχετικά Μαθήματα: Προχωρημένα θέματα βάσεων δεδομένων, Κατανεμημένα Συστήματα

Σχετική Βιβλιογραφία:

1. D. Tsoumakos, I. Konstantinou, C. Boumpouka, S. Sioutas and N. Koziris: Automated, Elastic Resource Provisioning for NoSQL Clusters Using TIRAMOLA. In proceedings of the 13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), Delft, The Netherlands, May 13-16, 2013. Best Paper Award

2. K. Lolos, I. Konstantinou, V. Kantere and N. Koziris: Elastic Management of Cloud Applications using Adaptive Reinforcement Learning In proceedings of the 2017 IEEE International Conference on Big Data (BigData 2017), Boston, MA, USA December 11-14 2017

3. K. Bitsakos, I. Konstantinou and N. Koziris: DERP: A Deep Reinforcement Learning Cloud System for Elastic Resource Provisioning In proceedings of the 2018 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), Nicosia, Cyprus December 10-13 2018

4. M. Chrysopoulos, I. Konstantinou, and N. Koziris: Deep Reinforcement Learning in Cloud Elasticity through Offline Learning and Return Based Scaling In proceedings of the 16th IEEE International Conference on Cloud Computing (CLOUD), Chicago, IL, USA, July 2-8 2023.

Επικοινωνία: Ιωάννης Κωνσταντίνου, ikons@cslab.ece.ntua.gr

16 Συγκριτική Ανάλυση των Αρχιτεκτονικών Data Lake: Apache Iceberg, Apache Hudi και Delta Lake

Στον σημερινό κόσμο, που κυριαρχεί η χρήση των δεδομένων, οι οργανισμοί αντιμετωπίζουν διαρκώς προκλήσεις στη διαχείριση και επεξεργασία μεγάλων όγκων δεδομένων αποτελεσματικά και αξιόπιστα. Οι πίνακες μορφών (Table formats) επιτρέπουν την αλληλεπίδραση με τα data lakes με τον ίδιο εύκολο τρόπο που αλληλεπιδρούμε με βάσεις δεδομένων, χρησιμοποιώντας τα αγαπημένα μας εργαλεία και γλώσσες. Σε αυτή τη διπλωματική εργασία καλείστε να πραγματοποιήσετε μία περιεκτική συγκριτική ανάλυση τριών δημοφιλών αρχιτεκτονικών data lake: του Apache Iceberg [1], του Apache Hudi [2] και του Delta Lake [3]. Ο κύριος στόχος αυτής της μελέτης είναι να αξιολογήσει τα πλεονεκτήματα και τα αδυναμίες κάθε αρχιτεκτονικής data lake όσον αφορά το schema versioning, το scalability και τη δυνατότητα του time-travel. Με τη διεξοδική εξέταση αυτών των τεχνολογιών, αυτή η διπλωματική εργασία σκοπεύει να βοηθήσει τους οργανισμούς να προβούν σε ενημερωμένες αποφάσεις κατά την επιλογή της πλέον κατάλληλης αρχιτεκτονικής data lake και συνόλου εργαλείων για τις συγκεκριμένες τους ανάγκες στην επεξεργασία δεδομένων. Αυτές οι αρχιτεκτονικές data lake μπορούν επίσης να ενσωματωθούν με εργαλεία επεξεργασίας δεδομένων όπως το Apache Hive, το Apache Spark, το Spark Operator και το Trino (προηγουμένως γνωστό ως Presto), ενώ η πτυχή του performance μπορεί να εξεταστεί χρησιμοποιώντας το TPC-DS benchmark [4].

Σχετική Βιβλιογραφία:

[1] https://iceberg.apache.org/
[2] https://hudi.apache.org/blog/2021/07/21/streaming-data-lake-platform/
[3] https://delta.io/
[4] https://www.tpc.org/tpcds/

Επικοινωνία: Ιωάννης Κωνσταντίνου, ikons@cslab.ece.ntua.gr

17 Υλοποίηση και σύγκριση απόδοσης διαφορετικών κατανεμημένων συστημάτων επικοινωνίας (distributed frameworks) για χρήση σε μηχανική μάθηση και ανάλυση δεδομένων

Ο συνδυασμός διαχείρισης δεδομένων (data management), η επεξεργασία τους σε συστήματα υψηλής απόδοσης (high-performance computing) και η χρήση τους σε αλγόριθμους μηχανικής μάθησης (machine learning) χρησιμοποιείται όλο και περισσότερο. Οι εργασίες αυτές απαιτούν τόσο μεγάλη μνήμη όσο και ισχυρή επεξεργαστική ισχύ, σε κατανεμημένα συστήματα με πολλούς διαφορετικούς

υπολογιστικούς κόμβους (distributed nodes). Οι κόμβοι αυτοί επικοινωνούν συνεχώς κατά την εκτέλεση ανταλλάσσοντας μεγάλο όγκο δεδομένων. Στόχος της διπλωματικής αυτής είναι η μελέτη διαφορετικών τρόπων επικοινωνίας σε κατανεμημένα συστήματα και η υλοποίηση διαφορετικών τρόπων επικοινωνίας.

Το σύστημα Daphne[1][2] χρησιμοποιείται για την εκτέλεση αλγορίθμων μηχανικής μάθησης και ανάλυσης δεδομένων, και έχει τη δυνατότητα εκτέλεσης σε ένα μηχάνημα όσο και σε ένα κατανεμημένο σύστημα (distributed runtime). Αυτή τη στιγμή το Daphne χρησιμοποιεί το framework επικοινωνίας gRPC[3], ένα ανοιχτού κώδικα υψηλής απόδοσης Remote Procedure Call (RPC), ενώ σύντομα θα υπάρχει και υποστήριξη για MPI. Στα πλαίσια αυτής της διπλωματικής θα χρησιμοποιηθεί το Daphne για να μελετηθούν (i) διαφορετικά frameworks τα οποία χρησιμοποιούνται σε distributed περιβάλλοντα για machine learning και data analytics, (ii) θα γίνει προσπάθεια να υλοποιηθούν μερικά από αυτά στο Daphne και τέλος (iii) θα γίνει μια αξιολόγηση και σύγκριση των διαφορετικών frameworks χρησιμοποιώντας κάποια benchmarks στο distributed runtime του Daphne.

Σχετικά Μαθήματα: Συστήματα Παράλληλης Επεξεργασίας, Κατανεμημένα Συστήματα

Σχετική Βιβλιογραφία:

1. ”DAPHNE: An Open and Extensible System Infrastructure for Integrated Data Analysis Pipelines” in conference on Innovative Data Systems Research, CIDR, 2022.
2. `https://daphne-eu.eu/`
3. `https://grpc.io/`

Επικοινωνία: Αριστοτέλης Βοντζαλίδης, avontz@cslab.ece.ntua.gr
Γεώργιος Γκούμας , goumas@cslab.ece.ntua.gr, 210-772-4133
Στράτος Ψωμαδάκης, psomas@cslab.ece.ntua.gr
Κωνσταντίνος Μπιτσάκος, kbitsak@cslab.ece.ntua.gr

18 Υλοποίηση μηχανισμού αξιοποίησης παραλληλισμού δεδομένων (data parallelism) για αλγορίθμους μηχανικής μάθησης (machine learning) και ανάλυσης δεδομένων (data analytics)

Οι αλγόριθμοι μηχανικής μάθησης (ML) και ανάλυσης δεδομένων μεγάλης κλίμακας (big data analytics) συχνά μπορούν να παραλληλοποιηθούν ως προς τα δεδομένα (data parallelism). Οι τρόποι με τους οποίους οποίους μπορούν να παραλληλοποιθούν οι συγκεκριμένοι αλγόριθμοι συχνά είναι παρόμοιοι, ανεξάρτητα με το πρόβλημα το οποίο επιλύουν. Από την άλλη η διαδικασία είναι αρκετά απαιτητική προγραμματιστικά. Η παραλληλοποίηση μπορεί να αυτοματοποιηθεί για συγκεκριμένες εντολές (operations), ειδικά όταν τα δεδομένα μπορούν να επεξεργαστούν ανεξάρτητα το ένα από το άλλο. Για παράδειγμα σε ένα πολυπύρηνο σύστημα τα δεδομένα μοιράζονται σε διαφορετικά μερίσματα (chunks) και κάθε πυρήνας εκτελεί πράξεις σε ένα μέρος των συνολικών δεδομένων. Σε ένα κατανεμημένο σύστημα μπορούμε να μοιράσουμε τα δεδομένα σε πολύ περισσότερους κόμβους (και πυρήνες) ωστόσο πρέπει να λάβουμε υπόψιν το αυξημένο κόστος επικοινωνίας μεταξύ κόμβων.

Στόχος αυτής της διπλωματικής είναι η μελέτη τεχνικών παραλληλοποίησης τέτοιων αλγορίθμων για κατανεμημένα συστήματα. Θα μελετηθούν τεχνικές και κριτήρια για την αξιοποίηση του data parallelism σε κατανεμημένα συστήματα. Θα χρησιμοποιηθεί το σύστημα Daphne[1][2] το οποίο εκμεταλλεύεται το data parallelism και αυτοματοποιεί τη διαδικασία παραλληλοποίησης στο τοπικό

runtime. Με βάση αυτό θα υλοποιηθεί αντίστοιχα ο μηχανισμός παραλληλοποίησης αλγορίθμων για το κατανεμημένο σύστημα του Daphne.

Σχετικά Μαθήματα: Συστήματα Παράλληλης Επεξεργασίας, Κατανεμημένα Συστήματα

Σχετική Βιβλιογραφία:

1. "DAPHNE: An Open and Extensible System Infrastructure for Integrated Data Analysis Pipelines" in conference on Innovative Data Systems Research, CIDR, 2022.

2. `https://daphne-eu.eu/`

Επικοινωνία: Αριστοτέλης Βοντζαλίδης, avontz@cslab.ece.ntua.gr
Γεώργιος Γκούμας, goumas@cslab.ece.ntua.gr, 210-772-4133
Στράτος Ψωμαδάκης, psomas@cslab.ece.ntua.gr
Κωνσταντίνος Μπιτσάκος, kbitsak@cslab.ece.ntua.gr

19 Resource Scheduling με χρήση Βαθειάς Ενισχυτικής Μάθησης με εφαρμογή σε Distributed Frameworks Μηχανικής Μάθησης

Την τελευταία δεκαετία η χρήση κατανεμημένων συστημάτων (πολλές φορές πλέον ετερογενών αρχιτεκτονικών, επεξεργαστών και πόρων - λ.χ. CPU, GPU, TPU, FPGA) για επεξαργασία δεδομένων μεγάλης κλίμακας, κυριααρχεί τόσο στο πεδίο της ακαδημαϊκής έρευνας όσο και στο industry.

Προκειμένου η συνεργασία των διαφορετικών components ενός ενιαίου κατανεμημένου συστήματος να βελτιστοποιηθεί, οι πόροι να χρησιμοποιούνται βέλτιστα αλλά και οικονομικά, χρησιμοποιούνται μοντέλα για scheduling. Στην παρούσα εργασία θα ασχοληθούμε με τέτοια μοντέλα πάνω σε frameworks μηχανικής μάθησης.

Framework μηχανικής μάθησης όπως το tensorflow [1] και το keras [2] χρησιμοποιούν συστήματα όπως το kubeflow [3] για αποδοτικότερο scheduling, μεταξύ άλλων.

Στην παρούσα εργασία θα ασχοληθούμε με το scheduling στο Daphne. Το σύστημα Daphne [4][5] χρησιμοποιείται για την εκτέλεση αλγορίθμων μηχανικής μάθησης και ανάλυσης δεδομένων, και έχει τη δυνατότητα εκτέλεσης σε ένα μηχάνημα όσο και σε ένα κατανεμημένο σύστημα (distributed runtime).

Στόχος της διπλωματικής είναι ένα performance evaluation του Daphne, με χρήση benchmarking tools για distributed συστήματα και η κατασκευή μοντέλων βελτιστοποίησης του scheduling, βασισμένα στα δίκτυα βαθειάς ενισχυτικής μάθησης. Τα βαθειά νευρωνικά δίκτυα έχουν δείξει ανωτερότητα και αποτελεσματικότητα σε σχέση με τις παραδοσιακές τεχνικές μηχανικής μάθησης σε προβλήματα που ο χώρος καταστάσεων της εισόδου είναι αρκετά μεγάλος.

Σε προηγούμενες δουλειές στο εργαστήριο έχει χρησιμοποιηθεί το Double Deep Q learning [6] για dynamic scheduling και resource allocation σε κατανεμημένα συστήματα.

Στην παρούσα διπλωματική θα μελετήσουμε τα Double Deep Q learning δίκτυα [6], τα Dueling Deep Q learning δίκτυα, ενώ δίνεται η ελευθερία και για μελέτη διαφορετικών προσσεγίσεων.

Σχετικά Μαθήματα: Συστήματα Παράλληλης Επεξεργασίας, Κατανεμημένα Συστήματα, Προχωρημένα θέματα βάσεων δεδομένων

Σχετική Βιβλιογραφία:

1. `https://www.tensorflow.org/`

2. `https://keras.io/`

3. https://www.kubeflow.org/

4. "DAPHNE: An Open and Extensible System Infrastructure for Integrated Data Analysis Pipelines" in conference on Innovative Data Systems Research, CIDR, 2022.

5. https://daphne-eu.eu/

6. http://www.cslab.ece.ntua.gr/~ikons/derp_proceedings.pdf

7. https://www.tensorflow.org/agents/tutorials/0_intro_rl

Επικοινωνία: Κωνσταντίνος Μπιτσάκος, kbitsak@cslab.ece.ntua.gr
Γεώργιος Γκούμας , goumas@cslab.ece.ntua.gr, 210-772-4133
Αριστοτέλης Βοντζαλίδης, avontz@cslab.ece.ntua.gr
Στράτος Ψωμαδάκης, psomas@cslab.ece.ntua.gr

20 Εφαρμογές τεχνικών mechanism design (υποκλάδος της αλγοριθμικής θεωρίας παιγνίων) και deep learning για αποδοτικές online δημοπρασίες πόρων μεταξύ χρηστών σε περιβάλλοντα cloud

Στα σύγχρονα μεγάλα υπολογιστικά νέφη η δυναμική εκχώρηση πόρων σε χρήστες ή tasks χρηστών ανάλογα με την ανάγκη του χρήστη και τη διαθεσιμότητα πόρων στο δίκτυο μια δεδομένη στιγμή εφάπτεται στο πρόβλημα του αποδοτικού resource allocation.

Με την άνοδο του machine learning και την ανάπτυξη περισσότερων non-critical ή easy scalable jobs που μπορεί να τρέξει ο χρήστης σε μια cloud υπηρεσία και για να αντιμετωπιστεί το πρόβλημα των πόρων που μένουν αχρησιμοποίητοι στο AWS, η Amazon δημιούργησε την υπηρεσία Spot Instances όπου τιμές των Vms διαμορφώνονται κατά το δοκούν περιοδικά, ανάλογα με τη ζήτηση.

Αυτό εμπεριέχει τον κίνδυνο κάποιος χρήστης να απωλέσει μεγάλο αριθμό των πόρων του χωρίς προειδοποίηση. Στο εργαστήριο αναπτύξαμε έναν decision component, ονόματει Game Master, που εφαρμόζεται στο Spot Instances και διενεργεί online δημοπρασίες περιοδικά με στόχο οι πόροι ενός χρήστη να αυξομειώνονται ελαστικά.

Για να διασφαλίσουμε την εγκυρότητα, την τιμιότητα και τη φιλαλήθεια των παραπάνω δημοπρασιών χρησιμοποιούμε τεχνικές δανεισμένες από το mechanism design- ένας υποκλάδος της αλγοριθμικής θεωρίας παιγνίων που στόχο έχει την κατασκευή μηχανισμών που οδηγούν τους παίκτες ενός παιγνίου να παρουσιάζουν φιλαλήθη συμπεριφορά κατά τη συμμετοχή τους στο παίγνιο. Επίσης χρησιμοποιούμε μια προσέγγιση ενός deep learning min max νευρωνικού δικτύου.

Καταφέρνουμε να πετύχουμε οφέλη για σωρεία διαφορετικών περιπτώσεων όπως το να πετύχουμε μεγάλα κέρδη για τον cloud vendor ή μέγιστη κοινωνική ωφέλεια για τους χρήστες (οι χρήστες μένουν στην πλειονότητά τους ευχαριστημένοι) Οι προσομοιώσεις πάνω στις οποίες τεστάραμε τον component αφορούν στατιστικά στοιχεία από το Google Trace.

Στόχος μας είναι να η δημιουργία ενός actual cluster με διαφορετικούς χρήστες, με διαφορετικά non-critical applications που συμμετέχουν στις άνωθι δημοπρασίες του Game Master περιοδικά, προσθαφαιρούνται πόροι τους και να δείξουμε πως τα κριτήρια που εμφανώς πληρούνται στις προσομοιώσεις, πληρούνται και σε actual executions environemtns.

Παράλληλα η χρησιμοποίηση του Game Master σε περιβάλλοντα ετερογενών αρχιτεκτονικών θα μπορούσε να εξεταστεί σαν υποψήφιο θέμα.

Σχετική Βιβλιογραφία:

1. A. Tsiourvas, C. Bitsakos, I. Konstantinou, D. Fotakis, and N. Koziris: A Mechanism Design and Learning Approach for Revenue Maximization on Cloud Dynamic Spot Markets In proceedings of the 14th IEEE International Conference on Cloud Computing (CLOUD), September 5-11 2021

2. `https://aws.amazon.com/ec2/spot/`

3. L. Zhang, Z. Li and C. Wu, "Dynamic resource provisioning in cloud computing: A randomized auction approach," IEEE INFOCOM 2014 - IEEE Conference on Computer Communications, Toronto, ON, 2014, pp. 433-441.

4. W. Shi, L. Zhang, C. Wu, Z. Li and F. C. M. Lau, "An Online Auction Framework for Dynamic Resource Provisioning in Cloud Computing," in IEEE/ACM Transactions on Networking, vol. 24, no. 4, pp. 2060-2073, Aug. 2016.

5. Dütting, Paul, Zhe Feng, Harikrishna Narasimhan, David C. Parkes and Sai Srivatsa Ravindranath. "Optimal Auctions through Deep Learning." ICML (2017).

Επικοινωνία: Κωνσταντίνος Μπιτσάκος, kbitsak@cslab.ece.ntua.gr
Ιωάννης Κωνσταντίνου, ikons@cslab.ece.ntua.gr

21 Μελέτη, Υλοποίηση και σύγκριση Κβαντικών Αλγορίθμων Μηχανικής Μάθησης και Κβαντικών Νευρωνικών Δικτύων (Bayessian)

Το Quantum Computing[1] είναι εδώ για να μείνει. Ενώ η ενασχόληση με τον κλάδο για χρόνια περιοριζόταν σε θεωρητικό επίπεδο πλέον υλοποιήσεις Quantum Computers από κολοσσούς όπως η Google[2] και η IBM[3], δίνουν τη δυνατότητα πρακτικής εξέτασης κβαντικών αλγορίθμων σε πραγματικό χρόνο και τα αποτελέσματα αποδεικνύονται ολοένα και πιο υποσχόμενα.

Το Quantum Computing εκμεταλλεύεται αρχές της Κβαντομηχανικής (quantum superposition, interference, and entanglement)[4] και την πιθανοτικής της φύση (ένα σωματίδιο στους Κβαντικούς υπολογιστές πριν μετρηθεί δεν έχει μόνο δύο καταστάσεις που μπορεί να βρίσκεται αλλά άπειρες πάνω σε μια σφαίρα πιθανοτήτων -Bloch Sphere)) προκειμένου να παρουσιάσει εκθετική βελτίωση σε προβλήματα που μέχρι πρόσφατα βρίσκονταν στην NP κλάση, παρουσιάζοντας μια νέα κλάση πολυπλοκότητας, την BQP[5].

Στο εργαστήριο έχουμε μελετήσει Κβαντικούς αλγορίθμους μηχανικής μάθησης, ενώ έχουμε υλοποιήσει μια προσέγγιση ενός υβριδικού kmeans σε πραγματικό Κβαντικό υλικό, παρεχόμενο στο cloud της IBM. Συγκρίναμε τον υβριδικό kmeans με τον κλασσικό kmeans και βγάλαμε συμπεράσματα σχετικά με την αποδοτικότητα του ανοιχτού Quantum hardware για την ώρα.

Στόχος αυτής της εργασίας είναι η μελέτη των Κβταντικών Νευρωνικών Δικτύων[6]. Συγκεκριμένα θα ασχοληθούμε με τη μελέτη και την κατασκευή ενός Quantum Bayessian Network[7], όπου τα βάρη σε κάθε νευρώνα δεν είναι ντετερμινιστικά αλλά πιθανοτικά, για να εκμεταλλευτούμε την πιθανοτική φύση του Κβαντικού υπολογισμού.

Θα συγκρίνουμε τα Quantum νευρωνικά δίκτυα με τα αντίστοιχα κλασικά, θα προτείνουμε βελτιώσεις και θα βγάλουμε συμπεράσματα σχετικά με την υπάρχουσα αποδοτικότητα των ανοιχτών Κβαντικών cloud systems, προτείνοντας αλλαγές και βελτιώσεις.

Σχετικά Μαθήματα: Μηχανική μάθηση, Γραμμική Άλγεβρα, Αρχιτεκτονική Υπολογιστών
Σχετική Βιβλιογραφία:

1. https://en.wikipedia.org/wiki/Quantum_computing
2. https://quantumai.google/
3. https://www.ibm.com/quantum
4. https://www.worldscientific.com/doi/abs/10.1142/9781786348210_0004
5. https://en.wikipedia.org/wiki/Quantum_complexity_theory
6. https://en.wikipedia.org/wiki/Quantum_neural_network
7. https://ieeexplore.ieee.org/document/9759399

Επικοινωνία: Κωνσταντίνος Μπιτσάκος, kbitsak@cslab.ece.ntua.gr
Κωνσταντίνος Νίκας , knikas@cslab.ece.ntua.gr

V. Εργασίες σε συνεπίβλεψη με εξωτερικούς συνεργάτες

22 Create an Etcd Operator to Deploy, Manage, Scale, and Heal Etcd Clusters on Kubernetes Automatically

etcd [1] is a widely-used open source distributed key-value store; it often acts as the single source of truth in modern distributed systems. Most notably, it serves as the data store for Kubernetes [2], it is where Kubernetes stores the state of all objects on a cluster.

Kubernetes is not only the most popular container orchestrator today, but also the most widely used implementation of the "Operator Pattern" [3]; it combines custom data types [*Custom Resource Definitions* or CRDs] with controller programs which encapsulate the knowledge of human operators. The goal is to extend Kubernetes so it can deploy, manage, distribute, scale and heal custom workloads automatically.

Despite etcd's popularity as a distributed key-value store, and its role in Kubernetes, there currently is no established, easy-to-use, widely-accepted, open source operator to run etcd itself on Kubernetes. A number of etcd operators exist, for example [4, 5, 6, 7], but they have a variety of issues, including being unsupported, unmaintained, or outright buggy in their handling of persistent storage, which prevents their use in production environments out-of-the box.

As part of this project, we'll first define a set of requirements for running etcd in a completely automated way on Kubernetes, then evaluate the existing operators against these requirements. We will work to understand and document the most promising of the existing operators, then propose and implement enhancements to its design so it meets said requirements. Throughout this process, we will be exposing our work to the open source community with frequent PRs, so we can implement changes gradually, and ask for feedback.

Our end goal is to end up with a clean, simple, well-documentated etcd operator, which will just work.

Σχετική Βιβλιογραφία:

1. https://etcd.io
2. https://kubernetes.io
3. https://kubernetes.io/docs/concepts/extend-kubernetes/operator/
4. https://github.com/coreos/etcd-operator
5. https://www.infoq.com/presentations/kubernetes-operator-etcd/
6. https://github.com/improbable-eng/etcd-cluster-operator
7. https://github.com/cbws/etcd-operator

Σχετικά Μαθήματα: Εργαστήριο Λειτουργικών Συστημάτων, Κατανεμημένα Συστήματα

Επικοινωνία: Γεώργιος Γκούμας, goumas@cslab.ece.ntua.gr
Βαγγέλης Κούκης, vkoukis@cslab.ece.ntua.gr, vkoukis@arrikto.com

23 Explore Velero and Integrate it with Arrikto Rok and Rok Registry

Velero [1] is an open source tool to safely back up, recover, and migrate whole applications running on Kubernetes clusters, including custom object definitions [2] and persistent volumes [3]. It works both on premises and on most public clouds.

Arrikto Rok [4] is a generic storage data management solution for Kubernetes which enables super fast, low-latency data access over local NVMe SSDs with advanced data services [thin snapshots, fast clones, efficient peer-to-peer synchronization over a decentralized network]. It integrates with Kubernetes via the Container Storage Interface (CSI). [5]

Arrikto Rok Registry [6] is a control plane for building a federation of multiple Rok instances to create a decentralized network and enabled efficient encrypted peer-to-peer data movement across regions, with support for fine-grained authentication and authorization over a single pane of glass.

The objective of this diploma thesis is to explore the use cases for Velero, understand its strengths and limitations, and focus on ways to enhance its operation by integrating it with Rok and the Rok Registry. This comprises two distinct goals:

- Integrate Velero with Rok as its storage backend, both for PVC data and for Kubernetes object metadata
- Integrate Velero with Rok Registry and evaluate its usefulness for disaster recover scenarios and for geo-distributed execution of pipelines across Kubernetes clusters, in hybrid cloud environments.

The end goal is to achieve seamless, production-quality integration of Velero with Arrikto Rok and Rok Registry for real-world use cases.

Σχετική Βιβλιογραφία:

1. https://velero.io/
2. https://kubernetes.io/docs/concepts/extend-kubernetes/operator/
3. https://kubernetes.io/docs/concepts/storage/persistent-volumes/

4. https://docs.arrikto.com/introduction/ekf-features.html#rok
5. https://kubernetes-csi.github.io/docs/
6. https://www.arrikto.com/rok-data-management-platform/

Σχετικά Μαθήματα: Εργαστήριο Λειτουργικών Συστημάτων, Κατανεμημένα Συστήματα

Επικοινωνία: Γεώργιος Γκούμας, goumas@cslab.ece.ntua.gr
Βαγγέλης Κούκης, vkoukis@cslab.ece.ntua.gr, vkoukis@arrikto.com

24 Integrate caching of Raw and Kale-produced Kubeflow Pipelines with MLMD on Kubeflow

The development of Machine Learning Pipelines is a tedious, time-consuming process.

The Kubeflow Project [1] is an open source project to develop a platform for building, training, and serving Machine Learning Models on Kubernetes [2].

Tools like Arrikto Kale [3] aim to simplify the work of the data scientist by offering a simple, intuitive interface to create and orchestrate complex pipelines using Python and Jupyter Notebooks [4]. Kale starts from Python code, either as a script, or as a notebook, and converts it into distinct steps of a Kubeflow Pipeline. The Pipeline then runs on Kubeflow, on a Kubernetes cluster.

In their day-to-day work, a data scientist often runs multiple versions of the same ML pipeline in quick succession: At every iteration, they may make changes to only part of the pipeline, leaving the rest of its steps intact. Since each pipeline run may take hours or even days, any opportunity to re-use step outputs from previous pipeline runs whenever this doesn't impact correctness can lead to significant reduction in total execution time.

For example, if we have already executed a 4-step pipeline $A \rightarrow B \rightarrow C \rightarrow D$, we can accelerate the execution of a second pipeline $A \rightarrow B \rightarrow C \rightarrow D'$ by re-using the cached outputs of all steps up to C and only having to run D'.

The objective of this diploma thesis is to unify current caching approaches across Kale-produced pipelines and raw Kubeflow Pipelines so they both use a single, shared cache mechanism over Kubeflow's metadata store called ML Metadata - MLMD [5]. Completing this project successfully requires building in-depth knowledge of Kubeflow Pipelines, Kubernetes, Argo [6], Kale, and Arrikto Rok [7].

The end goal of this project is to have seamless re-use of cached outputs across runs of Kale-produced pipelines and raw Kubeflow pipelines.

Σχετική Βιβλιογραφία:

1. https://kubeflow.org
2. https://kubernetes.io
3. https://docs.arrikto.com/user/kale/index.html
4. https://jupyter.org/
5. https://github.com/google/ml-metadata/blob/master/g3doc/get_started.md
6. https://argoproj.github.io/argo-workflows/

7. https://docs.arrikto.com/introduction/ekf-features.html#rok

Σχετικά Μαθήματα: Εργαστήριο Λειτουργικών Συστημάτων, Κατανεμημένα Συστήματα

Επικοινωνία:| Γεώργιος Γκούμας, goumas@cslab.ece.ntua.gr
Βαγγέλης Κούκης, vkoukis@cslab.ece.ntua.gr, vkoukis@arrikto.com

25 Port a Lightweight Kubeflow Distribution to WSL 2 on Windows with GPU Support

The Kubeflow Project [1] is an open source project to develop a platform for building, training, and serving Machine Learning Models on Kubernetes [2].

MiniKF [3] is a single-node distribution of Kubeflow which combines Minikube – a lightweight, single-node Kubernetes implementation [4] – with Arrikto Rok [5], a data management solution for Kubernetes clusters, and Kubeflow.

MiniKF originally ran inside a distinct Virtual Machine, for reasons of portability, using Vagrant [6] and VirtualBox [7]. However, using VirtualBox has significant impact in performance, and is often the source of compatibility issues on Windows [8] and macOS [9].

On the other hand, Microsoft has invested considerable effort into bringing full-featured support for Linux applications to the Windows ecosystem, with the Windows Subsystem for Linux [10]. WSL 2 in particular has brought major new features, including much improved performance, and support for exposing GPUs to Linux applications [11].

This makes it possible to eliminate Vagrant and VirtualBox from the stack. This diploma thesis aims to run MiniKF directly on WSL 2, as one more containerized Linux distribution. This approach is similar to running Kind / Kubernetes-in-Docker [12].

To do this, we will understand and document the structure of MiniKF and the interaction of all of its components, then explore ways to run each one of them individually, on WSL 2. We expect a lot of smaller or bigger problems along the way, which we will expose to the relevant upstream communities, and fix with upstream Pull Requests. To complete this project successfully you will have to get your hands dirty, achieve deep understanding of different components in a UNIX system, and build a strong set of DevOps skills.

The end goal is to support seamless, single-click deployment of MiniKF on Windows via the Microsoft Store, and, optionally, to extend this support into any environment which supports Docker containers.

Σχετική Βιβλιογραφία:

1. https://kubeflow.org/
2. https://kubernetes.io/
3. https://www.arrikto.com/blog/kubeflow/news/minikf-a-fast-and-easy-way-to-deploy-kubeflow-on-you
4. https://minikube.sigs.k8s.io/docs/start/
5. https://docs.arrikto.com/introduction/ekf-features.html
6. https://www.vagrantup.com/
7. https://www.virtualbox.org/

8. https://learn.microsoft.com/en-us/troubleshoot/windows-client/application-management/virtualization-apps-not-work-with-hyper-v
9. https://apple.stackexchange.com/questions/410529/virtualbox-does-not-work-after-upgrading-to-bi
10. https://learn.microsoft.com/en-us/windows/wsl/install
11. https://learn.microsoft.com/en-us/windows/ai/directml/gpu-cuda-in-wsl
12. https://kind.sigs.k8s.io/

Σχετικά Μαθήματα: Εργαστήριο Λειτουργικών Συστημάτων, Κατανεμημένα Συστήματα

Επικοινωνία: Γεώργιος Γκούμας, goumas@cslab.ece.ntua.gr
Βαγγέλης Κούκης, vkoukis@cslab.ece.ntua.gr, vkoukis@arrikto.com

26 Understand, Document, Train and Serve a Well-Known Image Generation ML Model with Kubeflow

The Kubeflow Project [1] is an open source project to develop a platform for building, training, and serving Machine Learning Models on Kubernetes [2].

Stable Diffusion is a deep learning, generative model released by Stability AI [3]. One of its main functions is to produce unique synthetic images from text prompts. Its source code is publicly available [4], and researchers from LMU have trained it [5] on a subset of the LAION-5B dataset [6], a 250TB dataset comprising 5.6 billion images.

This diploma thesis aims to understand the architecture of Stable Diffusion, a latent diffusion model [7], to document its distinct components, use Kubeflow to fine-tune it on a custom dataset, and expose it as a service on the Web.

In the process, we will explore methods of accelerating training and serving the model on different architectures, including making code changes to trade-off accuracy for performance [8], running on non-NVIDIA GPUs [9], and parallelizing the model on Kubernetes with Kubeflow and the Training Operator [10].

The expected outcome is an end-to-end understanding of the architecture of Stable Diffusion and similar models, and of the challenges involved in training and serving them in production environments.

Σχετική Βιβλιογραφία:

1. https://kubeflow.org
2. https://kubernetes.io
3. https://stability.ai/blog/stable-diffusion-public-release
4. https://github.com/CompVis/stable-diffusion
5. https://huggingface.co/CompVis/stable-diffusion
6. https://laion.ai/blog/laion-5b
7. https://lilianweng.github.io/posts/2021-07-11-diffusion-models/#ldm

8. https://www.reddit.com/r/MachineLearning/comments/xuojma/n_stable_diffusion_reaches_new_record_with
9. https://twitter.com/pcuenq/status/1567927480253808647
10. https://github.com/kubeflow/training-operator

Σχετικά Μαθήματα: Εργαστήριο Λειτουργικών Συστημάτων, Κατανεμημένα Συστήματα

Επικοινωνία: Γεώργιος Γκούμας, goumas@cslab.ece.ntua.gr
Βαγγέλης Κούκης, vkoukis@cslab.ece.ntua.gr, vkoukis@arrikto.com

27 Explore Cross-Vendor GPU-based Machine Learning with DirectML over DirectX on WSL 2, extend to native Linux

Training and serving deep learning models often requires substantial compute, memory, and storage resources. For example, serving the Stable Diffusion model requires ∼7GB VRAM and an NVIDIA GPU [1].

It is possible to modify model code so it runs on a CPU [2], but this comes with a significant impact in performance; running Stable Diffusion on an Intel CPU takes ∼10 minutes to produce an image, without any GPU acceleration, although most Intel CPUs come with an integrated GPU [3].

DirectX has become the ubiquitous low-level graphics API which enables portability for video games and other GPU-intensive applications across GPUs in the Windows ecosystem. On the other hand, it is not trivial to run Stable Diffusion or other similar deep learning models on GPUs others than the one they have been designed for.

Microsoft has developed the low-level DirectML API [4], specifically targeting ML use cases, which uses DirectX underneath, so it can work across different GPUs. It has invested considerable effort into bringing full-featured support for Linux applications to the Windows ecosystem, with the Windows Subsystem for Linux [5].

The combination of DirectML with WSL 2 opens the way for running Machine Learning frameworks like Tensorflow [6] and PyTorch [7] on GPUs from different vendors [8].

This diploma thesis aims to explore the use of DirectML over WSL 2 for real-world ML use cases, understand and document its limitations, evaluate its performance across a variety of GPUs, and compare with the current CUDA-based approaches [9].

As an extension, it would be very interesting to explore the question of cross-vendor GPU-based training and inference for ML models under native Linux. There have been efforts to run DirectX on Linux via translation layers [10], and there are native GPU APIs on Linux [11]. The end goal is to run PyTorch across GPUs natively on Linux, evaluate its performance, and integrate support for it into Kubeflow [12].

Σχετική Βιβλιογραφία:

1. https://stability.ai/blog/stable-diffusion-public-release
2. https://towardsdatascience.com/how-to-generate-stunning-art-on-your-laptop-using-ai-a28192cbb49
3. https://en.wikipedia.org/wiki/Intel_Graphics_Technology
4. https://learn.microsoft.com/en-us/windows/ai/directml/dml-intro

5. https://learn.microsoft.com/en-us/windows/wsl/install
6. https://devblogs.microsoft.com/windowsai/directml-plugin-for-tensorflow-2-is-here/
7. https://devblogs.microsoft.com/windowsai/introducing-pytorch-directml-train-your-machine-learni
8. https://blogs.windows.com/windowsdeveloper/2021/01/28/bring-your-ai-to-any-gpu-with-directml/
9. https://github.com/microsoft/DirectML/issues/108
10. https://github.com/ValveSoftware/Proton
11. https://www.vulkan.org/
12. https://kubeflow.org

Σχετικά Μαθήματα: Εργαστήριο Λειτουργικών Συστημάτων, Κατανεμημένα Συστήματα

Επικοινωνία: Γεώργιος Γκούμας, goumas@cslab.ece.ntua.gr
Βαγγέλης Κούκης, vkoukis@cslab.ece.ntua.gr, vkoukis@arrikto.com

28 Evaluate the Dask and Dask-ML frameworks and Integrate them into Kubeflow

Developing Machine Learning Pipelines is a hard and time-consuming process. The most computationally-demanding part is training the Machine Learning model. To produce highly accurate models it is often necessary to train a very large amount of input data, in the order of GBs or even TBs, a process that may require days or even weeks of computations. For this reason, we are interested in exploring techniques for distributing the load to multiple processors on a large number of nodes (scale-out), essentially creating a distributed system.

Dask [1] is a flexible library for parallel computing in Python. It targets two main bottlenecks which are often the reason for needing to scale to more than one compute nodes: CPU, since running on multiple nodes means we can take advantage of multiple cores, and RAM, since the aggregate amount of RAM scales with the number of nodes, so we can keep much bigger datasets in memory.

It comprises two parts:

Dynamic task scheduling to enable multiple threads on multiple nodes to work on different parts of the input data in parallel

Re-implementations of collections like NumPy arrays [2] and Pandas dataframes [3], which mimic the existing APIs but distribute their data across multiple nodes, so they can operate in parallel on datasets that don't fit in the memory of a single node.

Dask-ML [4] provides scalable Machine Learning in Python using Dask alongside popular ML libraries like Scikit-Learn, XGBoost, and others.

The Kubeflow Project [5] is an open source project to develop a platform for building, training, and serving Machine Learning Models on Kubernetes [6].

Dask and Kubeflow share a common goal, supporting scalable Machine Learning across multiple nodes, but follow different approaches: Dask focuses on interactive workloads and schedules tasks dynamically

on a variety of workers, including Kubernetes Pods, while Kubeflow builds directly on Kubernetes and benefits from its production-quality support for autoscaling.

The goal of this diploma thesis is to bridge the worlds of Dask and Kubeflow by integrating Dask seamlessly as one more supported Kubeflow component. To do this, we need to understand the architecture of Dask and Kubeflow, compare their design decisions, document their strengths and limitations, then explore ways in which integrating Dask with Kubeflow would be beneficial to the end user. We will propose design changes to both upstream communities, implement them as Pull Requests, evaluate community feedback, and see them through all the way to the final merge.

There has already been interest in Dask/Kubeflow integration in the community, e.g., [7]. This project will need changes both in the backend, e.g., Kubernetes CRDs and controllers [8] to align the semantics of Kubeflow with those of Dask, and in the frontend, Jupyter Notebooks [9], to expose Dask seamlessly in the Kubeflow UI. It will help you build experience across the ML stack, all the way from pods running on Kubernetes to JavaScript running in the browser.

We expect Dask will provide a more intuitive way to express parallelism directly in Python, from within a Jupyter Notebook, without having to create Kubernetes manifests in YAML directly.

The end goal is to have Dask as a fully-supported, documented component in Kubeflow.

Σχετική Βιβλιογραφία:

1. `https://docs.dask.org/`
2. `https://numpy.org/doc/stable/reference/generated/numpy.array.html`
3. `https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html`
4. `https://ml.dask.org/`
5. `https://kubeflow.org`
6. `https://kubernetes.io`
7. `https://developer.nvidia.com/blog/accelerating-etl-on-kubeflow-with-rapids/`
8. `https://kubernetes.io/docs/concepts/extend-kubernetes/operator`
9. `https://jupyter.org/`

Σχετικά Μαθήματα: Εργαστήριο Λειτουργικών Συστημάτων, Κατανεμημένα Συστήματα

Επικοινωνία: Γεώργιος Γκούμας, goumas@cslab.ece.ntua.gr
Βαγγέλης Κούκης, vkoukis@cslab.ece.ntua.gr, vkoukis@arrikto.com

29 Explore the Ray ML Framework and Integrate it into Kubeflow

Ray [1] is a general-purpose distributed computing framework with a rich set of libraries for large-scale data processing, model training, and model serving.

The Kubeflow Project [2] is an open source project to develop a platform for building, training, and serving Machine Learning Models on Kubernetes [3].

The subject of this diploma thesis is to first understand the architecture of both Ray and Kubeflow, compare their design decisions, document their strengths and limitations, and then explore ways in

which integrating Ray with Kubeflow would be beneficial to the end user. We will propose design changes to both upstream communities, implement them as Pull Requests, evaluate community feedback, and see them through all the way to the final merge.

There has already been interest in this direction in the community [4]. This project will need changes both in the backend, e.g., Kubernetes CRDs and controllers [5] to align the semantics of Kubeflow with those of Ray, and in the frontend, Jupyter Notebooks [6], to expose Ray seamlessly in the Kubeflow UI. It will help you build experience across the ML stack, all the way from pods running on Kubernetes to JavaScript running in the browser.

The end goal is to have Ray as a fully-supported, documented component in Kubeflow.

Σχετική Βιβλιογραφία:

1. `https://docs.ray.io/en/master/`
2. `https://kubeflow.org`
3. `https://kubernetes.io`
4. `https://cloud.google.com/blog/products/ai-machine-learning/build-a-ml-platform-with-kubeflow-an`
5. `https://kubernetes.io/docs/concepts/extend-kubernetes/operator`
6. `https://jupyter.org/`

Σχετικά Μαθήματα: Εργαστήριο Λειτουργικών Συστημάτων, Κατανεμημένα Συστήματα

Επικοινωνία: Γεώργιος Γκούμας, goumas@cslab.ece.ntua.gr
Βαγγέλης Κούκης, vkoukis@cslab.ece.ntua.gr, vkoukis@arrikto.com

30 Integrate the MergerFS Union Filesystem with Arrikto Rok and Explore a Kernelspace Implementation

The Linux kernel is a monolithic OS kernel [1] that powers a wide range of systems, all the way from embedded systems like smartphones to literally every single system in the TOP 500 list [2].

Traditionally, filesystems have run as part of the kernel, but Linux offers FUSE [3], a mechanism to run filesystem code in userspace, which simplifies development considerably, enables rapid prototyping, improves isolation and stability, supports the use of different external libraries, and proceeds at a much higher pace compared to merging filesystem code into the kernel. However, FUSE comes with a significant performance penalty [4] since it has to switch between kernel and user contexts in the critical path, and this becomes even more apparent when working with super fast, very low-latency local storage over NVMe [5, 6].

On the other hand, union filesystems [7] like UnionFS [8], OverlayFS [9], and MergerFS [10] combine filesystem instances to enable powerful new use cases, including supporting read-write trees over read-only media (live CDs), read-write trees over read-only flash images (booting on embedded systems like OpenWRT), and aggregate storage pools.

Arrikto Rok [11] is a data management solution for Kubernetes [12] which enables super fast, low-latency data access over local NVMe SSDs with advanced data services [thin snapshots, fast clones, efficient peer-to-peer synchronization over a decentralized network].

The goal of this diploma thesis is to deploy MergerFS, evaluate its performance over NVMe devices, and integrate it with Arrikto Rok to support multi-TB filesystem hierarchies in the cloud.

Depending on the results of the performance evaluation, a parallel effort can be creating a proof-of-concept implementation of MergerFS in the Linux kernel, comparing its performance with the userspace implementation, and using the results to advocate for its inclusion in the mainline Linux kernel.

The end goal is to support multi-TB virtual filesystems over local NVMe devices, and a low-latency kernel-based data path.

Σχετική Βιβλιογραφία:

1. `https://groups.google.com/g/comp.os.minix/c/wlhw16QWltI`
2. `https://www.top500.org/statistics/details/osfam/1/`
3. `https://www.kernel.org/doc/html/latest/filesystems/fuse.html`
4. `https://www.usenix.org/system/files/conference/fast17/fast17-vangoor.pdf`
5. `https://www.arrikto.com/tutorials/data-management/why-your-cassandra-needs-local-nvme-and-rok/`
6. `https://www.theregister.com/2019/02/22/azure_nvme_flash_drives_hyperv_virtual_machines/`
7. `https://unix.stackexchange.com/questions/382326/unionfs-vs-aufs-vs-overlayfs-vs-mhddfs-which-on`
8. `https://unionfs.filesystems.org/`
9. `https://docs.kernel.org/filesystems/overlayfs.html`
10. `https://github.com/trapexit/mergerfs`
11. `https://docs.arrikto.com/introduction/ekf-features.html`
12. `https://kubernetes.io`

Σχετικά Μαθήματα: Εργαστήριο Λειτουργικών Συστημάτων, Κατανεμημένα Συστήματα

Επικοινωνία: Γεώργιος Γκούμας, goumas@cslab.ece.ntua.gr
Βαγγέλης Κούκης, vkoukis@cslab.ece.ntua.gr, vkoukis@arrikto.com

31 Explore the `io_uring` and `ublk` Linux Kernel Mechanisms and Integrate them with Arrikto Rok

`io_uring` [1] is an asynchronous system call interface for the Linux kernel, which has been merged in version 5.1 by Jens Axboe, the current Linux kernel maintainer of the kernel block layer for the Linux kernel.

Arrikto Rok [2] is a generic storage data management solution for Kubernetes which enables super fast, low-latency data access over local NVMe SSDs with advanced data services [thin snapshots, fast clones, efficient peer-to-peer synchronization over a decentralized network]. It integrates with Kubernetes via the Container Storage Interface (CSI). [3]

The objective of this diploma thesis is to evaluate the improvement in I/O performance when using `io_uring` over local NVMe devices [4], which offer very low, microsecond-level I/O latencies, and to

extend Arrikto Rok so it uses a userspace block device driver implemented via `ublk` [5, 6] instead of the current approach, which is based on the Linux SCSI Target [7].
The project will require building in-depth understanding of the Linux kernel I/O path, the design principles behind modern I/O protocols like SCSI and NVMe, and the performance characteristics of shared-memory communication. It will take considerable hands-on experimentation with the low-level internals of the kernel to complete.
The end goal is seamless integration of an `io_uring`-based I/O mechanism into Arrikto Rok and a complete performance evaluation [throughput, latency, CPU utilization] before and after the proposed change.

Σχετική Βιβλιογραφία:

1. `https://kernel.dk/io_uring.pdf`
2. `https://docs.arrikto.com/introduction/ekf-features.html#rok`
3. `https://kubernetes-csi.github.io/docs/`
4. `https://www.arrikto.com/tutorials/data-management/why-your-cassandra-needs-local-nvme-and-rok/`
5. `https://lwn.net/Articles/900690/`
6. `https://docs.kernel.org/block/ublk.html`
7. `http://linux-iscsi.org/wiki/LIO`

Σχετικά Μαθήματα: Εργαστήριο Λειτουργικών Συστημάτων, Κατανεμημένα Συστήματα

Επικοινωνία: Γεώργιος Γκούμας, goumas@cslab.ece.ntua.gr
Βαγγέλης Κούκης, vkoukis@cslab.ece.ntua.gr, vkoukis@arrikto.com