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 2022-23

Ι. Παράλληλα Συστήματα

1 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.

1.1 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
Νικέλα Παπαδοπούλου, nikela@cslab.ece.ntua.gr, 210-772-2279
Γεώργιος Γκούμας, goumas@cslab.ece.ntua.gr, 210-772-2402

2 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).

2.1 Αξιολόγηση και σύγκριση εργαλείων για τον προγραμματισμό GPGPU.

Μαζί με την εξέλιξη των δυνατοτήτων των GPU ήρθε μια μεγάλη ποικιλία εργαλείων λογισμικού, μεταγλωττιστών και προτύπων για τον προγραμματισμό GPU, με σκοπό την αξιοποίηση των δυνατοτήτων τους από τους προγραμματιστές και τους μηχανικούς απόδοσης. Έτσι, ενώ 10 χρόνια πριν ο προγραμματισμός GPGPU ήταν σχεδόν ισοδύναμος με προγραμματισμό σε CUDA, σήμερα υπάρχει μια μεγάλη ποικιλία επιλογών (CUDA, OpenMP, OpenCl, OpenACC, oneAPI) και μια μετατόπιση του ενδιαφέροντος προς αυτές. Αυτά τα εργαλεία προσεγγίζουν τον προγραμματισμό GPGPU με διαφορετικούς τρόπους, θέτοντας διαφορετικούς στόχους όσον αφορα την ευκολία προγραμματισμού, το έυρος εφαρμογής και την απόδοση, θυσιάζοντας πάντα κάτι. Ο σκοπός της διπλωματικής αυτης είναι η εξοικείωση με ένα υποσύνολο αυτών των διαφορετικών εργαλείων και η αξιολογήση τους σε διάφορους αλγορίθμους, προκειμένου να διερευνηθούν, να εκτιμηθούν και να αξιολογηθούν οι δυνατότητες, οι αδυναμίες και τα περιθώρια του καθενός, καθώς και αποτελεσματικών μεθόδων για την έυρεση αυτων. Ορισμένα πιο συγκεκριμένα θέματα θα μπορούσαν να περιλαμβάνουν:

- Ανάπτυξη τεχνικών για τη μεταφορά κώδικα μεταξύ αυτών των εργαλείων.
- Εύρεση συγκεκριμένων σημείων συμφόρησης/αδυναμιών σε αυτά και πρόταση επιλογών για την αποφυγή τους.
- Μοντελοποίηση της απόδοσής τους.

Σχετικά Μαθήματα: Συστήματα Παράλληλης Επεξεργασίας

Σχετική Βιβλιογραφία:

1. https://docs.nvidia.com/cuda/
2. https://www.openmp.org/
3. https://www.khronos.org/opencl/
4. https://www.openacc.org/
5. https://www.oneapi.com/

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

3 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.

3.1 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 με
- 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

4 Πρόβλεψη επίδοσης παράλληλων εφαρμογών

4.1 Μοντελοποίηση επίδοσης παράλληλων εφαρμογών

Η υλοποίηση και βελτιστοποίηση παράλληλων εφαρμογών σε μοντέρνες αρχιτεκτονικές υψηλής επίδοσης είναι ιδιαίτερα κοστοβόρα και απαιτητική. Δεδομένου ότι το προς παραλληλοποίηση πρόβλημα μπορεί να έχει εναλλακτικούς αλγορίθμους και κάθε αλγόριθμος διαφορετικές προσεγγίσεις παραλληλοποίησης, η λογική "trial and error", δηλαδή η υλοποίηση όλων των διαφορετικών εκδόσεων και η πειραματική αξιολόγηση της επίδοσής τους δεν είναι ο πλέον ενδεδειγμένος τρόπος. Στόχος της συγκεκριμένης διπλωματικής εργασίας είναι η κατάστρωση μοντέλων πρόβλεψης επίδοσης παράλληλων προγραμμάτων χωρίς να είναι αναγκαία η υλοποίησή τους. Στο πλαίσιο της εργασίας ο φοιτητής θα έχει στη διάθεσή του σειριακές υλοποιήσεις των εφαρμογών υπό εξέταση τις οποίες θα αναλύσει και στη συνέχεια θα καταστρώσει μοντέλα πρόβλεψης επίδοσης υποστηριζόμενα από μετροπρογράμματα (benchmarks). Ο έλεγχος των μοντέλων θα γίνει με σύγκριση των πραγματικών παράλληλων υλοποιήσεων που και αυτές θα είναι μέρος της διπλωματικής εργασίας.

Σχετικά Μαθήματα: Συστήματα Παράλληλης Επεξεργασίας

Επικοινωνία: Γεώργιος Γκούμας, nikela@cslab.ece.ntua.gr, 210-772-2402
Νικέλα Παπαδοπούλου, nikela@cslab.ece.ntua.gr, 210-772-2495

4.2 Τεχνικές προβολής της επίδοσης παράλληλων εφαρμογών σε υπολογιστικά συστήματα μεγάλης κλίμακας

Η πρόβλεψη της επίδοσης των παράλληλων εφαρμογών που εκτελούνται σε υπερυπολογιστές είναι κρίσιμη για το σχεδιασμό των συστημάτων επόμενης γενιάς. Ένα από τα σημαντικότερα ερωτήματα που καλούνται να απαντήσουν τα διάφορα μοντέλα πρόβλεψης είναι η επίδοση των εφαρμογών σε συστήματα μεγαλύτερης κλίμακας ή συστήματα με διαφορετικά αρχιτεκτονικά χαρακτηριστικά από τα υπάρχοντα, δηλαδή η προβολή της επίδοσης των εφαρμογών (performance extrapolation). Το συγκεκριμένο ερώτημα γίνεται επιτακτικό καθώς βρισκόμαστε στη φάση της μετάβασης από την εποχή των επιδόσεων της τάξης των PetaFLOPs στην εποχή των επιδόσεων της τάξης των ExaFLOPs. Στη βιβλιογραφία έχουν προταθεί τεχνικές για την κατάστρωση μοντέλων προβολής τόσο της επίδοσης, σε όρους χρόνου εκτέλεσης, όσο και των χαρακτηριστικών που καθορίζουν την επίδοση μιας εφαρμογής (π.χ. FLOPs, bytes, memory footprint), ως συναρτήσεων του αριθμού των πυρήνων/επεξεργαστών και του μεγέθους της εισόδου των εφαρμογών. Στην παρούσα διπλωματική θα μελετήσουμε την επέκταση αυτών των τεχνικών σε τρεις κατευθύνσεις: α) στην μοντελοποίηση χαρακτηριστικών εφαρμογών που η είσοδός τους είναι πολυπαραμετρική (π.χ. γράφοι αντί πινάκων), β) στη μοντελοποίηση του χρόνου εκτέλεσης ως συνάρτηση των παραπάνω χαρακτηριστικών, και γ) στην παραμετροποίηση αυτών των μοντέλων για να καταστεί εφικτή η μεταφορά τους από ένα σύστημα σε ένα άλλο.

Σχετική Βιβλιογραφία:

- Calotoiu, A., Beckinsale, D., Earl, C. W., Hoefler, T., Karlin, I., Schulz, M., Wolf, F. (2016, September). Fast multi-parameter performance modeling. In 2016 IEEE International Conference on Cluster Computing (CLUSTER) (pp. 172-181). IEEE.
- Ritter, M., Calotoiu, A., Rinke, S., Reimann, T., Hoefler, T., Wolf, F. (2020, May). Learning cost-effective sampling strategies for empirical performance modeling. In 2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS) (pp. 884-895). IEEE.

Σχετικά Μαθήματα: Συστήματα Παράλληλης Επεξεργασίας

Επικοινωνία: Γεώργιος Γκούμας, goumas@cslab.ece.ntua.gr, 210-772-2402
Νικέλα Παπαδοπούλου, nikela@cslab.ece.ntua.gr, 210-772-2495

5 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.

5.1 Διαχείριση Πόρων σε Συστήματα Μεγάλης Κλίμακας

5.1.1 Χαρακτηρισμός Εφαρμογών με χρήση intruding 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.
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

5.1.2 Διαχείριση πόρων σε Kubernetes clusters με χρήση resource managers της Intel

Η χρήση containers και του Kubernetes ως πλατφόρμας διαχείρισης τους φαίνεται να επικρατεί στη βιομηχανία τα τελευταία χρόνια έναντι της χρήσης VMs, επομένως η μελέτη των επιλογών που προσφέρονται για την αποδοτική εκτέλεση πολλών containers σε ένα υπολογιστικό κόμβο είναι αναμφισβήτητα η επόμενη πρόκληση. Η Intel αναπτύσσει ήδη Resource Managers (Platform Aware Scheduling, CRI Resource Manager) προς αυτή την κατεύθυνση. Στόχος της διπλωματικής είναι η εξερεύνηση των δυνατοτήτων που προσφέρουν αυτά τα εργαλεία και η χρήση τους για τη βελτίωση της επίδοσης διάφορων μετροπρογραμμάτων.
Σχετική Βιβλιογραφία:

1. Intel Platform Aware Scheduling

2. Intel CRI Resource Manager

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

5.2 Βέλτιστη αξιοποίηση υπολογιστικών πόρων σε συστήματα μεγάλης κλίμακας

Σε συστήματα μεγάλης κλίμακας υψηλών υπολογιστικών επιδόσεων, οι αλγόριθμοι δρομολόγησης εργασιών, όπως ο Back-Filling, για να λάβουν αποφάσεις, αξιοποιούν την πληροφορία που τους παρέχουν οι χρήστες, οι οποίοι, καθώς υποβάλουν την εργασία τους, αιτούνται τους απαραίτητους υπολογιστικούς πόρους (κόμβους, πυρήνες, μνήμη, επιταχυντές) και παρέχουν μια εκτίμηση για το χρόνο ολοκλήρωσης των εργασιών τους. Όσο πιο ακριβής είναι αυτή η πληροφορία, τόσο καλύτερη είναι η αξιοποίηση του συστήματος (throughput) και η ικανοποίηση των χρηστών (χαμηλοί χρόνοι αναμονής). Ωστόσο, καθώς οι εφαρμογές που εκτελούνται συχνά περιλαμβάνουν χιλιάδες γραμμές κώδικα και χρησιμοποιούν αρκετές επιπλέον βιβλιοθήκες και υπολογιστικά πακέτα, οι χρήστες δεν είναι πάντα σε θέση να εκτιμήσουν σωστά την επίδοση της εφαρμογής τους και τον αναμενόμενο χρόνο εκτέλεσής της. Έτσι, υποβάλουν εκτιμήσεις που είναι ανακριβείς ως προς τους ζητούμενους πόρους και καταλήγουν σε σπατάλη πόρων (π.χ. ο χρήστης θα μπορούσε να είχε λάβει αντίστοιχο χρόνο εκτέλεσης με λιγότερους υπολογιστικούς πόρους). Σε σχέση με τους χρόνους εκτέλεσης, οι χρήστες κατά κανόνα υπερεκτιμούν τον αναμενόμενο χρόνο εκτέλεσης της εφαρμογής τους.

Στην παρούσα διπλωματική, θα μελετήσουμε την επίδραση των εκτιμήσεων των χρηστών στην επίδοση του συστήματος και θα επεκτείνουμε υπάρχοντες αλγορίθμους χρονοδρομολόγησης για συστήματα μεγάλης κλίμακας με δυνατότητες διάδρασης με το χρήστη, για την εκτίμηση και επιλογή των κατάλληλων πόρων σε σχέση με την εργασία του χρήστη, με στόχο τη βέλτιστη αξιοποίηση των πόρων του συστήματος και τη μεγιστοποίηση της απόδοσης του συστήματος.

Επικοινωνία: Νικέλα Παπαδοπούλου, nikela@cslab.ece.ntua.gr
Γεώργιος Γκούμας, goumas@cslab.ece.ntua.gr, 210-772-2402

5.3 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.

5.3.1 Profiling MPI εφαρμογών και υλοποίηση ενσωμάτωσης σε εικονικό σύστημα διαχείρισης πόρων με χρήση Python (ατομική ή συνεργατική διπλωματική εργασία)

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-) ή χρήση δεικτών επίδοσης (performance counters) για την ταξινόμησή τους (classification). Η γνώση των χαρακτηριστικών μια εφαρμογής αποτελεί κομβικό σημείο για την περαιτέρω εφαρμογή αλγορίθμων χρονοδρομολόγησής της και/ή διανομής υπολογιστικών πόρων. Για το λόγω αυτό υπάρχουν αρκετά εργαλεία που μετρούν διάφορα χαρακτηριστικά των εφαρμογών, π.χ. μνήμη RAM που δεσμεύτηκε, cache misses, επεξεργαστική επίδοση με Instructions per clock (IPC), πλήθος και μέγεθος μηνυμάτων όταν μιλάμε για MPI εφαρμογή κτλ.
Σκοπός της παρούσας διπλωματικής είναι η ενσωμάτωση εργαλείων profiling MPI εφαρμογών (όπως: INTEL VTUNE, mpiP, Scalasca) και η εξαγωγή των αποτελεσμάτων με ενιαίο τρόπο στο υπάρχον λογισμικό που αναπτύσσεται στο εργαστήριο μας, το vRJMS (virtual Resource and Job Management System).

5.3.2 Ανάπτυξη εργαλείου για την εκτέλεση του διαχειριστή πόρων OAR3 εικονικά σε δουλειά του διαχειστιή 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).

5.3.3 Μελέτη και αξιολόγηση αλγορίθμων χρονοδρομολόγησης MPI εργασιών σε πραγματικό περιβάλλον διαχειριστή πόρων (ατομική ή συνεργατική διπλωματική εργασία)

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 στον ίδιο κόμβο), οδηγεί σε αποτελεσματικότερη χρήση των υπολογιστικών πόρων. Από την άλλη πλευρά, η επιλογή των εφαρμογών που θα εκτελεστούν ταυτόχρονα λαμβάνει σημαντικό ρόλο για την επίδοση που θα πετύχουν λόγω των race conditions που θα αναπτυχθούν ανάλογα με τους πόρους που ζητά η εκάστοτε εφαρμογή.
Σκοπός της διπλωματικής αποτελεί η πειραματική μελέτη και αξιολόγηση αλγορίθμων χρονοδρομολόγησης με χρήση υπαρχόντων benchmarks σε πραγματικό περιβάλλον διαχειριστή πόρων. Συγκεκριμένα, θα μελετηθεί (i) η κλιμακωσιμότητα των benchmarks σ' ένα cluster, (ii) η επίδοση για διαφορετικού είδους αναθέσεις πόρων, (iii) τα race conditions που αναπτύσσονται σε διάφορα είδη εφαρμογών (π.χ. memory bounded, compute bounded), (iv) η αξιολόγηση και σύγκριση αλγορίθμων χρονοδρομολόγησης για τα συγκεκριμένα benchmarks με ή χωρίς τεχνικές co-scheduling.

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/vRJMS`
9. `http://artemis.cslab.ece.ntua.gr:8080/jspui/handle/123456789/18315`

Λογισμικό του εργαστηρίου: vRJMS (virtual Resource and Job Management System)

Υλοποίηση εικονικού διαχειριστή πόρων για MPI εφαρμογές σε υπερυπολογιστικά συστήματα. Περισσότερες πληροφορίες στη σελίδα wiki του εργαλείου:

- `https://github.com/cslab-ntua/vRJMS/wiki`

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

6 Ανάλυση δεδομένων υπερυπολογιστικών συστημάτων

Οι διαχειριστές των σύγχρονων υπερυπολογιστικών συστημάτων έχουν τη δυνατότητα συλλογής σημαντικής πληροφορίας για τη συμπεριφορά των εφαρμογών που εκτελούνται στο σύστημα, τα αιτήματα των χρηστών και την ποιότητα υπηρεσίας που λαμβάνουν, αλλά και τη συνολική κατάσταση του συστήματος. Η συλλογή ιστορικών δεδομένων του συστήματος δίνει δυνατότητες για τη βελτίωση της χρήσης του συστήματος, αφού η επεξεργασία της μπορεί να συμβάλλει στη μείωση του χρόνου αναμονής από την πλευρά των χρηστών, στη βελτίωση των πολιτικών δέσμευσης πόρων και χρονοδρομολόγησης των εργασιών, αλλά και στην ανθεκτικότητα του συστήματος σε σφάλματα. Η παρακάτω εργασία εστιάζει στην ανθεκτικότητα του συστήματος σε σφάλματα.

6.1 Υλοποίηση συστήματος πρόβλεψης σφαλμάτων σε συστήματα υψηλής επίδοσης με τεχνικές μηχανικής μάθησης

Οι σημερινοί υπερυπολογιστές αντιμετωπίζουν συχνά σφάλματα σε καθημερινή βάση. Παρόλο που υπάρχουν πολλές προσεγγίσεις ανάκτησης (recovery), όπως το checkpoint/restart, κατά την ανάκτηση των διάφορων components από σφάλματα, χάνεται σημαντική υπολογιστική ισχύς. Τα νέα συστήματα κλίμακας exascale προβλέπεται ότι αντιμετωπίζουν ακόμα πιο ψηλά ποσοστά σφαλμάτων, λόγω του αυξημένου πλήθους των components που τα συνθέτουν. Επομένως, είναι σημαντικό να υπάρχουν καλώς ορισμένοι δείκτες σφαλμάτων (failure indicators). Τα αρχεία καταγραφής του συστήματος (logs) αποτελούνται από κείμενο που κρύβει πληροφορίες για την εκάστοτε "υγεία" του συστήματος. Συνεπώς, η αποτελεσματική πρόβλεψη σφαλμάτων του συστήματος μέσω των logs θα μπορούσε να επιτρέψει προληπτικούς μηχανισμούς ανάκτησης και άρα αύξηση της αξιοπιστίας (reliability). Στην παρούσα διπλωματική, θα μελετήσουμε την αξιοποίηση των logs και συγκεκριμένα του κειμένου που αυτά περιέχουν, θα αναλύσουμε τα κείμενα αυτά και θα αναπτύξουμε ένα σύστημα πρόβλεψης σφαλμάτων σε κόμβους του υπερυπολογιστή με τη χρήση Transformers. Για την χρήση των Transformers, η υλοποίηση θα γίνει τόσο χρησιμοποιώντας κάποιο pre-trained μοντέλο όσο και φτιάχνοντας το δικό μας Transformers μοντέλο.

Σχετική Βιβλιογραφία:

- Das, A., Mueller, F., Siegel, C., Vishnu, A. (2018, June). Desh: deep learning for system health prediction of lead times to failure in hpc. In Proceedings of the 27th International Symposium on High-Performance Parallel and Distributed Computing (pp. 40-51).
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. In Advances in neural information processing systems (pp. 5998-6008).

Σχετικά Μαθήματα: Συστήματα Παράλληλης Επεξεργασίας, Νευρωνικά Δίκτυα και Ευφυή Υπολογιστικά Συστήματα

Επικοινωνία: Μαριάννα Τζώρτζη mtzortzi@cslab.ece.ntua.gr
Νικέλα Παπαδοπούλου, nikela@cslab.ece.ntua.gr, 210-772-2279
Γεώργιος Γκούμας, goumas@cslab.ece.ntua.gr, 210-772-2402

II. Αρχιτεκτονική

7 Αρχιτεκτονική Υπολογιστών και Μηχανική Μάθηση

Αλγόριθμοι μηχανικής μάθησης ταξινόμησης (classification) και πρόβλεψης (prediction) εφαρμόζονται πλέον κατά κανόνα σε τομείς όπως η όραση υπολογιστών, η επεξεργασία φυσικής γλώσσας κ.α, πετυχαίνοντας εντυπωσιακά αποτελέσματα. Και ενώ συχνά σχεδιάζεται εξειδικευμένο hardware για την επιτάχυνση τους, λίγες είναι προς το παρόν 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.

7.1 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). 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

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

Επικοινωνία: Χλόη Αλβέρτη, xalverti@cslab.ece.ntua.gr, 210-772-2279
Κωστής Νίκας, knikas@cslab.ece.ntua.gr, 210-772-4159

7.2 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. T

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

7.3 Απεικόνιση αλγορίθμων Βαθιάς Μάθησης στην πλατφόρμα Graphcore IPU

Τα IPUs της Graphcore (www.graphcore.ai) υπόσχονται επιδόσεις σημαντικά καλύτερες από GPUs στην επεξεργασία εφαρμογών βαθιάς μάθησης (που μπορούν να εκφραστούν ως γράφοι) με χρήση παραλληλισμού και πολλαπλών τοπικών μνημών. Τα IPUs της GraphCore προγραμματίζονται με το Poplar toolflow και υποστηρίζουν την απεικόνιση CNN (π.χ. με PyTorch). Ενδεικτικά, η εργασία αυτή θα ασχοληθεί με τα παρακάτω:

- Απεικόνιση ενός αλγορίθμου Histogram-of-Oriented-Gradients (HOG) με χρήση του προγραμματιστικού περιβάλλοντος Poplar και σύγκριση με multi-core CPUs και GPUs ως προς επιδόσεις, κλιμακωσιμότητα, κλπ.
- Αξιοποίηση πολλαπλών IPU για την επίλυση μεγαλύτερων προβλημάτων
- Αντικατάσταση του αλγορίθμου HOG με full-featured CNN και σύγκριση σε IPUs.

Σχετική Βιβλιογραφία:

1. Ενδεικτική εφαρμογή για απεικόνιση:`https://arxiv.org/abs/2006.00816`

2. Πληροφορίες για την πλατφόρμα:`https://www.graphcore.ai/hubfs/Lead%20gen%20assets/DSS8440%20IPU%20Server%20White%20Paper_2020.pdf`

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

8 Κύρια Μνήμη Συστήματος

8.1 Επίδοση και προσαρμογή εφαρμογών σε συστήματα με Non-Volatile Memories

Πρόσφατα έχουν κάνει την εμπορική εμφάνιση τους αναδυόμενες μη πτητικές τεχνολογίες μνήμης (NVM) (δηλ. PCM, STTRAM κ.λπ.) οι οποίες προσφέρουν byte-addressable πρόσβαση σε μόνιμα δεδομένα (διατηρημένα σε αστοχίες ισχύος), με καθυστέρηση πρόσβασης κοντά σε αυτή της τεχνολογίας DRAM. Είναι προσπελάσιμες με απλές load/store εντολές CPU και μπορούν να προσφέρουν χωρητικότητες της τάξης των TBs. Η μη πτητικότητα της τεχνολογίας μνήμης προσφέρει μια μοναδική

ευκαιρία για τον επαναπροσδιορισμό της αυστηρής διάκρισης μεταξύ μνήμης (memory) και αποθήκευσης (storage) στη στοίβα υπολογιστών. Στόχος της παρούσας διπλωματικής είναι η εξοικείωση με τα προγραμματιστικά περιβάλλοντα για NVM μνήμες, η ανάλυση επίδοσης εφαρμογών σε αυτά τα συστήματα, και η βελτιστοποίηση εκτέλεσης τους.

8.1.1 Βελτιστοποίηση εφαρμογών

Εφαρμογές έντασης δεδομένων (π.χ. βάσεις) μπορούν να ξαναγραφούν ώστε να αποφύγουν το serialization / deserialization των δεδομενων τους και να κερδίσουν σε επίδοση όταν τρέχουν σε συστήματα με PMem devices. Εκτός από την ριζική αναπροσαρμογή, εφαρμογές μπορούν να ενσωματώσουν μερικώς προγραμματιστικές πρακτικές που εκμεταλλεύονται την PMem τεχνολογία για να επιταχύνουν τα IO operations (e.g. χρήση mmio και μονιμοποίηση (persistence) των δεδομένων από το χώρο χρήστη). Στον αντίποδα, η PMem εισάγει και κόστη στο IO (κυρίως στο mmio) σε σύγκριση με την προσωρινή αποθήκευση IO δεδομένων στη DRAM (page cache buffering): έχει περιορισμένο bandwidth, αυξημένο latency και ασύμμετρα κόστη στα read/write operations. Άρα οι εφαρμογές πρέπει να προσέξουν ποια δεδομένα θα τοποθετήσουν αμειγώς στη PMem και για ποια δεδομένα θα επιτρέψουν το κλασικό DRAM buffering over storage. Σκοπός αυτής της διπλωματικής είναι η μελέτη των παρα πάνω και ο πειραματισμός με την προσαρμογή νέων εφαρμογών σε συστήματα με PMem devices.

Σχετικά Μαθήματα: Προηγμένα Θέματα Αρχιτεκτονικής Υπολογιστών, Εργαστήριο Λειτουργικών Συστημάτων

Σχετική Βιβλιογραφία:

1. Finding and Fixing Performance Pathologies in Persistent Memory Software Stacks
2. Persistent Memory Development Kit
3. Basic Performance Measurements of theIntel Optane DC Persistent Memory Module

Επικοινωνία: Χλόη Αλβέρτη, xalverti@cslab.ece.ntua.gr, 210-772-2279
Γεώργιος Γκούμας, goumas@cslab.ece.ntua.gr, 210-772-4133

8.1.2 Δομές δεδομένων

Στο καινούργιο προγραμματιστικό μοντέλο που οδηγεί η χρήση PMem μνήμης και στο οποίο δεσπόζει ο έλεγχος και η επιβολή της μονιμότητας των δεδομένων (data persistency) από το χώρο χρήστη, ένας θεμελιώδες εργαλείο είναι η μη πτητική στοίβα (non-volatile heap), η χρήση δλδ μιας persistent malloc. Τετοιες λειτουργίες/διεπαφές προσφέρονται από επίσημες βιβλιοθήκες (π.χ. Intel PMDK). και δίνουν τη δυνατότητα σχεδιασμού persistent δομών δεδομένων (π.χ. δέντρα, λίστες, γράφους). Οι δομές αποθηκεύονται μόνιμα σε ενα PMem storage device σε binary μορφή και προσπελαύνονται/ανανεώνονται κατευθείαν στο device). Η δυσκολία που προκύπτει ειναι η δημιουργία δομών που μπορούν να εγγυηθούν τη συνοχη (atomicity/consistency) των δεδομένων και της ιδιας της δομής μετά από ένα σφάλμα συστήματος ή μια αστοχία ισχύος (λειτουργίες που παραδοσιακά προσφέρει ένα σύστημα αρχείων στο χώρο πυρήνα). Στην παρούσα διπλωματική θα μελετήσουμε την αντίστοιχη βιβλιογραφία, θα εξοικειωθούμε με τα προγραμματιστικά εργαλεία και μοντέλα για PMem και θα επιχειρήσουμε το σχεδιασμό μιας persistent δομής (π.χ. skiplist).

Σχετική Βιβλιογραφία:

1. MOD: Minimally Ordered Durable Data structures for Persistent Memory
2. Data structure primitives on persistent memory: an evaluation

3. NV-Heaps: making persistent objects fast and safe with next-generation, non-volatile memories

4. Persistent Memory Development Kit

Επικοινωνία: Κωστής Νίκας, knikas@cslab.ece.ntua.gr, 210-772-2279
Χλόη Αλβέρτη, xalverti@cslab.ece.ntua.gr, 210-772-2279
Δημήτριος Σιακαβάρας, jimsiak@cslab.ece.ntua.gr, 210-772-2495

9 Μελέτη οργανώσεων πινάκων σελίδων (page tables)

Το λειτουργικό σύστημα είναι υπεύθυνο για την δέσμευση φυσικής μνήμης και για την αποθήκευση και συντήρηση των αντιστοιχίσεων των εικονικών διευθύνσεων των εφαρμογών σε φυσικές διευθύνσεις. Τη πληροφορία αυτή τη συντηρεί το λειτουργικό σε ειδικές δομές ανά εφαρμογή, τους πίνακες σελίδων (page tables). Οι δομές αυτές είναι παραδοσιακά δενδρικές (radix trees) τις οποίες προσπελαύνει ειδικός μηχανισμός υλικού (hardware page tables walkers) για να βρει τις φυσικές μεταφράσεις εικονικών διευθύνσεων των εφαρμογών κατά την εκτέλεση τους. Το βάθος του δέντρου εξαρτάταται από το μέγεθος του χώρου διευθύνσεων (address space) και ως σήμερα τα δέντρα ήταν τεσσάρων επιπέδων. Με την ολοένα αυξανόμενη χωρητικότητα των κύριων μνημών, το βάθος των δέντρων επίκειται να μεγαλώσει και να γίνει 5 επιπέδων. Το βάθος επηρρεάζει το χρόνο που απαιτείται για την ανάκτηση μιας μετάφρασης και συνεπώς την επίδοση των εφαρμογών. Η επίδρασή πολλαπλασιάζεται όταν οι εφαρμογές εκτελούνται εντός εικονικών μηχανών (virtual machines) όπου χρειάζεται η εμφολευμένη προσπέλαση (nested paging) των πινάκων σελίδων τόσο του guest οσο και του host machine. Σύγχρονες ερευνητικές δουλειές προτείνουν εναλλακτικές οργανώσεις των πινάκων σελίδων, π.χ πίνακες κατακερματισμού αντί για δέντρα, για τη μείωση του χρόνο του page table walk. Σκοπός της παρούσας διπλωματικής είναι να προσομοιώσει και να αξιολογήσει διαφορετικές οργανώσεις page tables.

Σχετικά Μαθήματα: Προηγμένα Θέματα Αρχιτεκτονικής Υπολογιστών, Λειτουργικά Συστήματα

Σχετική Βιβλιογραφία:

1. Hash don't cache the page table

2. Elastic Cuckoo Page Tables: Rethinking Virtual Memory Translation for Parallelism

Επικοινωνία: Χλόη Αλβέρτη, xalverti@cslab.ece.ntua.gr, 210-772-2279

10 Επεκτάσεις Αρχιτεκτονικής

10.1 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

11 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).

11.1 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

11.2 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.

Τα θετικά οφέλη των TEE αντισταθμίζονται από την πιθανή μείωση της επίδοσης των εφαρμογών λόγω των παρενεργειών της πρόσθετης ασφάλειας. Το υψηλότερο κόστος των context-switches λόγω των επικείμενων TLB και Cache flushes, καθώς και το επιπλέον υπολογιστικό κόστος για integrity-confidentiality δημιουργούν την ανάγκη για εξέταση εναλλακτικών τεχνικών που να προσφέρουν ικανοποιητικά επίπεδα ασφάλειας, χωρίς την μείωση της επίδοσης. Στην συγκεκριμένη διπλωματική θα ασχοληθούμε με το Keystone TEE της αρχικτεκτονικής RISC-V, την ανάλυση της συμπεριφοράς του και τις πιθανές τεχνικές/μεθόδους βελτίωσης της επίδοσης των εφαρμογών που τρέχουν εντός ενός Keystone enclave. Οι πιθανές προεκτάσεις της παρούσας διπλωματικής μπορεί να αφορούν το Keystone Framework, το λειτουργικό σύστημα Linux ή και την υποκείμενη αρχιτεκτονική RISC-V.

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] 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

11.3 Σχεδιασμός, υλοποίηση, και αξιολόγηση επίδοσης προηγμένων σχημάτων κρυφής μνήμης του συστήματος διαχείρισης εικονικής μνήμης του Rocket Chip Generator

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.

Στα θέματα των υποενοτήτων που ακολουθούν θα ασχοληθούμε με το σύστημα διαχείρισης εικονικής μνήμης του Rocket Chip Generator. Το σύστημα διαχείρισης εικονικής μνήμης (Memory Management Unit - MMU) παίζει διττό ρόλο στα σύγχρονα υπολογιστικά συστήματα, (i) διασφαλίζει την μνήμη του συστήματος μέσω της απομόνωσης των διεργασιών και (ii) ενισχύει την παραγωγικότητα του προγραμματιστή. Τα παραπάνω επιτυγχάνονται με την χρήση εικονικών διευθύνσεων πρόσβασης στην μνήμη αντί για φυσικών, και με τον διαχωρισμό της μνήμης σε σελίδες (συνήθως των 4KB). Με την χρήση εικονικών διευθύνσεων κάθε εφαρμογή "νομίζει" ότι δουλεύει πάνω σε συνεχόμενες σελίδες μνήμης, ενώ στην πραγματικότητα οι σελίδες μπορεί να είναι διάσπαρτες στην φυσική μνήμη. Το σύστημα διαχείρισης εικονικής μνήμης γνωστών αρχιτεκτονικών (x86, ARM, RISC-V, κλπ) αποτελείται από την μονάδα του Page Table Walker ο οποίος είναι υπεύθυνος για την μετάφραση των διευθύνσεων από εικονικές σε φυσικές, καθώς και από τον Translation Lookaside Buffer (TLB), μία κρυφή μνήμη, στην οποία κρατούνται οι πρόσφατες εικονικές-σε-φυσικές μεταφράσεις διευθύνσεων. 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] The Chisel Language,
`https://www.chisel-lang.org`

11.3.1 Advanced MMU Caching Techniques for the Rocket Chip Generator

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. Η παρούσα διπλωματική στοχεύει σε συνέχεια προηγούμενης διπλωματικής εργασίας στην υλοποίηση κάποιου -ή κάποιων- από τα παραπάνω σχήματα στον Rocket Chip Generator (RCG), και να εξεταστεί η επίδοση τους έναντι του vanilla TLB του RCG.

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`

11.3.2 Enabling TLB Prefetching for the Rocket Chip Generator

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] του Rocket Chip Generator, 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`

11.3.3 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

11.4 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

12 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».

12.1 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

12.2 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

12.3 Υλοποίηση context-switch και preemption μηχανισμού για υπολογιστικούς πυρήνες σε FPGAs.

Στη σύγχρονη εποχή οι FPGAs ενσωματώνονται σε συστήματα cloud, με πιο χαρακτηριστικά παραδείγματα το F1 instance της Amazon και της Alibaba. Επιπλέον, οι χρήστες έχουν τη δυνατότητα να αξιοποιήσουν έτοιμες υλοποιημένες συναρτήσεις από διαθέσιμες βιβλιοθήκες των παρόχων στα προγράμματα τους, που έχουν σχεδιαστεί και ενσωματωθεί σε FPGAs (Software as a Service). Κάθε υπολογιστικός πυρήρας όμως μπορεί να εξυπηρετήσει μόνο έναν χρήστη (task-based), ενώ οι υπόλοιποι χρήστες πρέπει να περιμένουν στην ουρά προκειμένου να γίνει διαθέσιμος εκ νέου ο υπολογιστικός πυρήνας. Η κατάσταση αυτή οδηγεί σε μια σειρά προβλημάτων, με πιο κύρια την μη-δίκαιη κατανομή του πυρήνα μεταξύ των χρηστών. Μια λύση του παραπάνω προβλημάτος αποτελεί η δημιουργία ενός preemption και context-switch μηχανισμού, ο οποίος α) θα διακόπτει μια διεργασία, β) θα αποθηκεύει την κατάσταση της και στη συνέχεια γ) θα φορτώνει την επόμενη διεργασία της ουράς, όπως γίνεται στα λειτουγρικά συστήματα. Η ενσωμάτωση του παραπάνου μηχανισμού σε μια σχεδίαση ενός υπολογιστικού πυρήνα, επιτρέπει τη κοινή χρήση του επιταχυντή με σκοπό τη δίκαιη κατανομή του από διαφορετικούς χρήστες (ή διεργασίες). Σκοπός της διπλωματικής εργασίας είναι η ανάπτυξη ενός μηχανισμού που θα επιτρέπει τη κοινή χρήση ενός επιταχυντή, ο οποίος έχει σχεδιαστεί και υλοποιηθεί για FPGAs, από διαφορετικούς χρήστες (ή/και διεργασίες) στον άξονα του χρόνου, μέσω της υλοποίησης context-switch και preemption μηχανισμού.

Σχετικά Μαθήματα: Προηγμένα Θέματα Αρχιτεκτονικής Υπολογιστών

Σχετική Βιβλιογραφία:

1. Jiacheng Ma, Gefei Zuo, Kevin Loughlin, Xiaohe Cheng, Yanqiang Liu, Abel Mulugeta Eneyew, Zhengwei Qi, and Baris Kasikci. 2020. A Hypervisor for Shared-Memory FPGA Platforms. 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, 827–844. DOI:https://doi.org/10.1145/3373376.3378482

2. Ahmed Khawaja, Joshua Landgraf, Rohith Prakash, Michael Wei, Eric Schkufza, and Christopher J. Rossbach. 2018. Sharing, protection, and compatibility for reconfigurable fabric with Amorphos. In Proceedings of the 13th USENIX conference on Operating Systems Design and Implementation (OSDI'18). USENIX Association, USA, 107–127.

3. Joshua Landgraf, Tiffany Yang, Will Lin, Christopher J. Rossbach, and Eric Schkufza. 2021. Compiler-driven FPGA virtualization with SYNERGY. In Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2021). Association for Computing Machinery, New York, NY, USA, 818–831. DOI:https://doi.org/10.1145/3445814.3446755

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

III. Λειτουργικά Συστήματα - Εικονικές Μηχανές

13 Persistent Memory (PMem) – Μη πτητική μνήμη

Πρόσφατα έγιναν εμπορικά διαθέσιμες μη πτητικές τεχνολογίες μνήμης (NVM) (e.g. Intel Optane NVDIMM) οι οποίες έχουν καλύτερη κλιμάκωση πυκνότητας με χαμηλότερο κόστος σε σχέση με τη τεχνολογία DRAM. Οι NVM μνήμες προσφέρουν byte-addressable πρόσβαση σε μόνιμα δεδομένα (διατηρημένα σε αστοχίες ισχύος) μέσω εντολών μνήμης CPU (loads/stores). Ο χρόνος απόκρισης (latency) και το εύρος ζώνης (bandwidth) που προσφέρουν είναι "κοντά" σε αυτα της τεχνολογίας DRAM, αλλά παραμένουν σημαντικά χειρότερα παρουσιάζοντας ταυτόχρονα ιδιαίτερα χαρακτηριστικά (π.χ. ανομοιομορφία read/write access). Η μη πτητικότητα, η επίδοση και η πυκνότητα τετοιων μνημών (μπορούν να προσφέρουν χωρητικότητες της τάξης των TBs) δίνουν τη μοναδική ευκαιρία για χρήση τους είτε ως μέσου γρήγορης αποθήκευσης (storage), είτε ως ενός έξτρα επιπέδου πιο αργής μνήμης (slow memory). Η δυνατότητα άμεσης προσπέλασης από τη CPU (direct access via load/store) και στις δύο περιπτώσεις επιτρέπει ακόμα και τον ρηξικέλευθο επαναπροσδιορισμό της αυστηρής διάκρισης μεταξύ μνήμης και αποθήκευσης στη στοίβα υπολογιστών.

13.1 Persistent memory as storage

Δεδομένου ότι ο χρόνος απόκρισης της PMem είναι τάξεις μεγεθους μικρότερος από κλασικά storage devices (e.g. flash ssds or hdds), το κόστος του λογισμικού συτήματος εμφανίζεται συχνά ως το νέο σημείο συμφόρησης της επίδοσης των εφαρμογών έντασης IO. Γι'αυτό γίνεται προσπάθεια τόσο από την ακαδημαική κοινότητα όσο και από τη βιομηχανία να βελτιστοποιηθεί το ΛΣ για PMem συσκευές (π.χ. αναπτύσσονται ειδικά συστήματα αρχείων). Η PMem είναι συνδεδεμένη στο memory bus, και η προσπέλαση της γίνεται με CPU εντολες load και store. Ετσι, το πιο δημοφιλές interface για να προσπελάσει κανείς αρχεία αποθηκευμένα σε PMem ειναι με τη κλήση συστήματος mmap() (memory-mapped IO). Με τον τρόπο αυτό η εκάστοτε εφαρμογή αποκτά πρόσβαση σε μόνιμα δεδομένα χρησιμοποιώντας απλά ένα εύρος εικονικών διευθύνσεων (direct access –DAX), το πιο σύντομο μονοπάτι πρόσβασης σε χώρο αποθήκευσης. Οι παρακάτω δύο διπλωματικές θα μελετήσουν και θα αναζητήσουν λύσεις για δύο σημαντικές πηγές κόστους επίδοσης κατά το memory-mapped IO σε PMem.

13.1.1 Σελιδοποίηση και κόστος μετάφρασης σε συστήματα με PMem storage. Μελέτη επίδοσης και υποστήριξης μεγάλων σελίδων (2MB και 1GB).

Στο memory-mapped IO ένα από τα βασικά κόστη του λογισμικού συστήματος προκύπτει από τα σφάλματα σελίδας (page faults). Το κόστος αυτό μειώνεται δραστικά με τη χρήση μεγάλων σελίδων, οι οποίες εν δυνάμει επιταχύνουν και την εκτέλεση μιας εφαρμογής μειώνοντας των αριθμό των TLB αστοχιών μετάφρασης σε επίπεδο μικροαρχιτεκοτνικής. Ωστόσο τα συστήματα αρχείων και συγκεκριμένα οι κατανεμητές τους (allocators) δεν έχουν δομηθεί/βελτιστοποιηθεί για τη χρήση μεγάλων σελίδων. Επίσης, τα συστήματα αρχείων υποφέρουν από εξωτερικό κατακερματισμό (με διαφορετικά και μόνιμα χαρακτηριστικά σε σχέση με τον κατακερματισμό μνήμης) ο οποίος δυσχεραίνει τη χρήση μεγάλων σελίδων. Στη παρούσα διπλωματική θα μελετήσουμε τη χρήσης μεγάλων σελίδων στην επίδοση εφαρμογών έντασης IO και την επίπτωση του κατακερματισμού των συστημάτων αρχείων. Στη συνέχεια θα επιχειρήσουμε τρόπους αντιμετώπισης του κατακερματισμού είτε με παρεμβάσεις στον κατανεμητή είτε με εργαλεία ανασυγκρότησης στο ερευνητικό σύστημα αρχείων NOVA.

Σχετικά Μαθήματα: Εργαστήριο Λειουργικών Συστημάτων

Σχετική Βιβλιογραφία:

1. WineFS: a hugepage-aware file system for persistent memory that ages gracefully

Επικοινωνία: Χλόη Αλβέρτη, xalverti@cslab.ece.ntua.gr, 210-772-2279

13.2 Persistent memory as a memory tier

Οι μνήμες είναι ενα από τα πιο ακριβά κομμάτια των υπολογιστικών συστημάτων, και επηρρεάζουν σημαντικά την ταχύτητα εκτέλεσης σημαντικών εφαρμογών που κυριαρχούν στο υπολογιστικό νέφος (πχ. key-value stores, in-memory databases, graph analytics). Αλλά καθώς τα σύνολα δεδομένων των εφαρμογών συνεχίζουν να αυξάνονται, οι απαιτήσεις σε χωρητικότητα μνήμης αυξάνονται ακόμα περισσότερο. Ταυτόχρονα, η τεχνολογία της παραδοσιακής κύριας μνήμης (DRAM) έχει φτάσει σε ένα όριο κλιμάκωσης που περιορίζει την πυκνότητά της. Η κλιμακωσιμότητα της PMem μπορεί να δωσει λύση σε αυτό το πρόβλημα, αλλά η επίδοση της (latency και bandiwth) ειναι σημαντικά υποδεέστερη σε σχέση με τη DRAM. Έτσι εχει προταθεί η χρήση ιεαραρχιών μνημών, συνδυάζοντας αργές (PMem) και γρήγορες (DRAM) μνήμες. Σε τέτοια διατάξεις η μη πτητικότητα της PMem αγνοείται κα ιχρησιμοποιείται σαν μια πιο αργή και λιγότερο ενεργειακά κοστοβόρα μνήμη.

13.2.1 Συστήματα με υβριδική ιερραρχία μνήμης (DRAM + PMem). Μελέτη τεχνικών τοποθέτησης και μεταφοράς δεδομένων μεταξύ των διαφορετικών επιπέδων μνήμης.

Σε μια υβριδική ιεραρχία αργών και γρήγορων μνημών, η τοποθέτηση των δεδομένων στα διαφορετικά επίπεδα παίζει καταλυτικό ρόλο στην τελική επίδοση. Η τοποθέτηση μπορεί να γίνει αμειγώς από το υλικό (Intel Optane memory mode), αλλά ερευνητικές εργασίες δείχνουν οτι μια τετοια προσέγγιση παρότι διαφανής ειναι μάλλον μονολιθική, χάνοντας πολλές ευκαιρίες βελτιστοποίησης. Στη παρούσα διπλωματική θα μελετήσουμε τεχνικές τοποθέτησης στα διάφορα επίπεδα μνήμης από το λογισμικό στηριζόμενοι στη σχετική βιβλιογραφία. Θα αξιολογήσουμε τη πιθανή μεταφορά τους στο χώρο του πυρήνα και θα συμπεριλάβουμε και την κατανάλωση ενέργειας ως κριτήριο τοποθέτησης (εκτός από την επίδοση).

Σχετικά Μαθήματα: Εργαστήριο Λειουργικών Συστημάτων, Αρχιτεκτονική Υπολογιστών, Συστήματα Παράλληλης επεξεργασίας

Σχετική Βιβλιογραφία:

1. HeMem: Scalable Tiered Memory Management for Big Data Applications and Real NVM
2. An Empirical Guide to the Behavior and Use of Scalable Persistent Memory

Επικοινωνία: Χλόη Αλβέρτη, xalverti@cslab.ece.ntua.gr, 210-772-2279

14 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: Χλόη Αλβέρτη, xalverti@cslab.ece.ntua.gr, 210-772-2279

15 Μελέτη της αλληλεπίδρασης των μηχανισμών δέσμευσης (εικονικής) μνήμης χώρου χρήστη (userspace (virtual) memory allocators) με τους μηχανισμού δέσμευσης (φυσικής) μνήμης χώρου πυρήνα (kernelspace (physical) memory allocators)

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

16 Μελέτη, αξιολόγηση και επέκταση των μηχανισμών δέσμευσης και απόδοσης μνήμης για προγράμματα που συνδυάζουν υπολογισμούς υψηλής απόδοσης (HPC), μηχανική μάθηση (ML) και ανάλυση δεδομένων μεγάλης κλίμακας (Data Science και BigData).

Υπάρχει μια κατηγορία προβλημάτων η επίλυση των οποίων απαιτεί την χρησιμοποίηση διαφορετικών μοντέλων (paradigms) προγραμματισμού, π.χ. κώδικας που χρησιμοποιείται για επιστημονικές προσομοιώσεις συνήθως χρησιμοποιεί γλώσσες και βιβλιοθήκες από το πεδίο του High Performance Computing (HPC), όπως το MPI [3] για την παραλληλοποίηση του κώδικα. Ενδέχεται όμως να απαιτείται η χρήση μηχανικής μάθησης (ML) για να γίνει drive το simulation, π.χ. με κάποιο νευρωνικό δίκτυο γραμμένο σε Tensorflow [5], ή η ανάλυση των αποτελεσμάτων σε μεγάλη κλίμακα, χρησιμοποιώντας π.χ. το Spark [6]. Αυτά τα βήματα, που συνήθως γράφονται σε διαφορετικές γλώσσες και frameworks, τρέχουν συχνά και σε διαφορετική υποδομή από τον κώδικα της προσομοίωσης.

Για να μειωθεί το fragmentation σε επίπεδο προγραμματισμού αλλά και υποδομών και να διευκολυνθεί η ενοποιημένη ανάπτυξη κώδικα που θα αναλαμβάνει όλες τις φάσεις εκτέλεσης του simulation, έχει δημιουργεί το Daphne [1], μια γλώσσα προγραμματισμού ειδικού σκοπού (Domain Specific Language), η οποία δίνει τη δυνατότητα για ανάπτυξη κώδικα που θα μπορεί να ενσωματώνει χαρακτηριστικά και δυνατότητες και από τα τρία προαναφερθέντα paradigms. Το Daphne χρησιμοποιεί εσωτερικά το MLIR [2], ένα compiler framework βασισμένο στο LLVM [4], για την υλοποίηση της γλώσσας, και ενσωματώνει ένα runtime σε C++, το οποίο υλοποιεί τη λειτουργικότητα των operations που δίνει το Daphne στον τελικό χρήστη.

Σκοπός της διπλωματικής είναι να διερευνηθεί η επίδραση της δέσμευσης μνήμης (virtually KAI physically) στην επίδοση του Daphne runtime και τον τελικών προγραμμάτων. Συγκεκριμένα, πρόκειται να αξιολογηθεί η επιδραση της ανομοιόμορφης πρόσβασης στη μνήμη (NUMA) [7] και θα διερευνηθεί η επέκταση του Daphne runtime ώστε να υποστηρίζει NUMA-aware allocations, είτε άμεσα είτε με τη χρήση κάποιου NUMA-aware C / C++ allocator [10]. Επίσης, θα αξιολογηθεί η επίδραση της χρήσης μεγάλων σελίδων (transparent hugepages [8], hugetlbfs [9]), σε αρχιτεκτονικές x86 και ARMv8, στην επιδοση του runtime και εάν είναι αναγκαίο να επεκταθεί το memory allocation του Daphne runtime ώστε να υποστηρίζει hugepage-aware allocations.

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

Σχετική Βιβλιογραφία:

1. `https://github.com/daphne-eu/daphne`
2. `https://mlir.llvm.org/`
3. `https://en.wikipedia.org/wiki/Message_Passing_Interface`
4. `https://en.wikipedia.org/wiki/LLVM`
5. `https://en.wikipedia.org/wiki/TensorFlow`
6. `https://spark.apache.org/`
7. `https://www.kernel.org/doc/Documentation/vm/numa.rst`
8. `https://www.kernel.org/doc/Documentation/mm/transhuge.rst`
9. `https://www.kernel.org/doc/Documentation/vm/hugetlbpage.txt`
10. `https://en.wikipedia.org/wiki/C_dynamic_memory_allocation#Implementations`

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

IV. Κατανεμημένα Συστήματα - Προχωρημένα θέματα βάσεων δεδομένων

17 Αυτόματη επιλογή και ταξινόμηση δεδομένων εισόδου βάσει χρησιμότητας για αναλυτικές εργασίες μεγάλου όγκου δεδομένων

Ενώ η βελτιστοποίηση εργασιών στον τομέα των Big Data συνήθως υλοποιείται με την αύξηση της παραλληλοποίησης και τη χρήση πλατφορμών κατανεμημένης επεξεργασίας, λίγα έχουν γίνει σχετικά με την επιλογή των πιο κατάλληλων δεδομένων από μια μεγάλη συλλογή διαθέσιμων. Πολλές αναλυτικές εργασίες είναι ιδιαίτερα ευαίσθητες στο περιεχόμενο των δεδομένων και όχι τόσο στο μέγεθός τους (π.χ., content based advertising, social network analytics). Στις εργασίες [1, 2] παρουσιάσαμε μια γενική μεθοδολογία για γρήγορη σύγκριση και ταξινόμηση πολλαπλών διαθέσιμων δεδομένων εισόδου (datasets) με βάση το αποτέλεσμα που προκαλούν όταν εφαρμόζονται σε τελεστές αναλυτικής επεξεργασίας. Στη συγκεκριμένη διπλωματική, καλείστε να υλοποιήσετε και να βελτιστοποιήσετε την επέκταση του συστήματος σε μια από τις ακόλουθες κατευθύνσεις:

- Σε δεδομένα κειμένου. Συγκεκριμένα, για τελεστές που παίρνουν σαν είσοδο 1 αρχείο κειμένου το σύστημα πρέπει να μοντελοποιεί τις διάφορες ανάμεσα σε πολλά κείμενα καθώς και να προβλέπει την έξοδο του τελεστή για οποιοδήποτε κείμενο με το ελάχιστο σφάλμα.
- Σε τελεστές που δέχονται περισσότερες της μιας εισόδους (π.χ. Join operator). Το σύστημα θα πρέπει να τροποποιηθεί ώστε να μοντελοποιεί την επίδραση που έχουν 2 dataset εισόδου καθώς και οι ομοιότητές τους στην πρόβλεψη του αποτελέσματος του τελεστή.

Επιθυμητή και είναι η υποβολή δημοσίευσης από την παραπάνω εργασία σε σχετικό workshop.

Σχετικά Μαθήματα: Προχωρημένα θέματα βάσεων δεδομένων, Κατανεμημένα Συστήματα

Σχετική Βιβλιογραφία:

1. T. Bakogiannis, I. Giannakopoulos, D. Tsoumakos and N. Koziris: Apollo: A Dataset Profiling and Operator Modeling System. In Proceedings of the 2019 ACM SIGMOD/PODS.
2. I. Giannakopoulos, D. Tsoumakos and N. Koziris: A Content-Based Approach for Modeling Analytics Operators. In Proceedings of the 27th ACM International Conference on Information and Knowledge Management.

Επικοινωνία: Δημήτριος Τσουμάκος, dtsouma@cslab.ece.ntua.gr

18 Αποδοτικός συνεχής υπολογισμός ερωτημάτων σε δυναμικά δεδομένα γράφων με τη χρήση του Timely Dataflow

Η ανάλυση σε μεγάλα δεδομένα γράφων τόσο σε στατικό όσο και δυναμικό (δλδ, ο γράφος διαρκώς μεταβάλλεται με εισαγωγές & διαγραφές κόμβων και ακμών) επίπεδο έχει εξελιχθεί σε πολύ σημαντικό πεδίο έρευνας και ανάπτυξης. Ένα από τα πιο σύγχρονα εργαλεία που επιτρέπουν τέτοιους υπολογισμούς είναι το Naiad [1], που υλοποιεί το Timely-Dataflow υπολογιστικό μοντέλο.

Το μοντέλο υποστηρίζει επαναληπτικούς και συνεχείς υπολογισμούς. Δίνει τη δυνατότητα επεξεργασίας ροών και εργασιών δέσμης με ταχύτητα, χρησιμοποιώντας μια νέα προσέγγιση συντονισμού που συνδυάζει ασύγχρονη και σύγχρονη εκτέλεση. Το Differential dataflow [2] εκτελεί επαναληπτικό υπολογισμό σε ροές δεδομένων με τον υπολογισμό να υφίσταται μόνο σε απόκριση προς την αλλαγή των δεδομένων. Στη συγκεκριμένη διπλωματική, καλείστε να χρησιμοποιήσετε την open-source έκδοση σε Rust (`https://github.com/frankmcsherry/timely-dataflow` & `https://github.com/frankmcsherry/differential-dataflow`) και, αφού υλοποιήσετε γνωστούς αλγορίθμους γράφων (π.χ. Triangle counting, pagerank, centralities) να συγκρίνετε τη streaming εκδοχή τους στο Timely-Dataflow με το GraphX (Spark) [3].

Σχετικά Μαθήματα: Προχωρημένα θέματα βάσεων δεδομένων, Κατανεμημένα Συστήματα

Σχετική Βιβλιογραφία:

1. Derek G. Murray, Frank McSherry, Rebecca Isaacs, Michael Isard, Paul Barham, and Martín Abadi. Naiad: A timely dataflow system. In SOSP, pages 439–455, 2013.

2. Frank McSherry, Derek Gordon Murray, Rebecca Isaacs, and Michael Isard. 2013. Differential Dataflow. In Proc. Conf. on Innovative Data Systems Research (CIDR).

3. Joseph E. Gonzalez, Reynold S. Xin, Ankur Dave, Daniel Crankshaw, Michael J. Franklin, Ion Stoica. GraphX: Graph Processing in a Distributed Dataflow Framework. In USENIX Symposium on Operating Systems Design and Implementation (OSDI 14).

Επικοινωνία: Δημήτριος Τσουμάκος, dtsouma@cslab.ece.ntua.gr

19 Μελέτη και σύγκριση πολυ-συστημάτων (polystores) εκτέλεσης αναλυτικών SQL ερωτημάτων

Το Presto [1,2] είναι μια κατανεμημένη μηχανή εκτέλεσης SQL ερωτημάτων ανοιχτού κώδικα για τη διεξαγωγή διαδραστικών αναλυτικών ερωτημάτων σε πηγές δεδομένων όλων των μεγεθών που κυμαίνονται από gigabytes έως petabytes. Το Presto επιτρέπει την αναζήτηση δεδομένων σε πολλαπλές βάσεις όπως Hive, Cassandra, MySQL, MongoDB, ElasticSearch, κλπ.. Ένα ερώτημα Presto μπορεί να συνδυάσει δεδομένα από πολλαπλές πηγές, ανήκοντας στην κατηγορία των "polystores". Στη συγκεκριμένη διπλωματική, καλείστε να μελετήσετε τις δυνατότητες του Presto και να το συγκρίνετε με τα "συγγενικά" MuSQLE [3], Impala[4] και SparkSQL [5] σχετικά με:

- βελτιστοποίηση ερωτημάτων,
- κλιμακωσιμότητα,
- απόδοση.

Σχετικά Μαθήματα: Προχωρημένα θέματα βάσεων δεδομένων, Κατανεμημένα Συστήματα

Σχετική Βιβλιογραφία:

1. R. Sethi et al., "Presto: SQL on Everything," 2019 IEEE 35th International Conference on Data Engineering (ICDE).

2. `https://github.com/prestosql/presto`

3. V. Giannakouris, N. Papailiou, D. Tsoumakos and N. Koziris: MuSQLE: Distributed SQL Query Execution Over Multiple Engine Environments. In Proceedings of the 2016 IEEE International Conference on Big Data (BigData 2016).

4. `https://impala.apache.org/`

5. `https://spark.apache.org/sql/`

Επικοινωνία: Δημήτριος Τσουμάκος, dtsouma@cslab.ece.ntua.gr

20 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

21 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

22 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

23 Κατανεμημένη Βαθιά Μηχανική Μάθηση

Ο συνεχώς αυξανόμενος όγκος των δεδομένων της σημερινής εποχής, είτε πρόκειται για δομημένα δεδομένα (π.χ. csv, txt) είτε για δεδομένα χωρίς δομή (εικόνα, ήχος, κείμενο) καθιστά πολλές φορές την εκπαίδευση με κλασσικούς αλγορίθμους μηχανικής μάθησης χαμηλότερου επιπέδου, αφού οι τελευταίοι παρουσιάζουν έναν κορεσμό αναφορικά με την ακρίβεια που μπορούν να πετύχουν όσο αυξάνεται η πολυπλοκότητα των δεδομένων. Για καλύτερη αναπαράσταση του συνόλου των δεδομένων συνηθίζεται πλέον να γίνεται εκπαίδευση με χρήση νευρωνικών δικτύων. Ωστόσο, τόσο ο αυξανόμενος όγκος των δεδομένων όσο και η πολυπλοκότητα των δικτύων καθιστούν συνήθως απαγορευτική την

εκπαίδευση νευρωνικών δικτύων σε έναν υπολογιστικό κόμβο αναφορικά με το χρόνο εκπαίδευσης. Για το σκοπό αυτό, έχουν προταθεί δύο βασικοί τρόποι για τη βαθιά μηχανική μάθηση με κατανεμημένο τρόπο: η αρχιτεκτονική εξυπηρετήτη παραμέτρων (parameter server) και η κατανεμημένη εκπαίδευση με χρήση τεχνικών ολικής μείωσης (all-reduce).

23.1 Συγκρητική αξιολόγηση τεχνικών συγχρονισμού για την κατανεμημένη εκπαίδευση νευρωνικών δικτύων στην αρχιτεκτονική του εξυπηρετητή παραμέτρων.

Η αρχιτεκτονική εξυπηρετητή παραμέτρων είναι μία κεντρικού τύπου αρχιτεκτονική στην οποία συμμετέχουν δύο είδη διεργασιών: ο εξυπηρετητής παραμέτρων, που αποθηκεύει και διατηρεί το καθολικό μοντέλο, και οι εργάτες, οι οποίοι σε κάθε βήμα υπολογίζουν τα διανύσματα κλίσης της εκπαίδευσης. Η συγκεκριμένη αρχιτεκτονική μπορεί να λειτουργήσει τόσο με σύγχρονο τρόπο, όπου σε κάθε βήμα επανάληψης της εκπαίδευσης συγχρονίζουν όλοι οι εργάτες τα διανύσματα κλίσης που υπολόγισαν στον εξυπηρετητή, όσο και με ασύγχρονο τρόπο, όπου τα διανύσματα κλίσης που υπολογίζει ο καθένας αξιοποιούνται ανεξάρτητα. Η σύγχρονη εκτέλεση παρουσιάζει έντονη καθυστέρηση στο χρόνο εκτέλεσης, ενώ η ασύγχρονη, παρότι πιο γρήγορη, μπορεί να οδήγησει σε ένα τελικό μοντέλο που να υστερεί στην ακρίβεια. Παράλληλα, εκτός από τις δύο αυτές προσεγγίσεις έχουν προταθεί και άλλες ενδιάμεσες τεχνικές συγχρονισμού. Στόχος αυτής της διπλωματικής εργασίας είναι να γίνει μία πειραματική αποτίμηση των μεθόδων συγχρονισμού που έχουν προταθεί στην συγκεκριμένη αρχιτεκτονική.

Σχετική Βιβλιογραφία:

1. Li, Mu, et al. "Communication efficient distributed machine learning with the parameter server." Advances in Neural Information Processing Systems 27 (2014).

2. Li, Mu, et al. "Parameter server for distributed machine learning." Big learning NIPS workshop. Vol. 6. No. 2. 2013.

3. Ho, Qirong, et al. "More effective distributed ml via a stale synchronous parallel parameter server." Advances in neural information processing systems 26 (2013).

4. Gupta, Suyog, Wei Zhang, and Fei Wang. "Model accuracy and runtime tradeoff in distributed deep learning: A systematic study." 2016 IEEE 16th International Conference on Data Mining (ICDM). IEEE, 2016.

5. Dutta, Sanghamitra, et al. "Slow and stale gradients can win the race: Error-runtime trade-offs in distributed SGD." International conference on artificial intelligence and statistics. PMLR, 2018.

6. Zhao, Xing, et al. "Dynamic stale synchronous parallel distributed training for deep learning." 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS). IEEE, 2019.

23.2 Προσεγγιστικές τεχνικές ολικής μείωσης στην κατανεμημένη εκπαίδευση νευρωνικών δικτύων.

Η κατανεμημένη εκπαίδευση με χρήση τεχνικών ολικής μείωσης είναι μία αποκεντρωποιημένη διαδικασία εκπαίδευσης, εξ' ορισμού σύγχρονη, όπου σε κάθε βήμα της εκπαίδευσης, οι εργάτες που

συμμετέχουν συγχρονίζουν τα διανύσματα κλίσης που υπολόγισαν με χρήση τεχνικών ολικής μείωσης (all-reduce). Τα τελευταία χρόνια, λαμβάνοντας υπόψιν και την ετερογένεια που παρατηρείται πολλές φορές σε συστοιχίες υπολογιστών σε περιβάλλοντα υπολογιστικού νέφους, έχουν προταθεί ασθενέστερες, απο μεριάς επιπέδου συγχρονισμού προσεγγίσεις εφαρμογής της ολικής μείωσης, όπως ο αλγόριθμος AP-SGD και μία ασύγχρονη εκδοχή, όπου εφαρμόζεται η ολική μείωση ανά ομάδες. Στόχος της συγκεκριμένης διπλωματικής είναι η αξιολόγηση αυτών των μεθόδων, καθώς και η υλοποίηση και σύγκριση εναλλακτικής μεθόδου ασθενούς συγχρονισμού.

Σχετική Βιβλιογραφία:

1. Lian, Xiangru, et al. "Asynchronous decentralized parallel stochastic gradient descent." International Conference on Machine Learning. PMLR, 2018.

2. Luo, Qinyi, et al. "Prague: High-performance heterogeneity-aware asynchronous decentralized training." Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems. 2020.

Σχετικά Μαθήματα: Κατανεμημένα Συστήματα
Επικοινωνία: Νίκος Προβατάς, nprov@cslab.ece.ntua.gr

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

2. 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.

3. 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.

4. 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

25 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

26 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

27 Αποτίμηση λειτουργίας συστημάτων επεξεργασίας μεγάλου όγκου δεδομένων πάνω σε skewed σύνολα δεδομένων.

Η ανάλυση μεγάλου όγκου δεδομένων (Big Data) με κατανεμημένα συστήματα επεξεργασίας είναι ένας από τους πιο δημοφιλείς τομείς ανάπτυξης τα τελευταία χρόνια. Τα περισσότερα κατανεμημένα συστήματα επεξεργασίας όμως, δε μπορούν να επεξεργαστούν αποδοτικά δεδομένα του παρουσιάζουν ανομοιομορφίες (skew). Συγκεκριμένα, η ανάλυση δεδομένων με χρήση συνενώσεων (joins) και συνυπολογισμών (aggregations) επηρεάζεται σημαντικά από το skew των δεδομένων, και τα εν λόγω συστήματα επεξεργασίας παρουσιάζουν συνήθως χαμηλή απόδοση σε τέτοιου τύπου ανάλυση πάνω σε skewed δεδομένα. Παρόλο που έχουν γίνει ερευνητικές προσπάθειες αντιμετώπισης του φαινομένου του data skew σε αλγοριθμικό επίπεδο και πάνω σε ειδικά συστήματα [1,2,3], τα περισσότερα από τα υπάρχοντα κατανεμημένα συστήματα επεξεργασίας δεν έχουν υλοποιήσει σχετικούς μηχανισμούς για την αντιμετώπισή του. Το Apache Spark είναι ένα από τα πιο δημοφιλή συστήματα επεξεργασίας μεγάλου όγκου δεδομένων, και παρέχει το SparkSQL[4] module για την ανάλυση δεδομένων με χρήση SQL ερωτημάτων. Σε μια προσπάθεια αντιμετώπισης του προβλήματος αυτού, ανέπτυξε πρόσφατα το adaptive execution framework [5]. Στη συγκεκριμένη διπλωματική, καλείστε να μελετήσετε το adaptive execution framework του Spark SQL, και να αξιολογήσετε την απόδοση του συστήματος με διαφορετικά επίπεδα data skew και με διαφορετικές ρυθμίσεις του συστήματος χρησιμοποιώντας το TPC-DS benchmark.

Σχετικά Μαθήματα: Προχωρημένα θέματα βάσεων δεδομένων, Κατανεμημένα Συστήματα
Σχετική Βιβλιογραφία:

1. Y. Kwon, M. Balazinska, B. Howe, and J. Rolia. SkewTune: mitigating skew in mapreduce applications. In Proceedings of the 2012 ACM SIGMOD.
2. R. Li, M. Riedewald, and X. Deng. Submodularity of Distributed Join Computation. In Proceedings of the 2018 ACM SIGMOD.

3. W. Rödiger, S. Idicula, A. Kemper and T. Neumann. Flow-Join: Adaptive skew handling for distributed joins over high-speed networks. In Proccedings of the 2016 IEEE ICDE.

4. https://spark.apache.org/sql/

5. https://databricks.com/blog/2020/05/29/adaptive-query-execution-speeding-up-spark-sql-at-runtim
html

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

V. Εργασίες σε συνεπίβλεψη με εξωτερικούς συνεργάτες

28 Παραλληλοποίηση shell scripts

Ο προγραμματισμός με τη χρήση προγραμμάτων κελύφους (shell scripts) παραμένει εξαιρετικά δημοφιλής (8η πιο δημοφιλής γλώσσα προγραμματισμού στο Github) εξαιτίας των ιδιαίτερων χαρακτηριστικών του κελύφους, όπως η εύκολη σύνθεση προγραμμάτων γραμμένων σε διαφορετικές γλώσσες και η άμεση πρόσβαση και διαχείριση του συστήματος αρχείων (file system). Παρ'όλα αυτά, τα προγράμματα κελύφους εκτελούνται κυρίως σειριακά, οδηγώντας σε μεγάλους χρόνους εκτέλεσης, ιδιαίτερα για διαδικασίες επεξεργασίας δεδομένων.

Η πρόσφατη δημιουργία του PaSh (https://github.com/binpash/pash), ενός συστήματος αυτόματης παραλληλοποίησης προγραμμάτων κελύφους, έχει οδηγήσει σε σημαντική βελτίωση στο χρόνο εκτέλεσης τους. Το PaSh έχει λάβει αρκετά βραβεία από την ακαδημαική κοινότητα και είναι διαθέσιμο υπό την αιγίδα του Linux Foundation.

Σχετική Βιβλιογραφία:

1. Vasilakis, Nikos, Konstantinos Kallas, Konstantinos Mamouras, Achilles Benetopoulos, and Lazar Cvetković. "PaSh: light-touch data-parallel shell processing." In Proceedings of the Sixteenth European Conference on Computer Systems, pp. 49-66. 2021..

2. Konstantinos Kallas, Tammam Mustafa, Jan Bielak, Dimitris Karnikis, Thurston H.Y. Dang, Michael Greenberg, and Nikos Vasilakis. "Practically Correct, Just-in-Time Shell Script Parallelization." In Proceedings of the 16th USENIX Symposium on Operating Systems Design and Implementation.

28.1 Βέλτιστη Κατανομή shell scripts

Η αρχιτεκτονική του PaSh και οι τεχνικές παραλληλοποίησης επεκτείνονται ευθέως για την κατανομή προγραμμάτων κελύφους που διαχειρίζονται αρχεία σε τοπικά αλλά και κατανεμημένα συστήματα αρχείων (distributed file systems). Υπάρχει ήδη μια πρώτη υλοποίηση της κατανεμημένης επέκτασης του PaSh, η οποία όμως δεν είναι βέλτιστη. Συγκεκριμένα, αυτή η πρώτη κατανεμημένη υλοποίηση κάνει κάποιες υποθέσεις για τα χαρακτηριστικά κατανομής του υπολογιστικού συστήματος, τον τύπο των προγραμμάτων κελύφους, την κατανομή των δεδομένων, και τους στόχους του χρήστη. Αυτές οι υποθέσεις περιορίζουν την ευρεία χρήση του PaSh σε γενικότερα σενάρια τα οποία απαντώνται συχνά στην πράξη.

Στόχος: Η επέκταση του PaSh για τη βέλτιστη κατανομή προγραμμάτων κελύφους (shell scripts).

Σχετικά Μαθήματα: Συστήματα Παράλληλης επεξεργασίας, Κατανεμημένα Συστήματα

28.2 Αναζήτηση βέλτιστης παραλληλοποίησης

Το PaSh βελτιώνει σημαντικά το χρόνο εκτέλεσης για πολλά προγράμματα, αλλά επιλέγει διάφορες παραμέτρους παραλληλοποίησης (π.χ., τον βαθμό παραλληλισμού) χωρίς αναζήτηση, οδηγώντας συχνά σε μηδενική βελτίωση (ή ακόμα και χειροτέρευση) κάποιων προγραμμάτων. Αυτή η έλλειψη βελτίωσης μπορεί να οφείλεται στα χαρακτηριστικά του προγράμματος ή του υπολογιστικού συστήματος.

Για παράδειγμα, η παραλληλοποίηση κάποιων εντολών, π.χ., sort, δεν είναι γραμμική, με αποτέλεσμα ένας βαθμός παραλληλισμού πάνω από κάποιο σημείο να χειροτερεύει αντι να βελτιώνει το χρόνο εκτέλεσης τους. Ένα άλλο παράδειγμα είναι ότι, σε συστήματα με αργό δίσκο, οι εντολές buffering που προσθέτει το PaSh μπορεί να χειροτερεύουν τον χρόνο εκτέλεσης. Στόχος: Η βελτιστη παραλληλοποίηση shell scripts με τη χρήση στατικών (με ευριστικές) ή δυναμικών (με στατιστικά δεδομένα) μεθόδων.

Σχετικά Μαθήματα: Συστήματα Παράλληλης επεξεργασίας

28.3 Παραλληλοποίηση νέων και διαφορετικών εφαρμογών

Το PaSh καταφέρνει να παραλληλοποιήσει ένα σημαντικό αριθμό προγραμμάτων, αλλά το εύρος των εφαρμογών που μπορεί να βελτιωθεί δεν έχει διερευνηθεί πλήρως. Μεταξύ άλλων, πιθανές κλάσεις εφαρμογών στις οποίες μπορεί το PaSh να βρει εφαρμογή είναι:

- Προγράμματα κελύφους για εγκατάσταση λογισμικού (configure και build scripts)
- Προγράμματα για εφαρμογές βιοϊατρικής και βιοπληροφορικής (bioinformatics)
- Προγράμματα επεξεργασίας δεδομένων με πιο περίπλοκη δομή, π.χ., προγράμματα μηχανικής μάθησης (ML)

Στόχος: Η παραλληλοποίηση νέων και διαφορετικών εφαρμογών, περιλαμβάνοντας τα ακόλουθα βήματα:

- Παραλληλοποίηση καινούργιων προγραμμάτων με το χέρι (ή άλλα εργαλεία, π.χ., GNU-parallel) που μπορούν να χρησιμοποιηθούν ως βάσεις μελέτης για περαιτέρω βελτίωση του PaSh.
- Παραλληλοποίηση καινούργιων προγραμμάτων κελύφους με τη χρήση του PaSh, επεκτείνοντας το PaSh με συσσωρευτές (aggregators) και περιγραφές (annotations) για νέες εντολές, αλλά πιθανόν και διαμορφώνοντας τα προγράμματα για να απλοποιηθεί η παραλληλοποίησή τους.
- Πιο εκτενής επέκταση του PaSh, με νέες τεχνικές παραλληλοποίησης ώστε να δράσει σε προγράμματα στα οποία δεν μπορούσε στο παρελθόν

Σχετικά Μαθήματα: Συστήματα Παράλληλης επεξεργασίας

Επικοινωνία: Γεώργιος Γκούμας , goumas@cslab.ece.ntua.gr
Κωνσταντίνος Καλλάς, kallas@seas.upenn.edu, https://angelhof.github.io/
Νίκος Βασιλάκης nikos@vasilak.is, http://nikos.vasilak.is/

29 Anomaly detection of applications using eBPF

Applications interact with the OS using system calls. The goal of this project is to use eBPF to to monitor an application's system calls to build models, that can be, subsequently, used to detect abnormal application behavior.

Milestones

Application selection
Select an application that will to be used to build and evalute the proposed system. Commonly used OSS projects, such as nginx, coredns, redis, memcache, mySQL, postgreSQL, are well-suited for our purposes. In addition to the application, find a workload generator (in most cases one should exist already) as well as a number of "abnormal" conditions for these application. These can be anything ranging from: bugs, security exploits, and user misconfiguration (Note that for the former two categories, older versions of the software can be used to trigger them).

Collect data and build an application model
using eBPF collect system call data from applications. The data will include: a timestamp, the thread id, the system call executed and its arguments. Using these data, build an application model that can be used to detect abnormal application behavior. There are numerous methods for doing that [1], [2]. We suggest using Probabilistic suffix trees [3], but any suitable method can be used. Evaluate the application model using the data from the previous steps.

Literature:
[1] Outlier Analysis, Aggarwal, Chapter 10.
[2] Knowledge Discovery from Data Streams, Chapter 9
[3] https://www.jstatsoft.org/article/download/v072i03/1035

Σχετικά Μαθήματα: Εργαστήριο Λειτουργικών Συστημάτων

Επικοινωνία: Γεώργιος Γκούμας , goumas@cslab.ece.ntua.gr
Κορνήλιος Κούρτης, kkourt@kkourt.io, https://kkourt.io/

30 Επιτάχυνση Επικοινωνίας Εφαρμογών Υπολογιστικών Κέντρων με eBPF

Εφαρμογές που τρέχουν μέσα σε μεγάλα υπολογιστικά κέντρα, όπως key-value stores και map reduce, επικοινωνούν με πολύ συγκεκριμένα μοτίβα γνωστά ως fan-out/fan-in patterns. Κατά τη διάρκεια αυτής της επικοινωνίας, η εφαρμογή πελάτης (client) χρειάζεται να επικοινωνήσει με εκατοντάδες ή και χιλιάδες εφαρμογές εξυπηρετητή (server) και αφού συλλέξει όλες ή τις περισσότερες απαντήσεις, να συνθέσει μια τελική απάντηση. Κατά τη διάρκεια αυτής της επικοινωνίας, ο χώρος πυρήνα χρειάζεται να επικοινωνεί με το χώρο χρήστη για κάθε απάντηση που λαμβάνει, παρόλο που η εφαρμογή στο χώρο χρήστη θα μπορέσει να τρέξει μόνο όταν έχει συλλέξει όλες τις απαντήσεις.

Το BPF (extended Berkeley Packet Filter) [1], είναι μια τεχνολογία που επιτρέπει την ασφαλή εκτέλεση κώδικα χρήστη στο χώρο του πυρήνα. Επομένως, επιτρέπει την επέκταση του πυρήνα με βάση τις ανάγκες του χρήστη ή κάποιας εφαρμογής χωρίς αλλαγές στον κώδικα του πυρήνα και χωρίς επεκτάσεις. Σκοπός αυτής της εργασίας είναι ο σχεδιασμός και η υλοποίηση ενός συστήματος σε eBPF που

θα βελτιώσει τα συγκεκριμένα μοτίβα επικοινωνίας των εφαρμογών υπολογιστικών κέντρων, μειώνοντας την ανάγκη επικοινωνίας ανάμεσα στο χώρο χρήστη και πυρήνα, και υλοποιώντας κάποια από την λογική της εφαρμογής μέσα στο χώρο πυρήνα.

Σχετική βιβλιογραφία:
[1] https://ebpf.io/
https://www.usenix.org/conference/nsdi21/presentation/ghigoff
https://dl.acm.org/doi/10.1145/3419111.3421301

Σχετικά Μαθήματα: Εργαστήριο Λειτουργικών Συστημάτων

Επικοινωνία: Γεώργιος Γκούμας , goumas@cslab.ece.ntua.gr
Μάριος Κόγιας marioskogias@gmail.com, https://marioskogias.github.io/

31 Χρονοπρογραμματισμός Εφαρμογών σε Κλιμακα microsecond με τη Βοήθεια Προγραμματιζόμενων Συσκευών Δικτύου

Εφαρμογές υπολογισ τικών κέντρων, όπως in-memory key-value stores, χρησιμοποιούν Remote Procedure Calls (RPCs) για να επικοινωνήσουν. Η εξυπηρέτηση αυτ ών των RPCs διαρκεί μόνο μερικά microseconds και οι συγκεκριμένες εφαρμογές χρειάζεται να λειτουργήσουν κάτω από πολύ αυστηρά Service Level Objects (SLOs). Επομένως, ο χρονοπρογραμματισμός τέτοιων εφαρμογών είναι ένα δύσκολο και ανοιχτό πρόβλημα.

Ένας καινούργιος τρόπος υλοποίησης κατανεμημένων εφαρμογών βασίζεται στη χρήση νέων προγραμματιζόμενων συσκευών δικτύου, όπως switches και NICs. Το νέο αυτό μοντέλο (in-network compute) γίνεται όλο και πιο προσφιλές αφού ελευθερώνει υπολογιστικούς πόρους που πλέον μπορούν να χρησιμοποιηθούν για άλλους σκοπούς. Οι συσκευές αυτές μπορούν να προγραμματιστούν με νέες γλώσσες προγραμματισμού ειδικού σκοπού όπως η P4. Η εργασία αυτή αφορά τη χρήση τέτοιων προγραμματιζόμενων συσκευών στο χρονοπρογραμματισμό RPCs. Σκοπός είναι η υλοποίηση ενός αλγορίθμου χρονοπρογραμματισμού σε προγραμματιζόμενα switches που θα προσαρμόζεται δυναμικά σε διαφορετικά φορτία, θα μπορεί να εξυπηρετήσει διαφορετικά είδη εφαρμογών ανεξάρτητα από την κατανομή του χρόνου εξυπηρέτησης, και θα εκμεταλλεύεται πληροφορίες που δέχεται από τις εφαρμογές και τις ανάγκες τους.

Σχετική βιβλιογραφία:
https://p4.org/
https://www.usenix.org/conference/atc19/presentation/kogias-r2p2
https://dl.acm.org/doi/abs/10.1145/3477132.3483571

Σχετικά Μαθήματα: Εργαστήριο Λειτουργικών Συστημάτων

Επικοινωνία: Γεώργιος Γκούμας , goumas@cslab.ece.ntua.gr
Μάριος Κόγιας marioskogias@gmail.com, https://marioskogias.github.io/

32 Βελτιστοποίηση του υπολογιστικού πυρήνα πολλαπλασιασμού αραιού πίνακα με πυκνό πίνακα (Sparse-Matrix-Dense-Matrix Multiplication) σε αρχιτεκτονικές με επεξεργασία κοντά στη μνήμη (Processing-In-Memory)

Οι πρόσφατες εξελίξεις στην αρχιτεκτονική 3D-stack τεχνολογιών μνήμης (για παράδειγμα High-Bandwidth Memory, HBM) έχουν ανανεώσει το ενδιαφέρον για Επεξεργασία Κοντά στη Μνήμη, ή αλλιώς Processing-In-Memory (PIM). Οι PIM αρχιτεκτονικές έχουν σχεδιαστεί με στόχο να μειώσουν την κίνηση δεδομένων (data movement) μεταξύ του επεξεργαστή και της κύριας μνήμης, τοποθετώντας πυρήνες χαμηλής κατανάλωσης ενέργειας και μικρού κόστους κοντά στην κύρια μνήμη. Πρόσφατες εργασίες [1-3] δείχνουν τα οφέλη των NDP αρχιτεκτονικών για παράλληλες εφαρμογές όπως γραμμική άλγεβρα, βιοπληροφορική, ανάλυση δεδομένων και βάσεις δεδομένων. Στην παρούσα διπλωματική εργασία θα μελετηθεί η υλοποίηση και η βελτιστοποίηση του υπολογιστικού πυρήνα πολλαπλασιασμού αραιού πίνακα με πυκνό πίνακα σε PIM αρχιτεκτονικές, και συγκεκριμένα στην UPMEM αρχιτεκτονική [4]. Θα υλοποιηθεί μία βιβλιοθήκη για αυτόν τον υπολογιστικό πυρήνα βασισμένη στην υπάρχουσα SparseP βιβλιοθήκη [3], και θα εφαρμοστούν τεχνικές βελτιστοποιήσης με στόχο την επίτευξη της μέγιστης δυνατής επίδοσης στις συγκεκριμένες αρχιτεκτονικές.

Σχετικά Μαθήματα: Προηγμένα Θέματα Αρχιτεκτονικής Υπολογιστών, Συστήματα Παράλληλης Επεξεργασίας

Σχετική Βιβλιογραφία:

1. Christina Giannoula et al., "SparseP: Towards Efficient Sparse Matrix Vector Multiplication on Real Processing-In-Memory Architectures", ACM SIGMETRICS 2022.
2. Juan Gómez-Luna et al., "Benchmarking a New Paradigm: Experimental Analysis and Characterization of a Real Processing-in-Memory System", IEEE Access 2022.
3. `https://github.com/CMU-SAFARI/SparseP`
4. `https://www.upmem.com/`

Επικοινωνία: Γεώργιος Γκούμας, goumas@cslab.ece.ntua.gr
Χριστίνα Γιαννούλα, christina.giann@gmail.com, https://cgiannoula.github.io/
Juan Gómez-Luna, el1goluj@gmail.com, https://scholar.google.com/citations?user=UXlFLY0AAAAJ

33 Βελτιστοποίηση αλγορίθμων για εφαρμογές Big Data Analytics

Η εκτίμηση πληθικότητας (cardinality estimation, το πλήθος μοναδικών αντικειμένων σε μία σειρά) είναι ένα σύνηθες πρόβλημα στη διαχείριση μεγάλου όγκου δεδομένων. Καθώς το κόστος συμβατικών λύσεων (πχ hash tables) είναι απαγορευτικό, το ενδιαφέρον έχει στραφεί σε πιθανοτικές δομές δεδομένων, παρέχοντας καλές εκτιμήσεις πολύ αποδοτικά. Αυτή η διπλωματική εργασία α) θα μελετήσει τις διάφορες διαθέσιμες λύσεις, β) θα υλοποιήσει μία ή περισσότερες από αυτές και γ) θα τις αξιολογήσει σε ένα παραγωγικό (production) analytics περιβάλλον, συγκρίνοντας με υπάρχουσες υλοποιήσεις.

Σχετική βιβλιογραφία:

https://en.wikipedia.org/wiki/Count-distinct$_problem$
https://en.wikipedia.org/wiki/HyperLogLog

Σχετικά Μαθήματα: Συστήματα Παράλληλης Επεξεργασίας, Κατανεμημένα Συστήματα

Επικοινωνία: Γεώργιος Γκούμας , goumas@cslab.ece.ntua.gr
Φώτης Ξενάκης, Fotios.Xenakis@mobileum.com, https://www.linkedin.com/in/fotis-xenakis/

34 Βελτιστοποιήσεις σε συστήματα βάσεων δεδομένων χρονοσειρών (time series databases)

Οι βάσεις δεδομένων χρονοσειρών λαμβάνουν πυκνές ροές (streams) εγγραφών σε πραγματικό χρόνο και επιτρέπουν την αποδοτική εκτέλεση ερωτημάτων (queries) σε αυτές (πιθανώς κατόπιν ευρετηρίασης, indexing). Τα workloads τους συνήθως είναι write-heavy, ενώ το γεγονός ότι υποστηρίζουν απλές λειτουργίες στα δεδομένα τους επιτρέπει να κλιμακώνουν και συχνά να πιέζουν τα όρια του hardware, σε CPU και I/O. Σε αυτό το πλαίσιο, η παρούσα διπλωματική εργασία θα μπορούσε να εξερευνήσει προβλήματα όπως:

- Παράλληλη εγγραφή buffer στο δίσκο (με συμπίεση): εξερεύνηση του design space, υλοποίηση και αξιολόγηση.
- Ταυτόχρονη αναδιάταξη buffer κατά την εγγραφή (concurrent insertion buffer reordering): εξερεύνηση των επιλογών για την αποφυγή της εισαγωγής εκτός σειράς (out-of-timestamp- order), υλοποίηση και αξιολόγηση.

Σχετική βιβλιογραφία:
https://en.wikipedia.org/wiki/Time$_s eries_d atabase$
https://facebook.github.io/zstd/
http://lz4.github.io/lz4/
https://en.wikipedia.org/wiki/Skip$_l ist$

Σχετικά Μαθήματα: Συστήματα Παράλληλης Επεξεργασίας, Προχωρημένα Θέματα Βάσεων Δεδομένων, Λειτουργικά Συστήματα

Επικοινωνία: Γεώργιος Γκούμας , goumas@cslab.ece.ntua.gr
Φώτης Ξενάκης, Fotios.Xenakis@mobileum.com, https://www.linkedin.com/in/fotis-xenakis/

35 Ταυτόχρονες δομές δεδομένων

Ενώ ο παράλληλος υπολογισμός παρέχει μία οδό για υψηλότερες επιδόσεις, ταυτόχρονα σχηματίζει ένα πιο περίπλοκο περιβάλλον προγραμματισμού. Για να κλιμακώσει ένα πρόγραμμα σε περισσότερους επεξεργαστές, χρειάζεται να διαιρέσουμε την εργασία σε περισσότερα κομμάτια. Η ιδανική διαίρεση απαιτεί απλώς πολλαπλά στιγμιότυπα του ίδιου προγράμματος να εργάζονται σε διαφορετικά κομμάτια του προβλήματος. Μία τέτοια διαίρεση είναι συχνά αδύνατη, ανούσια ή μη πρακτική, οπότε χρειαζόμαστε σε ένα βαθμό την αλληλεπίδραση μεταξύ διαφορετικών ροών εκτέλεσης (πχ threads). Εκεί έρχονται οι ταυτόχρονες (concurrent) δομές δεδομένων να προσφέρουν έναν οργανωμένο τρόπο χειρισμού του κοινού state ανάμεσα σε τμήματα που εκτελούνται σε διακριτές CPUs. Αυτή η διπλωματική εργασία θα μελετήσει, υλοποιήσει και αξιολογήσει την επίδοση κάποιας concurrent δομής δεδομένων σε ένα παράλληλο περιβάλλον εκτέλεσης, για παράδειγμα:

- Concurrent hash table
- Concurrent aging map
- Longest prefix match trie

Σχετική βιβλιογραφία:
https://en.wikipedia.org/wiki/Concurrent$_h ash_t able$
https://en.wikipedia.org/wiki/Longest$_p refix_m atch$

Σχετικά Μαθήματα: Συστήματα Παράλληλης Επεξεργασίας

Επικοινωνία: Γεώργιος Γκούμας , goumas@cslab.ece.ntua.gr
Φώτης Ξενάκης, Fotios.Xenakis@mobileum.com, https://www.linkedin.com/in/fotis-xenakis/

36 Μέτρηση latency σε επίπεδο αρχιτεκτονικής για δικτυακές εφαρμογές

Οι εφαρμογές που είναι ευαίσθητες στο latency, όπως οι εφαρμογές δικτύωσης, δεν εξυπηρετούνται από δειγματοληπτικές (sampling) τεχνικές (πχ profiling) για την ανάλυση επίδοσης, μιας και τα latency spikes είναι συνήθως σπάνια και απρόβλεπτα. Ο εντοπισμός των αιτιών πίσω από τέτοια latency spikes απαιτεί ιχνηλάτηση (tracing) κάθε αιτήματος (request) ή ακολουθίας επεξεργασίας κάποιου γεγονότος (event), με τρόπο αποδοτικό και χωρίς παρεμβολές. Οι υπερ-υψηλές επιδόσεις των σύγχρονων δικτύων πιέζουν το κομμάτι της επεξεργασίας κάθε αίτηματος ή γεγονότος να ολοκληρωθεί σε πολύ σύντομο χρονικό περιθώριο (στις ακραίες περιπτώσεις μετράται σε κύκλους ρολογιού). Κατά συνέπεια, ένας τέτοιος μηχανισμός μέτρησης latency πρέπει να επιτυγχάνει υψηλή ακρίβεια. Τέτοιοι μηχανισμοί συχνά υλοποιούνται με instrumentation του κώδικα της εφαρμογής. Αυτή η διπλωματική εργασία α) θα εξερευνήσει τις διάφορες τεχνολογίες instrumentation και β) θα συγκρίνει το κόστος και την ακρίβεια τους σε παραγωγικό (production) περιβάλλον σύγχρονων δικτυακών συστημάτων υψηλών επιδόσεων.

Σχετική βιβλιογραφία:
https://gcc.gnu.org/onlinedocs/gcc/Instrumentation-Options.html
https://dynamorio.org/
https://www.dyninst.org/
https://www.intel.com/content/www/us/en/developer/articles/tool/pin-a-dynamic-binary-instrumentation-tool.html
https://www.intel.com/content/www/us/en/developer/tools/oneapi/vtune-profiler.htmlgs.f1qe9f
https://www.intel.com/content/www/us/en/developer/tools/oneapi/vtune-profiler.htmlgs.f1qe9f
https://www.intel.com/content/dam/www/public/us/en/documents/white-papers/ia-32-ia-64-benchmark-code-execution-paper.pdf

Σχετικά Μαθήματα: Προηγμένα Θέματα Αρχιτεκτονικής Υπολογιστών, Μεταγλωττιστές

Επικοινωνία: Γεώργιος Γκούμας , goumas@cslab.ece.ntua.gr
Φώτης Ξενάκης, Fotios.Xenakis@mobileum.com, https://www.linkedin.com/in/fotis-xenakis/

37 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

38 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

39 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 $A \to B \to C \to D$, we can accelerate the execution of a second pipeline $A \to B \to C \to D'$ 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

40 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

41 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

42 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

43 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

44 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

45 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

46 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