**Hard-to-Predict (H2P) σύμφωνα με το paper:**

*Branch Prediction Is Not A Solved Problem: Measurements, Opportunities, and Future Directions.pdf* (βρίσκεται στο drive):

Σύμφωνα με το Paper:

Παρά την πρόοδο στους predictors βλέπουμε ότι υπάρχει ακόμα μεγάλο περιθώριο για σωστό branch prediction. Ειδικά αν κάνουμε scale up το capacity του pipeline ενός επεξεργαστή, καλύτερο branch prediction συνεπάγεται και καλύτερη αξιοποίηση του pipeline άρα και καλύτερη απόδοση του επεξεργαστή. Παρατηρούμε ότι ένα μεγάλο μέρος των λανθασμένων προβλέψεων προκαλούνται κυρίως από κάποια λίγα συγκεκριμένα branches (Hard-to-Predict branches). Εάν βρούμε έναν τρόπο να προβλέψουμε σωστά αυτά τα branches θα κερδίσουμε ένα σημαντικό IPC boost.

Εντοπισμός των H2P branches:

1. Τρέχουμε τα benchmarks με διαφορετικα workloads (κάθε benchmark δέχεται ενα input i.e. workload, π.χ. το 602.gcc\_s κάνει compile C κώδικα, τα input είναι διαφορετικοι C source κώδικες)
2. Για κάθε trace, ανα 30M εντολές, εντοπίζουμε branches που:

1) εμφανίζονται τουλάχιστον 15.000 φορές,

2) δημιουργούν τουλάχιστον 1000 misspredictions και

3) έχουν accuracy λίγοτερο απο 99% κάτω από τον Tage 8ΚΒ predictor.

Τα κριτήρια επιλέγονται με τη λογική: πολλά misspredicts (άρα περιθώρια βελτείωσης) και πολλά occurrences άρα κατάληλλα για machine learning προσέγγιση.

1. Εντοπίζουμε κάποια λίγα branches (5 εως 50 ανάλογα το benchmark). Αυτά τα PC βλέπουμε ότι εμφανίζονται πάντα ανεξάρτητα απο το workload (input) που έχουμε όταν τρέχουμε το benchmark.
2. Βλέπουμε ότι αν γίνει σωστή πρόβλεψη των συγκεκριμένων branches έχουμε μια αύξηση στο IPC, ειδικά όταν κάνουμε scale up το pipeline (reorder buffer size, fetch width, execution width, load/store queues) το IPC αυξάνεται ακόμα περισσότερο.
3. Παρατηρούμε ακόμα ότι αύξηση των resources του Tage Predictor (π.χ. απο 8ΚΒ σε 64ΚΒ) δεν συνεπάγεται και καλύτερη απόδοση στα συγκεκριμένα branches.

Σύγκριση αποτελεσμάτων:

Στο paper βρίσκουν H2P από το ίδιο benchmark, χρησιμοποιώντας πολλά διαφορετικά application inputs (workloads). Εμείς βρίσκουμε H2P μόνο στο 1 input που έχουμε διαθέσιμο για κάθε benchmark.

Στο paper:

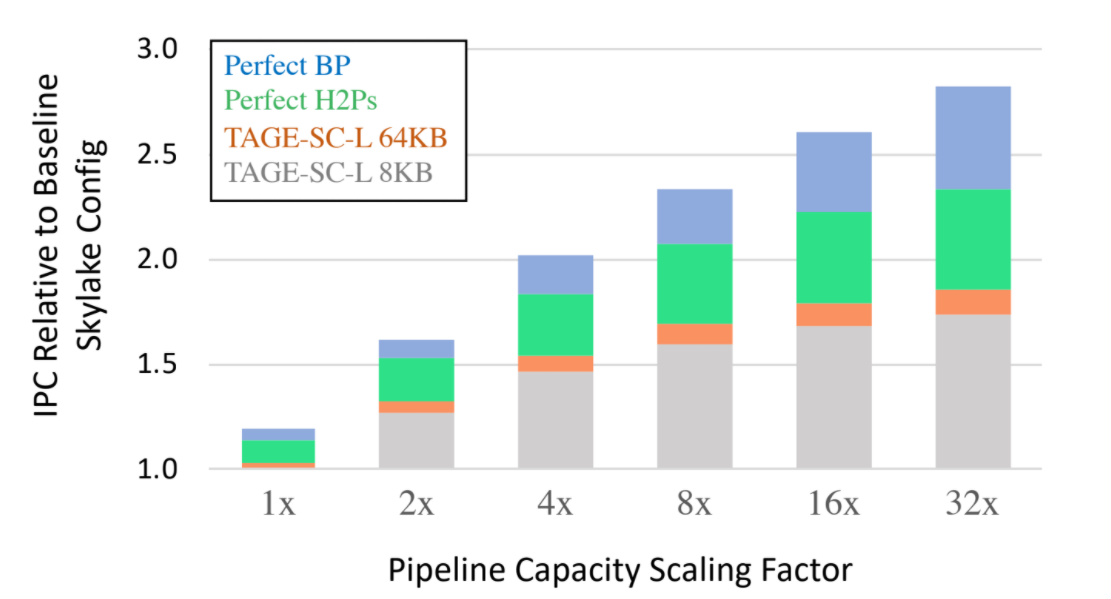
|  |  |  |
| --- | --- | --- |
| **Benchmark** | **# workloads** | **Avg. H2P per workload** |
| 600.perlbench | 4 | 21.5 |
| 605.mcf | 8 | 19.0 |
| 620.omnetpp | 5 | 28.0 |
| 623.xalancbmk | 4 | 14.5 |
| 625.x264 | 14 | 6.0 |
| 631.deepsjeng | 12 | 40.0 |
| 641.leela | 10 | 56.5 |
| 648.exchange2 | 5 | 20.0 |
| 657.xz | 5 | 63.0 |

Εμείς:

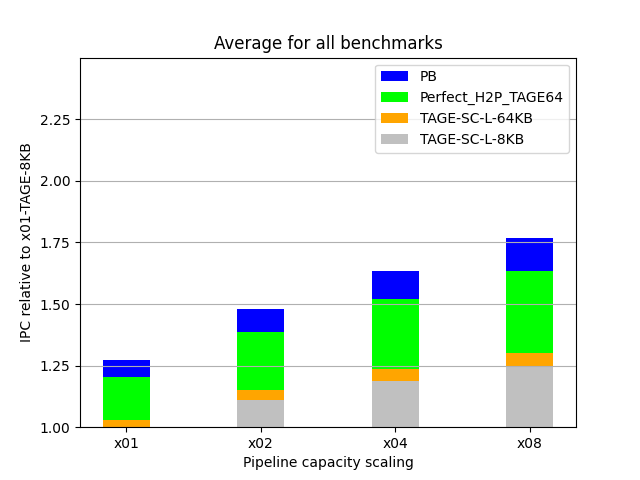
|  |  |
| --- | --- |
| **Benchmark** | **H2P** |
| 600.perlbench | 12 |
| 605.mcf | 18 |
| 620.omnetpp | 10 |
| 623.xalancbmk | 4 |
| 625.x264 | 5 |
| 631.deepsjeng | 32 |
| 641.leela | 50 |
| 648.exchange2 | 20 |
| 657.xz | 61 |

IPC boost αν έχουμε σωστό prediction των παραπάνω Branches:

Στο Paper:



Εμείς:



**Στη συνέχεια στο paper:**

*Improving Branch Prediction By Modeling Global History with Convolutional Neural Networks.pdf* (βρίσκεται στο drive)

1. Δημιουργεί CNN νευρωνικό το οποίο δέχεται input πίνακα 200x256. Κάθε γραμμή απο τις 200 συμβολίζει με τη σειρά τα προηγούμενα αποτελέσματα των branches που έχουν παρέλθει. Κάθε γραμμή είναι γεμάτη με 0 και έχει έναν και μόνο άσσο σε μία απο τις 256 θέσεις σύμφωνα με: Χρησιμοποιούμε τα 7 LSB του Program Counter (PC), κάνουμε 1 logic shift left και προσθέτουμε 0 ή 1 ανάλογα με το αποτέλεσμα του συγκεκριμένου branch. Για παράδειγμα αν θέλουμε να κωδικοποιήσουμε το 10ο στη σειρά branch με PC 8325735 και αποτέλεσμα True, τότε στην 10η σειρά του πίνακα θα γεμίσουμε μηδενικά και έναν μόνο άσσο. Για να βρούμε τη θέσου του άσσου αρκεί να πάρουμε τα 7LSB του PC, δηλαδή τον δεκαδικό αριθμό 103, να κάνουμε shift logic left, άρα γίνεται 206 και να προσθέσουμε 1, αφού το αποτέλεσμα ήταν True, άρα στη στήλη 207 (σειρά 10) βάζουμε άσσο.
2. Στο paper βλέπουμε στα αποτελέσματα ότι κάνοντας train για κάποιο συγκεκριμένο benchmark και workload, το νευρωνικό αποδίδει καλά για οποιοδήποτε άλλο workload του ίδιου benchmark. Συνεπώς μπορούμε να φτιάξουμε κάποια αρχιτεκτονική η οποία αποδίδει καλά σε στοχευμένα προβλήματα το οποίο μπορεί να αξιοποιηθεί π.χ. σε κάποιο datacenter που επιτελούν συγκεκριμένες εργασίες.
3. Γνωρίζουμε για το benchmark (ή οποιδήποτε application) που έχουμε εκ των προτέρων τα H2P Program counter και κάθε φορά που συναντάμε ένα, χρησιμοποιούμε το νευρωνικό για την πρόβλεψη το οποίο αποδίδει καλύτερα απο τους κλασσικούς predictors για τα συγκεκριμένα branch. Το αποτέλεσμα είναι ενα IPC boost σε κάποιο συγκεκριμένο application ή διαδικασία, ασχέτα με το input το οποίο μπορεί να αλλάζει.

**Πρώτα αποτελέσματα:**

Δεν έχουμε πρόσβαση στα benchmarks ώστε να μπορέσουμε να δοκιμάσουμε με διαφορετικά workloads. Ο μόνος τρόπος για να προσεγγίσουμε το ζήτημα, είναι να κάνουμε Train σε κάποιο simpoint και επαληθεύοντας την απόδοση του νευρωνικού σε κάποιο άλλο simpoint.

Βλέπουμε ότι παρά τα διαφορετικά SimPoints τα H2P είναι κοινά, για παράδειγμα το SimPoint 625.x264\_s-39B περιλαμβάνει όλα τα H2P που υπάρχουν στα υπόλοιπα SimPoint του 625 (ωστόσο εντοπίζουμε διαφορετικό ιστορικό). Αντίστοιχα στο 600.perlbench έχουμε σε κάθε simpoint από 10 H2P τα οποία είναι κοινά στα διαφορετικά SimPoint. Από πειράματα βλέπουμε ότι κάνοντας training σε κάποιο SimPoint, το νευωρωνικό μαθαίνει πως να προβλέπει τα H2P branches του συγκεκριμένου benchmark και κάνει generalize σε οποιδήποτε διαφορετικό ιστορικό μπορεί να προκύψει πριν από κάποιο τέτοιο branch.

Στο excel φαίνονται τα αποτελέσματα απο τα πρώτα νευρωνικά. Είναι όλα lstm μοντέλα. Τα πρώτα 6 δεχόνται input το ιστορικό True/False των τελευταίων 200 branches, χωρίς να υπάρχει καμία πληροφορία για το PC του κάθε branch. Στα επόμενα βλέπουμε τα αποτελέσματα όταν το input είναι 200 tuples (True/False, encodedPC). Το encodedPC είναι ενας αριθμός απο το 0 έως το 256 και η κωδικοποιήση γίνεται οπως περιγράφεται παραπάνω. Σημειώνεται ότι από το input tuple (True/False, PC) (όλοκληρο το PC) το νευρωνικό δεν απέδιδε καθόλου ενώ αντίθετα με κωδικοποίηση του αριθμού αποδίδει πολύ καλά.

Στο excel βλέπουμε την αρχιτεκτονική του μοντέλου (2η στήλη) (τα layers του νευρωνικού), τι input χρησιμοποιήθηκε (ποιο benchmark, ποια samples) και πως απέδωσε το νευρωνικό, είτε σε validation set (ιστορικό απο το ίδιο simpoint στο οποίο όμως δεν έχει γίνει training το νευρωνικό), είτε σε προβλέποντας H2P σε ιστορικό απο διαφορετικό SimPoint του ιδιου benchmark.

**Επόμενα βήματα:**

1. Να ελέγξουμε άλλα inputs για τα μοντέλα που έχουμε ήδη δεί. Π.χ. αντί για input 200 tuples (encodedPC, True/False), να έχουμε 1 hot encoded πίνακα. Ή ακόμα να δώσουμε παραπάνω πληροφορίες στο νευρωνικό (αντί μόνο του ιστορικού των branches). Όπως π.χ. πληροφοριες για το περιεχόμενο των registers, πριν απο το H2P branch.
2. Να δοκιμασουμε άλλα μοντέλα όπως CNN, Transformer και GRU. Να δούμε πως αποδίδουν και κυρίως ποιο απο αυτά μπορεί να υλοποιηθεί στο hardware.
3. Να ενσωματώσουμε το νευρωνικό στον ChampSim και να «τρέξουμε» τα benchmark χρησιμοποιώντας προβλέψεις του νευρωνικού και να δούμε τι IPC gain έχουμε. (Ιδανικά άν έχουμε νευρωνικό με 100% accuracy θα δούμε IPC gain όπως στη γραφική παράσταση παραπάνω).