Ατομική Διπλωματική Εργασία

Detecting Wrong Path Events in the Speculative Branch Resolution for Improving Performance and Security

Μάριος Κελεπέσιης

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2019

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Detecting Wrong Path Events in the Speculative Branch Resolution for Improving Performance and Security

Μάριος Κελεπέσιης

Επιβλέπων Καθηγητής
Γιαννάκης Σαζεΐδης

Η Ατομική Διπλωματική Εργασία υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων απόκτησης του πτυχίου Πληροφορικής του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου

Μάιος 2019

Ευχαριστίες

Θα ήθελα να ευχαριστήσω ιδιαίτερα τον Επιβλέποντα Καθηγητή μου Δρ. Γιαννάκη Σαζεΐδη για την πολύτιμη καθοδήγησή του, την ενθάρρυνση και τις συμβουλές που μου παρέδωσε κατά τη συγγραφή της διπλωματικής μου εργασίας. Το βαθύ ενδιαφέρον και οι πλούσιες γνώσεις του Δρ. Σαζεΐδη για το πεδίο της αρχιτεκτονικής υπολογιστών, αποτέλεσε πηγή έμπνευσης για μένα και η καθοδήγησή του έχει κάνει την προσπάθειά μου συναρπαστική εμπειρία.

Επιπλέον θα ήθελα να ευχαριστήσω όλους τους καθηγητές μου από τους οποίους έλαβα σημαντικές γνώσεις και με βοήθησαν στο να ανελιχθώ σε Επιστήμονα Πληροφορικής μέσα από τα 4 χρόνια σπουδών μου στο Τμήμα Πληροφορικής του Πανεπιστήμιο Κύπρου.

Περίληψη

Η κερδοσκοπική εκτέλεση (speculative execution) σε λανθασμένα μονοπάτια καταλήγει να επιβαρύνει το χρόνο εκτέλεσης των προγραμμάτων εκτελώντας άσκοπες εντολές. Επιπρόσθετα, όπως ανακαλύφθηκε πρόσφατα, η κερδοσκοπική εκτέλεση (speculative execution) σε λανθασμένο μονοπάτι, δημιουργία κενά ασφάλειας που μπορεί κάποιος να εκμεταλλευτεί κακόβουλα για να αντλήσει δεδομένα από την μνήμη του υπολογιστή παρακάμπτοντας ελέγχους ασφάλειας.

Κατά την διάρκεια της μελέτης αυτής, εστίασα στην μείωση του φαινομένου της λανθασμένης κερδοσκοπικής εκτέλεσης, προσπαθώντας να μειώσω το Penalty από τα Miss-Predictions.

Η βασική ιδέα για επίτευξη του στόχου αυτού, βρίσκεται στο ότι κατά την διάρκεια εκτέλεσης λανθασμένου μονοπατιού στην κερδοσκοπική εκτέλεση, συμβαίνουν γεγονότα μέσα στον επεξεργαστή τα οποία μπορούν να «προδώσουν» το γεγονός ότι η κερδοσκοπική εκτέλεση βρίσκεται σε λάθος μονοπάτι. Αν τα γεγονότα αυτά εντοπιστούν εγκαίρως, δηλαδή πριν να προλάβει να ολοκληρώσει την εκτέλεση του το εν λόγω Miss-Predicted Branch, τότε θα μπορούσαμε να σταματήσουμε την λανθασμένη κερδοσκοπική εκτέλεση σε εκείνο το σημείο, αποφεύγοντας να πληρώσουμε ολόκληρο το κόστος από το Miss-Prediction Penalty.

Κατά την διάρκεια της Διπλωματικής μου Εργασίας, πρότεινα κάποια γεγονότα τα οποία έκρινα ότι υποδηλώνουν εγκαίρως την κερδοσκοπική εκτέλεση σε λάθος μονοπάτι, και ακολούθως τα αξιολόγησα βάση της συχνότητας εμφάνισης τους. Ακολούθως, προχώρησα στην θεωρητική αξιολόγησή του μηχανισμού αυτού και στην δυνητική ωφέλεια που θα προσέφερε στο χρόνο εκτέλεσης των προγραμμάτων.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1.1 Κίνητρο της Διπλωματικής Εργασίας

Οι σύγχρονοι επεξεργαστές χρησιμοποιούν διάφορες τεχνικές για να επιτύχουν την μεγιστοποίηση της απόδοσης τους. Κάποιες από αυτές είναι το out-of-order execution [1], branch prediction [2] και speculative execution. Παρότι, οι τεχνικές αυτές κατόρθωσαν να βελτιώσουν σε ένα σημαντικό βαθμό την επίδοση των επεξεργαστών, υπάρχει περιθώριο βελτίωσης όταν οι τεχνικές αυτές αποτυγχάνουν. Μια τέτοια περίπτωση είναι όταν η κερδοσκοπική εκτέλεση (speculative execution) γίνεται στο λανθασμένο μονοπάτι, καταλήγοντας να επιβαρύνει το χρόνο εκτέλεσης εκτελώντας άσκοπες εντολές.

Επιπρόσθετα, όπως ανακαλύφθηκε πρόσφατα, η κερδοσκοπική εκτέλεση (speculative execution) σε λανθασμένο μονοπάτι, δημιουργία κενά ασφάλειας που μπορεί κάποιος να εκμεταλλευτεί κακόβουλα για να αντλήσει δεδομένα από την μνήμη του υπολογιστή παρακάμπτοντας ελέγχους ασφάλειας.

Τα προβλήματα αυτά αποτέλεσαν κίνητρο για ενασχόληση με την κερδοσκοπική εκτέλεση (speculative execution) σε λάθος μονοπάτι, με σκοπό εύρεσης κάποιας τεχνικής για μείωση του φαινομένου αυτού.

1.2 Ιδέα της Διπλωματικής Εργασίας

Για να καταφέρουμε να μειώσουμε τους κύκλους μηχανής που χρησιμοποιούνται για κερδοσκοπική εκτέλεση εντολών σε λάθος μονοπάτι, θα πρέπει να επιτύχουμε είτε βελτίωση της επίδοσης των Branch Predictors, είτε να μειώσουμε το Penalty από τα Miss-Predictions. Κατά την μελέτη της Διπλωματικής μου Εργασίας, εστίασα στην μείωση του Penalty από τα Miss-Predictions.

Η βασική ιδέα για επίτευξη του στόχου αυτού, βρίσκεται στο ότι κατά την διάρκεια εκτέλεσης λανθασμένου μονοπατιού στην κερδοσκοπική εκτέλεση, συμβαίνουν γεγονότα μέσα στον επεξεργαστή τα οποία μπορούν να «προδώσουν» το γεγονός ότι η κερδοσκοπική εκτέλεση βρίσκεται σε λάθος μονοπάτι. Αν τα γεγονότα αυτά εντοπιστούν εγκαίρως, δηλαδή πριν να προλάβει να ολοκληρώσει την εκτέλεση του το εν λόγω Miss-Predicted Branch, τότε θα μπορούσαμε να σταματήσουμε την λανθασμένη κερδοσκοπική εκτέλεση σε εκείνο το σημείο, αποφεύγοντας να πληρώσουμε ολόκληρο το κόστος από το Miss-Prediction Penalty [3].

Ο τρόπος με τον οποίο σκεφτήκαμε να εντοπίσουμε τα γεγονότα λάθος μονοπατιού (Wrong Path Events) [3], είναι κατηγοριοποιώντας την εκτέλεση κάθε conditional branch κατά την ώρα του execution του, χρησιμοποιώντας 8 κριτήρια που αφορούν την κατάσταση που επικρατεί στο pipeline εκείνη τη χρονική στιγμή. Βάση αυτών των 8 κριτηρίων, το conditional branch λαμβάνει μια κλάση μεταξύ 0-255. Ακολούθως, παρατηρούμε για κάθε conditional branch ποιες είναι οι κλάσεις οι οποίες καταλήγουν να κάνουν Commit (Correct Path) και ποιες κάνουν μόνο Execute (Wrong Path). Βάση αυτής της διαφοροποίησης, ονομάζουμε Wrong Path Events τα γεγονότα που περιγράφουν τις κλάσεις των conditional branch που κάνουν μόνο Execute (Wrong Path).

Τα 8 κριτήρια με τα οποία αποφασίσαμε να κατηγοριοποιήσουμε τις εκτελέσεις των conditional branches, επιλέχθηκαν με βάση την ιδέα ότι κατά την κερδοσκοπική εκτέλεση στο λάθος μονοπάτι συμβαίνουν περίεργα γεγονότα που δεν συμβαίνουν συχνά στο σωστό μονοπάτι. Για το λόγω αυτό, τα 8 κριτήρια ελέγχουν για την ύπαρξή είτε νεότερων είτε παλαιότερων conditional branches και την κατάσταση τους.

1.3 Key Contributions & Findings

Κατά τη μελέτη της Διπλωματικής μου Εργασίας, κατάφερα να αναλύσω την συμπεριφορά των Benchmarks της σουίτας Spec2006 ως προς το κόστος στην επίδοση που προκαλεί η επιβάρυνση από τα Miss-Prediction Penalty. Η ανάλυση αυτή, ανέδειξε την σημαντικότητα του εγχειρήματος εύρεσης μηχανισμού που να βελτιστοποιεί το κόστος από τα Miss-Prediction Penalty.

Επιπρόσθετα, όρισα την συμπεριφορά ενός μηχανισμού ανάκαμψής από λάθος μονοπάτι στην κερδοσκοπική εκτέλεση, και πως αυτός ο μηχανισμός θε επιτύγχανε την αναγνώριση των Wrong Path Events που όρισα με την χρήση των 8 κριτήριων.

Ακολούθως, απέδειξα πως με την χρήση των 8 κριτηρίων για κατηγοριοποίηση των εκτελέσεων των conditional branches, επιτυγχάνω επιτυχώς να διαχωρίσω τα γεγονότα που συμβαίνουν πιο συχνά στο λάθος μονοπάτι, καταφέρνοντας έτσι να ορίσω Wrong Path Events για κάθε conditional branch ξεχωριστά.

Στην συνέχεια, για αξιολόγηση των αποτελεσμάτων, όρισα ένα Performance Model που υπολογίζει δυνητικά την ωφελιμότητα του μηχανισμού ανάκαμψής, βάση κάποιων παραμέτρων του εκάστοτε προγράμματος. Για το Opportunity Window, μια πολύ σημαντική παράμετρο του Performance Model, το οποίο περιγράφει το περιθώριο χρόνου που κερδίζεται από κάθε reversal, έκανα ανάλυση για κάθε ένα από τα Benchmarks της σουίτας Spec2006, αποδεικνύοντας πως ο συγκεκριμένος μηχανισμός επιτρέπει σημαντικά επίπεδα βελτίωσης στην επίδοση των προγραμμάτων.

Τέλος, χρησιμοποιώντας το Performance Model που προτείνα κατά τη διάρκεια αυτής της μελέτης, και αξιολόγησα σε βάθος τη βελτίωση του Benchmark astar, ενός Benchmark με πολλά MPKI (Miss-Predicts Per Kilo Instructions), αποδεικνιωντας ότι για τις καταλληλες παραμετρους του Performance Model μηχανισμός ανάκαμψής από Wrong Path στην κερδοσκοπική εκτέλεση χρησιμοποιώντας τα WPE που ορίσαμε, μπορεί να βελτιώσει σημαντικά την επίδοση του προγράμματος.

Κεφάλαιο 2

Υπόβαθρο

2.1 Pipeline

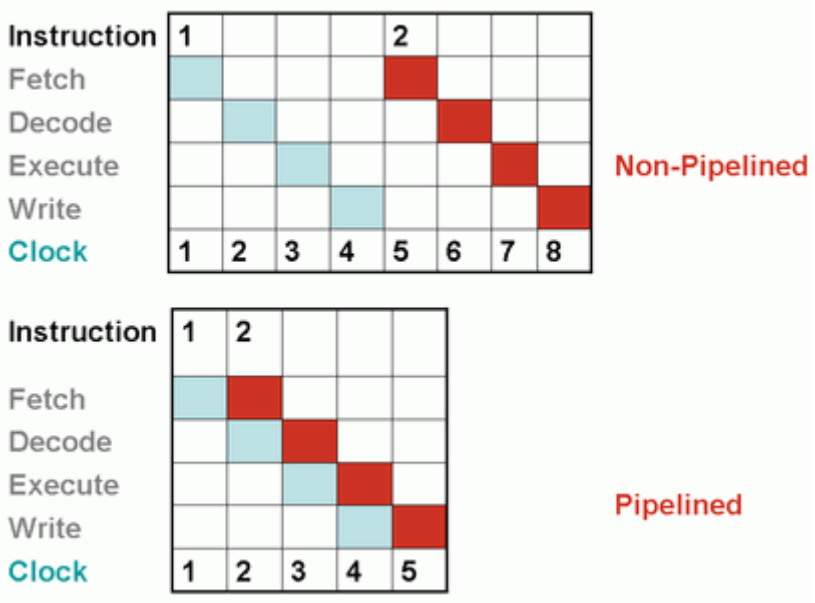
Στην επιστήμη των υπολογιστών, το pipelining είναι μια τεχνική για την υλοποίηση παραλληλισμού σε επίπεδο εντολής μέσα σε έναν ενιαίο επεξεργαστή. Κατά τη μέθοδο αυτή επιδιώκετε να κρατήσει κάθε τμήμα του επεξεργαστή απασχολημένο με κάποιο Instruction, διαιρώντας τα εισερχόμενα Instructions σε μια σειρά από διαδοχικά βήματα που εκτελούνται από διαφορετικά τμήματα του επεξεργαστή με αποτέλεσμα διαφορετικά μέρη πολλαπλών Instructions να επεξεργάζονται παράλληλα. Επιτρέπει την ταχύτερη διακίνηση Instructions στο CPU από ότι θα ήταν εφικτό σε ένα δεδομένο ρυθμό ρολογιού, αλλά μπορεί να αυξήσει την καθυστέρηση λόγω της προστιθέμενης επιβάρυνσης της ίδιας της διαδικασίας του pipelining (Pipeline hazards: data hazards, structural hazards, και control hazards).

Η βασική ιδέα είναι να χωρίζετε κάθε Instruction του επεξεργαστή σε μια σειρά από μικρά ανεξάρτητα στάδια. Κάθε στάδιο έχει σχεδιαστεί για να εκτελέσει ένα συγκεκριμένο μέρος του Instruction. Σε ένα πολύ βασικό επίπεδο, αυτά τα στάδια μπορούν να αναλυθούν στα εξής 5:

- **Instruction Fetch Unit:** Fetch an instruction from memory
- **Instruction Decode Unit:** Decode the instruction be executed
- **Execute Unit**: Execute the instruction
- **Memory Access Unit** Write the result back to memory (RAM)
- **Register Write Back Unit:** Write the result back to register

Clock cycle / Instr. No.	1	2	3	4	5	6	7
1	IF	ID	EX	MEM	WB		
2		IF	ID	EX	MEM	WB	
3			IF	ID	EX	MEM	WB
4				IF	ID	EX	MEM
5					IF	ID	EX

Σύγκριση ενός επεξεργαστή με 4 στάδια Pipeline σε σχέση με ένα επεξεργαστή χωρίς Pipeline (παρατηρούμε μεγαλύτερο throughput στην υλοποίηση με Pipeline):



2.2 Data and Control Hazards

Κατά την υλοποίηση ενός επεξεργαστή με την τεχνική του Pipeline, είναι εφικτό συγκεκριμένη σειρά εντολών να παράξει λανθασμένα αποτελέσματα. Τέτοιες περιπτώσεις ονομάζονται Pipeline Hazards και χωρίζονται στις εξής 3 κατηγορίες: data hazards, structural hazards, και control hazards. Πολλά από αυτά τα hazards μπορούν να συμβούν λόγω out-of-order execution.
Υπάρχουν διάφορες μέθοδοι για αντιμετώπιση αυτών των hazards έτσι ώστε να αποφευχθεί η παραγωγή λανθασμένου αποτελέσματος από τον επεξεργαστή,. Η πιο απλή μέθοδος είναι αυτή του stalling (ή pipeline bubling), στην οποία εισάγονται στο Pipeline κενές εντολές για να δημιουργήσουν κάποια καθυστέρηση μεταξύ των εντολών που δημιουργούν τα hazards , μέχρι ώστε να μην υπάρχει πλέον κίνδυνος.

Data Hazards

Data hazards μπορούν να συμβούν όταν εντολές τροποποιήσουν δεδομένα που χεριάζετε άλλη εντολή σε λανθασμένη χρονική σειρά. Τα Data hazards μπορούν να χωριστούν στις ακόλουθες 3 κατηγορίες:

- read after write (RAW)
  Συμβαίνει όταν μια εντολή αναφέρεται σε δεδομένα που ακόμα δεν έχουν υπολογιστεί ή αποθηκευτεί
  π.χ.

```
i1. R2 <- R1 + R3
i2. R4 <- R2 + R3
```

  Η εντολή i2 πρέπει να διαβάσει τα δεδομένα της αυστηρά αφότου ολοκληρωθεί η εκτέλεση της i1.

- write after read (WAR)
  Συμβαίνει συνήθως στη παράλληλη εκτέλεση 2 εντολών.
  π.χ.

```
i1. R4 <- R1 + R5
i2. R5 <- R1 + R2
```

  Η εντολή i2 πρέπει να γράψει τα δεδομένα της αυστηρά αφότου ολοκληρωθεί η εκτέλεση της i1.

- write after write (WAW)
  Συμβαίνει συνήθως στη παράλληλη εκτέλεση 2 εντολών.
  π.χ.

```
i1. R2 <- R4 + R7
i2. R2 <- R1 + R3
```

  Η εντολή i2 πρέπει να εκτελεστεί αυστηρά αφότου ολοκληρωθεί η εκτέλεση της i1.

Control Hazards

Control hazards συμβαίνουν κατά την είσοδο Branch εντολών στο Pipeline. Συγκεκριμένα το πρόβλημα προκύπτει όταν ο επεξεργαστής πρέπει να ορίσει το επόμενο PC και δεν ξέρει αν ένα Branch θα γίνει taken ή not-taken. Μια μέθοδος αντιμετώπισης αυτού του κίνδυνου, είναι η εισαγωγή stalls μέχρι η εντολή Branch να φτάσει το στάδιο μνήμης που θα αποκαλύπτει το νέο PC.

Structural Hazards

Structural hazards συμβαίνουν όταν κάποιο κομμάτι του επεξεργαστή το χρειάζονται δυο ή περισσότερες εντολές ταυτόχρονα. Για παράδειγμα αν ένας επεξεργαστής έχει μόνο μια πόρτα μνήμης και οι δυο εντολές που βρίσκονται στο Pipeline, η μια θέλει να διαβάσει γιατί βρίσκεται στο MEM Stage και η άλλη πρέπει να γίνει retrieve από την μνήμη γιατί βρίσκεται στο Fetch Stage, δεν μπορούν να έχουν πρόσβαση στην μνήμη ταυτόχρονα λόγω του περιορισμού στο hardware του επεξεργαστή.

2.3 Out of Order Execution

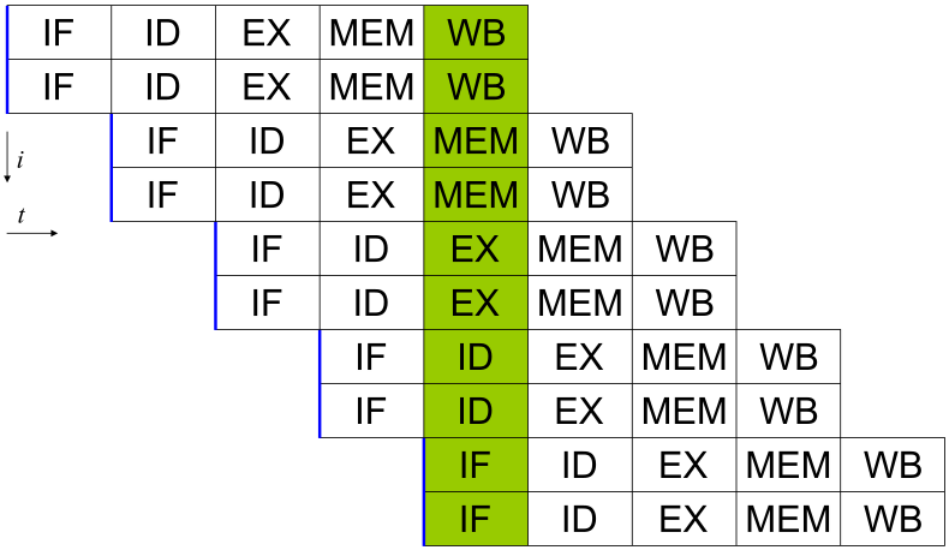
Out of Order Execution ονομάζετε η τεχνική στους συγχρόνους μικροεπεξεργαστές, η οποία αλλάζει την σειρά των εντολών του προγράμματος που εκτελούντες με σκοπό να αποφύγει την χρήση stalls, αλλά και ταυτόχρονα να πράξει εν τέλη αποτέλεσμα ίδιο με αυτό που θα είχε αν ακολουθούσε την κανονική σειρά των εντολών.

Η βασική ιδέα της επεξεργασίας Out of Order είναι να επιτρέψει στον επεξεργαστή να αποφύγει τα stalls που συμβαίνουν όταν τα δεδομένα που απαιτούνται για την εκτέλεση μιας εντολής δεν είναι διαθέσιμα.

Οι επεξεργαστές Out of Order συμπληρώνουν εγκαίρως αυτές τις "υποδοχές" με άλλες εντολές που είναι έτοιμες και στη συνέχεια αναδιατάσσουν τα αποτελέσματα στο τέλος για να φανεί ότι οι εντολές έγιναν με την κανονική σειρά. Η σειρά με την οποία εκτελούνται οι εντολές στον αρχικό κώδικα του υπολογιστή είναι γνωστή ως σειρά προγραμμάτων (program order).

2.4 Superscalar Processor

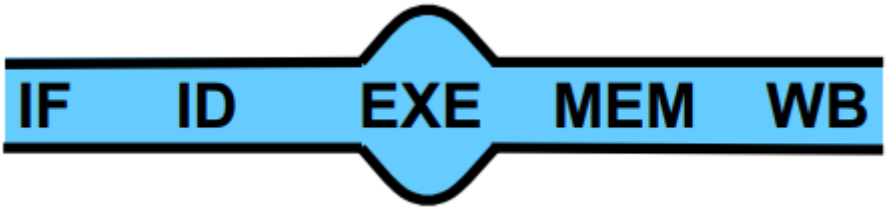
Ένας superscalar processor είναι ένα ειδος CPU που υλοποιεί μια μορφή παραλληλισμού που ονομάζεται παραλληλισμός επιπέδου εντολής (instruction-level parallelism) μέσα σε έναν ενιαίο επεξεργαστή. Σε αντίθεση με έναν Scalar επεξεργαστή ο οποίος μπορεί να εκτελεί το πολύ μία μόνο εντολή ανά κύκλο ρολογιού, ένας Superscalar επεξεργαστής μπορεί να εκτελέσει περισσότερες από μία εντολές κατά τη διάρκεια ενός κύκλου ρολογιού με ταυτόχρονη αποστολή πολλαπλών εντολών σε διαφορετικές μονάδες εκτέλεσης στον επεξεργαστή. Επομένως, επιτρέπει μεγαλύτερη απόδοση (τον αριθμό των εντολών που μπορούν να εκτελεστούν σε μια μονάδα χρόνου) από ότι διαφορετικά θα ήταν εφικτό σε ένα δεδομένο ρυθμό ρολογιού. Κάθε μονάδα εκτέλεσης δεν είναι ξεχωριστός επεξεργαστής (ή πυρήνας αν ο επεξεργαστής είναι επεξεργαστής πολλαπλών πυρήνων), αλλά ένας πόρος εκτέλεσης μέσα σε μία CPU, όπως μια αριθμητική λογική μονάδα.



Πιο πάνω παρατηρούμε ένα διάγραμμα απλού Superscalar επεξεργαστή. Κάθε ένα από τα 5 στάδια του Pipeline υπάρχει υλοποιημένο 2 φορές (διπλάσιο Hardware), με αποτέλεσμα να μπορούν να ολοκληρώνονται μέχρι και 2 εντολές σε κάθε κύκλο.

Σημείωση

Διπλασιάζοντας το Hardware μόνο σε ένα στάδιο του Pipeline δεν είναι αρκετό για να διπλασιαστεί το throughput εντολών στον επεξεργαστή. Αν για παράδειγμα διπλασιάσουμε την ALU (αριθμητική λογική μονάδα), θα έχουμε bottleneck στα υπόλοιπα στάδια του Pipeline.



Συμπέρασμα

Θα πρέπει να πολλαπλασιάζουμε το Hardware σε όλα τα στάδιο του Pipeline για να πολλαπλασιάσουμε το throughput εντολών στον επεξεργαστή (ιδανικά μιλώντας, εφόσον μπορεί να υπάρχουν εξαρτήσεις μεταξύ των εντολών που να μην επιτρέπουν τον παραλληλισμό)

2.5 Branch Predictor and Speculative Execution

Speculative execution είναι η ικανότητα του επεξεργαστή να εκτελεί εντολές που υπάρχουν πέρα από ένα conditional branch που δεν έχει ακόμη γίνει resolve, με απώτερο σκοπό να κάνει commit τα αποτελέσματα των εντολών αυτών στην σειρά της αρχικής ροής εντολών (program order). Το Speculative execution αποσκοπεί στην εκτέλεση εντολών πριν να είναι γνωστό αν είναι ασφαλές να το πράξει, δηλαδή αν είναι επικίνδυνο (hazards). Βασίζεται στην πρόβλεψη του Branch Predictor για να ακολουθήσει την κατάλληλη κατεύθυνση του unresolved Brach.

Το Speculative execution είναι μια από τις βασικές τεχνικές που χρησιμοποιείτε από τους πιο σύγχρονους επεξεργαστές υψηλής απόδοσης για τη βελτίωση της απόδοσης. Η ιδέα πίσω από το Speculative execution είναι ότι οι εντολές εκτελούνται πριν γνωρίσουν ότι απαιτούνται. Χωρίς Speculative execution, ο επεξεργαστής θα χρειαστεί να περιμένει τις προηγούμενες εντολές να γινουν resolve πριν εκτελέσουν τις επόμενες. Με την εκτέλεση των εντολών κερδοσκοπικά (speculatively), η απόδοση μπορεί να αυξηθεί ελαχιστοποιώντας την καθυστέρηση και εξάγοντας μεγαλύτερο παραλληλισμό. Τα αποτελέσματα μπορεί να απορριφθούν εάν διαπιστωθεί ότι οι εντολές δεν ήταν τελικά απαραίτητες.

Η πιο συνηθισμένη μορφή κερδοσκοπικής εκτέλεσης περιλαμβάνει τη ροή ελέγχου ενός προγράμματος. Αντί να περιμένει όλες τις εντολές διακλάδωσης να γινουν resolve για να καθορίσουν ποιες εντολές είναι απαραίτητες για την εκτέλεση, ο επεξεργαστής προβλέπει τη ροή ελέγχου χρησιμοποιώντας ένα εξαιρετικά εξελιγμένο σύνολο μηχανισμών (Branch Predictor). Συνήθως οι προβλέψεις είναι σωστές, πράγμα που επιτρέπει την επίτευξη υψηλών επιδόσεων με την απόκρυψη της καθυστέρησης των λειτουργιών που καθορίζουν τη ροή ελέγχου και την αύξηση του παραλληλισμού που μπορεί να εξαγάγει ο επεξεργαστής έχοντας ένα μεγαλύτερο σύνολο εντολών για ανάλυση. Ωστόσο, εάν μια πρόβλεψη είναι λάθος, τότε η εργασία που εκτελέστηκε κερδοσκοπικά απορρίπτεται και ο επεξεργαστής θα ανακατευθυνθεί για να εκτελέσει την σωστή διαδρομή εντολών.

Ενώ οι κερδοσκοπικές λειτουργίες δεν επηρεάζουν την αρχιτεκτονική κατάσταση του επεξεργαστή, μπορούν να επηρεάσουν την κατάσταση της μικροαρχιτεκτονικής, όπως οι πληροφορίες που αποθηκεύονται στα Translation Lookaside Buffers (TLB) και τις caches. Μέσω side chanel μεθόδων κάποιος θα μπορούσε να εκμεταλλευτεί την προσωρινή κατάσταση στη μικροαρχιτεκτονική που προκύπτει λόγω Speculative execution, και να αντλήσει δεδομένα παρακάμπτοντας ελέγχους ασφάλειας (Meltdown Bug και Spectre Bug).

Κεφάλαιο 3

Security Vulnerabilities

3.1 Speculative Execution Exploitation

Αρκετές τεχνικές μικροαρχιτεκτονικού σχεδιασμού έχουν διευκολύνει την αύξηση της ταχύτητας του επεξεργαστή τις τελευταίες δεκαετίες. Μία τέτοια πρόοδος είναι η κερδοσκοπική εκτέλεση (speculative execution), η οποία χρησιμοποιείται ευρέως για την αύξηση της απόδοσης και συνεπάγεται ότι η CPU υποθέτει πιθανές μελλοντικές διαδρομές (paths) εκτέλεσης και εκτελεί πρόωρα εντολές σχετικά με αυτές τις διαδρομές.

Για παράδειγμα, όταν η ροή ελέγχου του προγράμματος εξαρτάται από μια τιμή που δεν βρίσκεται σε κάποια cache αλλά σε εξωτερική φυσική μνήμη. Καθώς αυτή η μνήμη είναι πολύ πιο αργή από την CPU, απαιτούνται συχνά αρκετές εκατοντάδες κύκλοι ρολογιού για να γίνει γνωστή η τιμή. Αντί να σπαταλάει αυτούς τους κύκλους περιμένοντας, η CPU προσπαθεί να μαντέψει την κατεύθυνση της ροής ελέγχου, αποθηκεύει ένα checkpoint της κατάστασης και προχωρά στην κερδοσκοπική εκτέλεση του προγράμματος στην εικασμένη διαδρομή. Όταν η τιμή φθάσει τελικά από τη μνήμη, η CPU ελέγχει την ορθότητα της αρχικής της εικασίας. Εάν η εικασία ήταν λανθασμένη, η CPU απορρίπτει την εσφαλμένη εκτέλεση κερδοσκοπίας επαναφέροντας την κατάσταση του μητρώου στο αποθηκευμένο checkpoint, με αποτέλεσμα η απόδοση να φτάνει στο επίπεδο που θα ήταν αν περίμενε εξαρχής την τιμή να φθάσει από την εξωτερική μνήμη στη CPU. Ωστόσο, αν η εικασία ήταν σωστή,

τα κερδοσκοπικά αποτελέσματα εκτέλεσης αποθηκεύονται, αποδίδοντας ένα σημαντικό κέρδος απόδοσης, καθώς η χρήσιμη δουλειά ολοκληρώθηκε κατά τη διάρκεια της καθυστέρησης που θα υπήρχε περιμένοντας την τιμή να φθάσει από την εξωτερική μνήμη στη CPU.

Από την άποψη της ασφάλειας, η κερδοσκοπική εκτέλεση συνεπάγεται την εκτέλεση ενός προγράμματος με πιθανούς λανθασμένους τρόπους. Ωστόσο, επειδή οι επεξεργαστές έχουν σχεδιαστεί για να διατηρούν τη λειτουργική ορθότητα επαναφέροντας τα αποτελέσματα των λανθασμένων κερδοσκοπικών εκτελέσεων στις προηγούμενες καταστάσεις τους, τα λάθη αυτά θεωρήθηκαν προηγουμένως ασφαλή.

Πλέον, από άποψη ασφάλειας η κερδοσκοπική εκτέλεση δεν θεωρείται απόλυτα ασφαλής. Αυτό συμβαίνει γιατί διάφορα ήδη επιθέσεων έχουν ανακαλυφτεί (Meltdown [4], Spectre [5]) τα οποία εκμεταλλεύονται την κερδοσκοπική εκτέλεση και το γεγονός ότι δεν επαναφέρει τα αποτελέσματα της σε επίπεδο μικροαρχιτεκτονικής (π.χ. caches) με αποτέλεσμα να μπορούν να διαβάζουν μνήμη που δεν θα έπρεπε να μπορούν να διαβάσουν.

3.2 Meltdown Attack

Η ασφάλεια των συστημάτων υπολογιστών στηρίζεται βασικά στην απομόνωση της μνήμης, π.χ. kernel address ranges χαρακτηρίζονται ως μη προσπελάσιμα και προστατεύονται από user access. Το Meltdown [4] εκμεταλλεύεται τις παρενέργειες της εκτέλεσης out-of-order σε σύγχρονους επεξεργαστές για να διαβάσει αυθαίρετες τοποθεσίες μνήμης του kernel, συμπεριλαμβανομένων προσωπικών δεδομένων και κωδικών πρόσβασης. Η εκτέλεση out-of-order είναι ένα απαραίτητο χαρακτηριστικό απόδοσης και παρουσιάζεται σε ένα ευρύ φάσμα σύγχρονων επεξεργαστών. Η επίθεση είναι ανεξάρτητη από το λειτουργικό σύστημα και δεν βασίζεται σε ευπάθειες λογισμικού. Το Meltdown σπάει όλες τις εγγυήσεις ασφάλειας που παρέχονται από την απομόνωση του χώρου διευθύνσεων καθώς και από τα paravirtualized περιβάλλοντα και, συνεπώς, από κάθε μηχανισμό ασφαλείας που βασίζεται σε αυτό το θεμέλιο.
Στα επηρεαζόμενα συστήματα, το Meltdown δίνει τη δυνατότητα στον attacker να διαβάσει τη μνήμη άλλων διαδικασιών (processes) ή virtual machines στο cloud χωρίς να έχει κάποια δικαιώματα, επηρεάζοντας ουσιαστικά όλους τους χρήστες προσωπικού υπολογιστή.

Ένα κεντρικό χαρακτηριστικό ασφαλείας των σημερινών λειτουργικών συστημάτων είναι η απομόνωση μνήμης (memory isolation). Τα λειτουργικά συστήματα εξασφαλίζουν ότι τα προγράμματα των χρηστών δεν μπορούν να έχουν πρόσβαση στη μνήμη άλλων προγραμμάτων ή στο kernel memory. Αυτή η απομόνωση αποτελεί τον ακρογωνιαίο λίθο των υπολογιστικών μας περιβαλλόντων και επιτρέπει την εκτέλεση πολλαπλών εφαρμογών ταυτόχρονα σε προσωπικές συσκευές ή την εκτέλεση διαδικασιών πολλαπλών χρηστών σε ένα μόνο μηχάνημα στο cloud.

Από άποψη ασφάλειας, οι out-of-order CPUs επιτρέπουν σε ένα unprivileged process να φορτώσει δεδομένα από μια privileged διεύθυνση σε ένα προσωρινό καταχωρητή. Επιπλέον, η CPU εκτελεί ακόμη και άλλους υπολογισμούς βάσει αυτής της τιμής του καταχωρητή, π.χ. πρόσβαση σε ένα array με βάση την τιμή του καταχωρητή. Με απλή απόρριψη των αποτελεσμάτων των ενεργειών αυτών αν αποδειχθεί ότι δεν θα έπρεπε να εκτελεστεί μια εντολή, ο επεξεργαστής εξασφαλίζει την ορθή εκτέλεση του προγράμματος. Ως εκ τούτου, στο αρχιτεκτονικό επίπεδο (π.χ., ο αφηρημένος ορισμός

του τρόπου με τον οποίο ο επεξεργαστής πρέπει να εκτελεί υπολογισμούς) δεν προκύπτει κανένα πρόβλημα ασφαλείας. Αυτό όμως δεν συμβαίνει στο επίπεδο μικροαρχιτεκτονικής (π.χ. caches).

Παρατηρήσαμε ότι τα out-of-order memory lookups επηρεάζουν την cache, η οποία με τη σειρά της μπορεί να ανιχνευθεί μέσω side channel attack. Ως αποτέλεσμα, ένας attacker μπορεί να διαβασει ολόκληρο το kernel memory διαβάζοντας την προνομιακή μνήμη σε μια ροή εκτέλεσης out-of-order και να μεταδώσει τα δεδομένα από αυτή την αόριστη κατάσταση μέσω ενός μικροαρχιτεκτονικού side channel (π.χ. Flush + Reload) στον έξω κόσμο. Ως εκ τούτου, σε επίπεδο μικροαρχιτεκτονικής (π.χ. την πραγματική υλοποίηση υλικού), υπάρχει ένα εκμεταλλεύσιμο πρόβλημα ασφαλείας.

3.3 Spectre Attack

Οι σύγχρονοι επεξεργαστές χρησιμοποιούν branch prediction και speculative execution για μεγιστοποίηση της απόδοσης. Για παράδειγμα, εάν ο προορισμός ενός branch εξαρτάται από μια τιμή μνήμης που βρίσκεται σε διαδικασία ανάγνωσης, οι επεξεργαστές θα προσπαθήσουν να μαντέψουν τον προορισμό και να επιχειρήσουν να εκτελέσουν πρόωρα. Όταν η τιμή της μνήμης τελικά φθάσει, η CPU απορρίπτει ή δεσμεύει τον κερδοσκοπικό υπολογισμό. Το speculative execution μπορεί να έχει πρόσβαση στη μνήμη και τους καταχωρητές του θύματος και μπορεί να εκτελεί πράξεις με σημαντικές παρενέργειες. Αν η πρόβλεψη ήταν λανθασμένη και οι ενέργειες που εκτελέστηκαν παραβίασαν την απομόνωση μνήμης (memory isolation), τότε η CPU θα απορρίψει την αποθήκευση τους προσπαθώντας να επιστρέψει σε συνεπές κατάσταση σε αρχιτεκτονικό επίπεδο.

Το Spectre [5] είναι μια ευπάθεια που επηρεάζει τους σύγχρονους μικροεπεξεργαστές που εκτελούν branch prediction. Στους περισσότερους επεξεργαστές, η κερδοσκοπική εκτέλεση που προκύπτει από ενα εσφαλμένο branch prediction (misprediction) μπορεί να αφήνει παρατηρήσιμες παρενέργειες που μπορεί να αποκαλύψουν ιδιωτικά δεδομένα σε εισβολείς. Για παράδειγμα, εάν η κατασταση της μνήμης cache μετα από μια τέτοια κερδοσκοπική εκτέλεση εξαρτάται από ιδιωτικά δεδομένα, η προκύπτουσα κατάσταση της μνήμης cache αποτελεί ένα side channel μέσω του οποίου ένας εισβολέας μπορεί να αποκτήσει πληροφορίες σχετικά με τα ιδιωτικά δεδομένα χρησιμοποιώντας μια επίθεση χρονισμού (timing attack). Αυτό γίνετε εφικτό γιατί πολλές φορές η κρυφή μνήμη cache ενός επεξεργαστή μπορεί να μην επηρεαστεί από την διαδικασία αναίρεσης των ενεργειών ενός εσφαλμένου branch prediction (misprediction) αφήνοντας παρατηρήσιμες παρενέργειες.

Side Channel Cache Methods

Μια side channel μέθοδος λειτουργεί με την απόκτηση πληροφοριών μέσω της παρατήρησης του συστήματος, όπως με τη μέτρηση των μικροαρχιτεκτονικών ιδιοτήτων του συστήματος (π.χ. την πραγματική υλοποίηση υλικού όπως caches). Σε αντίθεση με τις υπερχείλισης buffer και άλλες κατηγορίες ευπάθειας, οι side channel ευπάθειες δεν επηρεάζουν άμεσα την εκτέλεση του προγράμματος ούτε επιτρέπουν την τροποποίηση ή τη διαγραφή δεδομένων.

Ένα cache timing side channel περιλαμβάνει έναν πράκτορα που ανιχνεύει εάν υπάρχει ένα κομμάτι δεδομένων σε ένα συγκεκριμένο επίπεδο των cache του επεξεργαστή, όπου η παρουσία του μπορεί να χρησιμοποιηθεί για να συναχθεί κάποια άλλη πληροφορία. Μία μέθοδος για την ανίχνευση της παρουσίας των εν λόγω δεδομένων είναι η χρήση timers για τη μέτρηση της καθυστέρησης πρόσβασης στη μνήμη στη διεύθυνση. Εάν η πρόσβαση στη μνήμη διαρκεί σύντομα, τότε τα δεδομένα πρέπει να υπάρχουν σε μια κοντινή μνήμη cache. Εάν η πρόσβαση διαρκεί περισσότερο, τότε τα δεδομένα ενδέχεται να μην βρίσκονται στη κοντινή μνήμη cache.

3.4 Σύνοψη

Οι σύγχρονοι επεξεργαστές χρησιμοποιούν out-of-order execution, branch prediction και speculative execution για μεγιστοποίηση της απόδοσης τους. Αυτό δημιουργεί κατά συνέπεια κάποιες παρενέργειες, όπως η ύπαρξη δεδομένων στην cache από περιοχές μνήμης που διαβάστηκαν λανθασμένα. Ως αποτέλεσμα, δημιουργούνται στον τομέα της ασφάλειας ευπάθειές που επιτρέπουν την κακοήθη εκμετάλλευση αυτών των παρενεργειών. Πιο γνωστά παραδείγματα ευπαθειών που εκμεταλλεύονται αυτές τις λειτουργίες είναι αυτά του Meltdown και Spectre που περιεγράφηκαν πιο πάνω.

Παρατηρούμε όμως, πως οι συγκεκριμένες ευπάθειές λαμβάνουν χώρο κατά την κερδοσκοπική εκτέλεση που προκύπτει από ένα εσφαλμένο branch prediction (misprediction) . Αν λοιπόν, με κάποιο τρόπο ο συνολικός χρόνος που ξόδευε ένας επεξεργαστής στην κερδοσκοπική εκτέλεση εσφαλμένου branch prediction μειωνόταν, τότε θα μειωνόταν και η δυνατότητα ενός attacker να χρησιμοποίηση τα bugs αυτά (Meltdown και Spectre) για να διαβάσει δεδομένα που δεν είναι εξουσιοδοτημένος.
Βλέπουμε λοιπόν ότι όσων αφορά την ασφάλεια ενός επεξεργαστή, ένας τρόπος μείωσης των κίνδυνων που προκύπτουν από αυτήν την κατηγορία επιθέσεων είναι η μείωση του χρόνου που εκτελείτε κερδοσκοπικά ένα εσφαλμένο branch prediction. Ο σκοπός της διπλωματικής μου εργασίας αφορά ακριβώς τον εντοπισμό γεγονότων που υποδηλώνουν ένα εσφαλμένο branch prediction κατά την κερδοσκοπική εκτέλεση, με σκοπό να επιδιορθώνεται η κατεύθυνση του εν λόγω branch (reversal) κατά την κερδοσκοπική εκτέλεση όσο το δυνατό νωρίτερα. Αυτό κατά συνέπεια θα μείωνε τον χρόνο που θα περνούσε συνολικά ένας επεξεργαστής στη κερδοσκοπική εκτέλεση εσφαλμένων branch predictions και έτσι θα ανέβαζε το επίπεδο ανθεκτικότητας στα bugs Meltdown και Spectre, μειώνοντας το παράθυρο χρόνου όπου θα υπήρχε η ευπάθεια.

Κεφάλαιο 4

Speculative Branch Reversal Mechanism

4.1 High Level Description of the Reversal Mechanism

Το branch prediction και speculative execution χρησιμοποιούνται ευρέως για τη βελτίωση της απόδοσης του επεξεργαστή. Η σωστή κερδοσκοπική εκτέλεση μπορεί να μειώσει τον χρόνο εκτέλεσης, αλλά η εσφαλμένη κερδοσκοπική εκτέλεση μπορεί να οδηγήσει σε αυξημένο χρόνο εκτέλεσης και μεγαλύτερη κατανάλωση ενέργειας.

Τα Wrong Path Events (WPE) αποτελούν γεγονότα που συμβαίνουν μέσα στον επεξεργαστή κατά τις περιόδους εσφαλμένης κερδοσκοπικής εκτέλεσης. Ένα Wrong Path Event είναι μια περίπτωση παράνομης ή ασυνήθιστης συμπεριφοράς του προγράμματος που είναι πιο πιθανό να εμφανιστεί σε λάθος διαδρομή από ότι στη σωστή διαδρομή. Εντοπίζοντας τα Wrong Path Events, ο επεξεργαστής μπορεί να προβλέψει ότι βρίσκεται σε λάθος μονοπάτι εκτέλεσης κατά το speculative execution και να ανάκαμψη νωρίτερα (reversal), κερδίζοντας έτσι τους εναπομείναντες κύκλους από την χρονική στιγμή που θα έκανε resolve το branch το οποίο προβλέπτικε λανθασμένα από τον branch predictor.

Ο σκοπός του προτεινόμενου μηχανισμού είναι να βελτιώσει την αποτελεσματικότητα της κερδοσκοπικής εκτέλεσης σε έναν επεξεργαστή βοηθώντας να διασφαλίσει ότι ο επεξεργαστής παραμένει "στη σωστή πορεία" σε όλες τις περιόδους κερδοσκοπικής εκτέλεσης.

4.2 Functionalities & Timing

Για την υλοποίηση του μηχανισμού ανάκαμψής από Wrong Path στην κερδοσκοπική εκτέλεση χρησιμοποιώντας τα WPE που ορίσαμε, ένας σύγχρονος επεξεργαστής με Pipeline και Speculative Execution θα πρέπει να υποστηρίζει τις ακόλουθες λειτουργίες:

- Στο Execution στάδιο του Pipeline, θα πρέπει για κάθε εντολή conditional branch που εκτελείται να γίνεται έλεγχος για την ύπαρξη των 8 χαρακτηριστικών που μελετούμε για εύρεση WPE. Ακολούθως, η τιμή που προκύπτει από το 8-bit vector θα πρέπει να προσκολλάτε στην εντολή για να την ακολουθήσει στα επόμενα στάδια του Pipeline. Επίσης, σε αυτό το στάδιο θα πρέπει να ενημερώνεται η δομή που κρατά τη συχνότητα εμφανίσεων των κλάσεων για αυτήν την συγκεκριμένη εντολή, αυξάνοντας τον μετρητή που αφορά τα **Executes** αυτής της εντολής με την κλάση που προέκυψε.

- Στο Commit στάδιο του Pipeline, εφοσον κάθε conditional branch περασε από το σταδιο του Execution και εχει καποια κλαση βαση των χαρακτηριστικων της την ωρα του Execution, θα πρέπει να ενημερώνεται η δομή που κρατά τη συχνότητα εμφανίσεων των κλάσεων για αυτήν την συγκεκριμένη εντολή, αυξάνοντας τον μετρητή που αφορά τα **Commits** αυτής της εντολής με την κλάση που προέκυψε.

- Με το πέρας κάποιου αριθμού κύκλων μηχανής που θα είναι αρκετή για να ξεχωρίσουν οι κλάσεις των conditional branch εντολών που καταλήγουν σε Commit (Correct Path) και αυτές που δεν καταλήγουν (Wrong Path), θα αρχίσουν να γίνονται τα reversals. Δηλαδή στο στάδιο του Execution, αφού λάβει μια conditional branch εντολή την κλάση της, θα γίνεται έλεγχος στη δομή που κρατά τη συχνότητα εμφανίσεων των κλάσεων για την συγκεκριμένη εντολή, και αν η κλάση αυτή έχει πολύ μεγαλύτερη συχνότητα στα Executes σε σχέση με τα Commits, τότε θα εγείρει reversal. Δηλαδή, θα προκαλεί Flush των νεότερων εντολών και το conditional branch θα αλλάζει κατεύθυνση στην κερδοσκοπική εκτέλεση (δηλαδή θα γίνεται το ανάποδο από αυτό που προέβλεψε ο branch predictor όταν έγινε fetch η εντολή).

Opportunity Window

Opportunity Window ορίζουμε το περιθώριο χρόνου που προκύπτει από την εύρεση ενός Wrong Path Event μέχρι την χρονική στιγμή που θα έκανε resolve το conditional branch που μελετούμε. Δηλαδή, μας δίνει τον αριθμό των κύκλων που δυνητικά θα επωφεληθούμε αν το conditional branch βρίσκεται σε Wrong Path και εγείρουμε ανάκαμψή (reversal) την χρονική στιγμή που εντοπίσουμε το WPE.

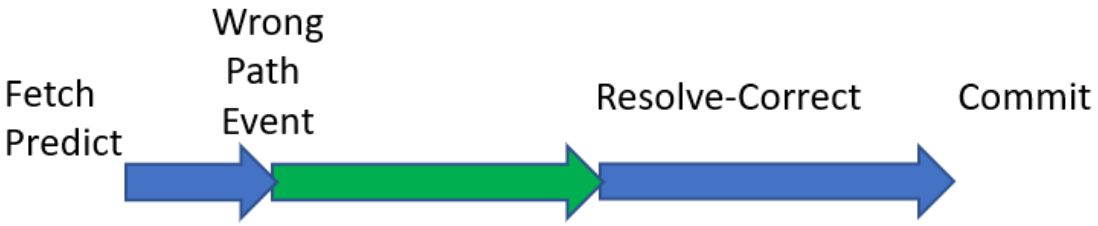
Συγκεκριμένα, ο τρόπος με τον οποίο υπολογίζουμε το στατιστικό του Opportunity Window για θεωρητική μελέτη του μηχανισμού ανάκαμψής, είναι βρίσκοντας το μέγιστο αριθμό κύκλων από τη διαφορά μεταξύ μιας εντολής που κάνει Resolve και όλων των νεότερων εντολών σε program order που έχουν κάνει Resolve και βρίσκονται στο Reorder Buffer. Δηλαδή είναι η μέγιστη διαφορά σε κύκλους από τον κύκλο που έγινε Resolve η εντολή που εξετάζουμε σε σχέση με όλες τις νεότερες εντολές που έκαναν Resolve νωρίτερα. (Οι νεότερες εντολές θα πρέπει να είναι είτε F_CTRL εντολές, είτε να αποτελούν Replay Traps)

Το στατιστικό του Opportunity Window είναι ιδιαίτερα χρήσιμο, εφόσον χρησιμοποιείται στο Performance Model για να υπολογιστεί το περιθώριο βελτίωσης από τον μηχανισμό ανάκαμψής που προτείνουμε σε αυτή την μελέτη.

Miss-Predicted Condition Branch without the Reversal Mechanism

Με κόκκινο απεικονίζεται το **Miss-Prediction Penalty**.

Miss-Predicted Condition Branch with the Reversal Mechanism



Με πράσινο απεικονίζεται το **Opportunity Window**.

4.3 Performance Model

Το Performance Model έχει σκοπό να αναδείξει την θεωρητική επίδραση που θα έχει στην επίδοση ενός προγράμματος η εφαρμογή του μηχανισμού ανάκαμψής των Wrong Path Events, βάση κάποιων παραμέτρων, όπως το Opportunity Window και Miss-Predict Penalty.

V= A – C + P

A= CPI * Commits
C= CorrectReversals * (OppurtunityWindow)
P= WrongReversals * (avg_br_correct_latency_resolution)

A: Total Cycles
C: Cycles saved from Correct Reversals
P: Cycles overhead from Wrong Reversals (Miss-Predict Penalty)
V: Total Cycles with the mechanism of WPE reversals

Το Performance Model είναι αρκετά απλό. Επιστρέφει τον αναμενόμενο αριθμό κύκλων που θα έχει ένα πρόγραμμα μετά την εφαρμογή του μηχανισμού ανάκαμψής των Wrong Path Events. Για τον υπολογισμό αυτό λαμβάνει τις ακόλουθες παραμέτρους:

- CPI: Clocks Per Instruction
- Commits: Total number of Instruction Committed
- CorrectReversals: Total number of Miss-predicts reversed correctly
- WrongReversals: Total number of Correct-predicts reversed incorrectly
- avg_br_correct_latency_resolution: Average number of cycles that Correct-predicts need to resolve (from fetch to resolution)

Για να υπολογίσει κάποιος την βελτίωση που προφέρει ο μηχανισμός ανάκαμψής των Wrong Path Events σε σχέση με την απουσία, σε ποσοστό, αρκεί να υπολογίσει το πιο κάτω τύπο:

Improvement_Rate= A / V

A: Total Cycles
V: Total Cycles with the mechanism of WPE reversals

4.4 Key Differences from related work that proposed WPE

Μια ομάδα ερευνητών από το University of Texas, πριν κάποια χρόνια, εισηγήθηκε την ιδέα των Wrong Path Events και την χρήση τους για βελτίωση της επίδοσης του επεξεργαστή. Η ομάδα αυτή επέλεξε να εστιάσει την έρευνα της στον εντοπισμό WPE ανά κατηγορία εντολών.

Memory Instructions: Έθεσαν ως Wrong Path Event το dereference ενός NULL Pointer, γιατί όπως παρατήρησε η ομάδα των ερευνητών, αυτό δεν συμβαίνει ποτέ ότι υπό κανονικές συνθήκες (Correct Path) και έτσι σηματοδοτεί Wrong Path κατά την κερδοσκοπική εκτέλεση.

Control Flow Instructions: Έθεσαν για ένα conditional branch ως Wrong Path Event την ύπαρξη τριών νεότερων conditional branches στο Reorder Buffer τα οποία έγιναν resolve ως Miss-Predicts. Το γεγονός αυτό δικαιολογείται από την παρατήρηση πως κατά το λάθος μονοπάτι η συμπεριφορά των conditional branches είναι περίεργη και έτσι ο branch predictor κάνει πολλά διαδοχικά Miss-Predicts.

Arithmetic Instructions: Έθεσαν ως Wrong Path Event τις αριθμητικές εντολές που εκτελούσαν αδύνατες μαθηματικές πράξεις, όπως η διαίρεση με το μηδέν ή εύρεση τετραγωνικής ρίζας αρνητικού αριθμού.

Όπως παρατηρούμε, η ομάδα αυτή των ερευνητών εστίασε την προσοχή της σε συγκεκριμένα γεγονότα για εύρεση Wrong Path Events και η προσέγγιση τους διαφέρει αρκετά από την δική μας. Στην δική μας μελέτη χρησιμοποιήσαμε ένα set από 8 χαρακτηριστικά για κατηγοριοποίηση των εκτελέσεων των conditional branches και αφήσαμε την εύρεση των Wrong Path Events να καθοριστεί από μονή της κατά την εκτέλεση των προγραμμάτων, με την διαφοροποίηση που θα πρόκυπτε μεταξύ της συχνότητας των εμφανιζομένων κλάσεων στα Commits και τα Executes κάθε εντολής.

Κεφάλαιο 5

Wrong Path Events

5.1 Προσέγγιση της Διπλωματικής Εργασίας

Το ζητούμενο που προκύπτει για την εφαρμογή του μηχανισμού ανάκαμψής (reversal) από λάθος διαδρομή στη κερδοσκοπική εκτέλεση , είναι η εύρεση των Wrong Path Events (WPE). Για να γίνει αυτό επιτυχώς, θα πρέπει πρώτα να οριστούν τα γεγονότα που υποδεικνύουν το λάθος μονοπάτι και ακολούθως να αξιολογηθεί η αποτελεσματικότητας τους.

Επιτυχημένο reversal (correct reversal) ονομάζουμε την αλλαγή της διαδρομής ενός miss-predicted branch στην διαδρομή όπου είναι ορθό να προχωρήσει η εκτέλεση. Αντίστοιχος, αποτυχημένο reversal (incorrect reversal) () ονομάζουμε την αλλαγή της διαδρομή ενός correctly-predicted branch στην λανθασμένη διαδρομή εκτέλεσης. Είναι σημαντικό να αναφέρουμε ότι στην περίπτωση των correct reversals είναι πολύ σημαντικός ο χρόνος απόστασής από το reversal μέχρι την εκτέλεση (resolve) του εν λόγω branch, εφόσον αυτή η διαφορά, σε κύκλους μηχανής, θα είναι το κέρδος στην επίδοση του προγράμματος από το συγκεκριμένο reversal. Επίσης, από τα incorrect reversal, η λανθασμένη αλλαγή της διαδρομή του εν λόγω branch στη κερδοσκοπική εκτέλεση, θα επιφέρει επιπρόσθετο κόστος στο συνολικό χρόνο εκτέλεσης του προγράμματος. Ο υπολογισμός αυτών των παραμέτρων που ορίζουν την ωφελιμότητα του μηχανισμού ανάκαμψής (reversal) από λάθος διαδρομή στη κερδοσκοπική εκτέλεση, ορίζεται στο Performance Model της διπλωματικής εργασίας.

Η αποτελεσματικότητα των γεγονότων που ορίζουμε ως Wrong Path Events (WPE), κρίνεται από το πλήθος των εντολών που παρουσιάζουν αυτά τα γεγονότα κατά την κερδοσκοπική τους εκτέλεση, αλλά και από το ποσοστό correct reversal έναντί των incorrect.

5.2 Επιλογή των Wrong Path Events

Για την επιλογή των Wrong Path Events, ορίσαμε 8 κριτήρια με τα οποία κατηγοριοποιούμε την εκτέλεση κάθε conditional branch εντολής σε 256 κλάσεις. Ακολούθως, για κάθε μοναδικό PC (Program Counter) μετρούμε τον αριθμό των φορών που έγινε Commit και τον αριθμό των φορών που έγινε Resolve σε κάθε κλάση (δηλαδή για κάθε conditional branch PC κρατάμε 512 μετρητές που αφορούν τους μετρητές των Commits στις 256 κλάσεις και τους μετρητές των Resolves στις 256 κλάσεις).

Αν ο αριθμός των Commits και Resolves για ένα PC σε μια κλάση έχουν μεγάλη διαφορά (ratio), τότε τα γεγονότα τα οποία περιγράφουν την κλάση αυτού του PC, αποτελούν Wrong Path Events για το συγκεκριμένο PC.

Με βάση την μεθοδολογία αυτή, ταξινομούμε τα ratio των μετρητών των 256 κλάσεων για όλα τα conditional branch PCs, και ορίζουμε ως Wrong Path Events για κάθε PC ξεχωριστά, τα γεγονότα τα οποία περιγράφουν τις κλάσεις με μεγάλο ratio.

Τα 8 κριτήρια με τα οποία κατηγοριοποιούμε τις εκτελέσεις των conditional branches αποτελούν Boolean τιμές που λαμβάνουν τιμή 1 αν το conditional branch ικανοποιεί την συνθήκη τους κατά την ώρα του Resolve, αλλιώς λαμβάνουν τιμή 0. Στη συνέχεια αυτό το 8-bit vector μετατρέπεται σε μια δεκαδική τιμή μεταξύ 0-256, που αποτελεί την κλάση την συγκεκριμένης εκτέλεσης του conditional branch. Για κάθε Resolve conditional branch αυξάνεται ο μετρητής «Resolves» του conditional branch της συγκεκριμένης κλάσης. Αν αυτό το conditional branch φτάσει σε Commit, τότε αυξάνεται και ο μετρητής «Commits» του conditional branch της συγκεκριμένης κλάσης (που έλαβε κατά την ώρα του Resolve).

The 8 Features

Τα 8 κριτήρια με τα οποία ορίζετε το bit vector για την εύρεση της κλάσης μιας εκτέλεσης ενός conditional branch είναι τα ακόλουθα:

- Older MissPredicted F_CTRL Instructions
- Older CorrectPredicted F_CTRL Instructions
- Older Unresolved F_CTRL Instructions
- Younger MissPredicted F_CTRL Instructions
- Younger CorrectPredicted F_CTRL Instructions
- Younger Unresolved F_CTRL Instructions
- Younger Replay TRAP Instructions
- Is Misspredicted

F_CTRL Instructions ορίζονται οι εντολές που μπορούν να αλλάξουν τη ροή ελέγχου (Flow Control) του προγράμματος.

F_CTRL Instructions: BR, FBEQ, FBLT, FBLE, BSR, FBNE, FBGE, FBGT, BLBC, BEQ, BLT, BLE, BLBS, BNE, BGE, BGT, JMP, JSR, RETN, JSR_COROUTINE

Η επιλογή των συγκεκριμένων κριτήριων έγινε χρησιμοποιώντας κυρίως 2 ευρετικά (heuristic). Το πρώτο αφορά το γεγονός ότι ένα Wrong Path Event ακολουθείται συνήθως από άλλα Wrong Path Events, και έτσι ένα miss-predict ή ένα replay trap στο Reorder Buffer μπορεί να υποδηλώνει ότι βρισκόμαστε σε λάθος διαδρομή. Το άλλο ευρετικό, αφορά το γεγονός πως η χρονική σειρά με την οποία εκτελούνται τα branches στο Reorder Buffer πιθανόν να προδίδει ότι βρισκόμαστε σε λάθος διαδρομή.

Κεφάλαιο 6

Πειραματική Αξιολόγηση

6.1 Προσομοιωτής Sim-Alpha

Ο προσομοιωτής Sim-Alpha είναι ένας execution-driven simulator που μοντελοποιεί τους περιορισμούς υλοποίησης και τα low-level χαρακτηριστικά επίδοσης του Alpha 21264, ενός RISC μικροεπεξεργαστή με Alpha Instruction Set Architecture. Ο προσομοιωτής αυτός, παράγει συγκρίσιμα αποτελέσματα με πραγματικό hardware και αυτό επιτυγχάνετε χάρης στην ακρίβεια με την οποία μοντελοποιεί τον Alpha μικροεπεξεργαστή. Ο Sim-Alpha εκτελεί τις εντολές στο βάθος των miss-predicts paths όπως θα τις εκτελούσε και ένας πραγματικός επεξεργαστής. Χρησιμοποιείτε κατά κόρων από ερευνητές για την ακρίβεια του και για το λόγω αυτό θα ήταν κατάλληλος για την έρευνα αυτής της διπλωματικής.

Για ακρίβεια των αποτελεσμάτων της έρευνα μας, είναι πολύ σημαντικό ο branch predictor που χρησιμοποιείτε από τον προσομοιωτή Sim-Alpha να είναι ο καλύτερος δυνατός. Για το λόγω αυτό χρησιμοποίησα μια αναβαθμισμένη έκδοση του Sim-Alpha, η οποία ενσωματώνει τον branch predictor L-TAGE, ο οποίος αποτελεί ένα state-of -the-art-predictor που προτάθηκε από τον André Seznec και κέρδισε το διαγωνισμό CBP 2016. Επιπροσθετα η αναβαθμισμένη έκδοση του Sim-Alpha υποστηρίζει και out-of-order Brach Execution.

6.2 Spec2006 Benchmark Suite

Τα Benchmark που περιλαμβάνονται στο πακέτο των Spec2006 αποτελούν βιομηχανική τυποποιημένη σουίτα benchmark για CPU, που σκοπό έχουν να στρεσάρουν τον επεξεργαστή του συστήματος, το υποσύστημα μνήμης και τον μεταγλωττιστή (compiler).

Η SPEC σχεδίασε αυτή τη σουίτα για να παρέχει ένα συγκριτικό μέτρο επιδόσεων εντάσεως υπολογισμών σε όλο το ευρύτερο φάσμα υλικού που χρησιμοποιεί workloads που αναπτύχθηκαν από πραγματικές εφαρμογές χρηστών. Αυτά τα benchmarks παρέχονται ως πηγαίος κώδικας και απαιτούν από τον χρήστη να τα μεταγλωττίσει σε εκτελέσιμα binaries. Στην περίπτωση μας, τα benchmarks αυτά μεταγλωττίστηκαν για να τρέχουν σε επεξεργαστή της αρχιτεκτονικής Alpha.

Τα Benchmark από τη σουίτα Spec2006 που χρησιμοποιήσαμε στην μελέτη αυτή, έτρεχαν πάντα σε ντετερμινιστικό περιβάλλον και ήταν τα ακόλουθα:

astar, gobmk, bzip2, hmmer, sjeng, soplex, namd, sphinx3, h264ref, mcf, perlbench, omnetpp, gamess, gromacs, bwaves, leslie3d, zeusmp, GemsFDTD, milc, cactusADM

6.3 Αρχιτεκτονική Υλοποίησης

Για τους σκοπούς της μελέτης αυτής, ο προσομοιωτής Sim-Alpha επεκτάθηκε για να καταγράφει τα Wrong Path Events μέσα από τη χρήση του 8-bit vector που αναλύσαμε πιο πάνω, αλλά και για να καταγράφει κάποιες στατιστικές μετρήσεις που θα επέτρεπαν την αξιολόγηση του Performance Model για το μηχανισμό ανάκαμψής που προτείνουμε μέσα από αυτή τη διπλωματική εργασία.

Opportunity Window ορίζουμε το μέγιστο αριθμό κύκλων από τη διαφορά μεταξύ μιας εντολής που κάνει Resolve και όλων των νεότερων εντολών σε program order που έχουν κάνει Resolve και βρίσκονται στο Reorder Buffer. Δηλαδή είναι η μέγιστη διαφορά σε κύκλους από τον κύκλο που έγινε Resolve η εντολή που εξετάζουμε σε σχέση με όλες τις νεότερες εντολές που έκαναν Resolve νωρίτερα. (Οι νεότερες εντολές θα πρέπει να είναι είτε F_CTRL εντολές, είτε να αποτελούν Replay Traps)

Το στατιστικό του Opportunity Window είναι ιδιαίτερα χρήσιμο, εφόσον χρησιμοποιείται στο Performance Model για να υπολογιστεί το περιθώριο βελτίωσης από τον μηχανισμό ανάκαμψής που προτείνουμε σε αυτή την μελέτη.

Οι μετρικές που εξάγει ο αναβαθμισμένος προσομοιωτής Sim-Alpha είναι οι ακόλουθες:

- **IPC**: Instructions Per Cycle
- **CPI**: Cycles Per Instruction
- **sim_IPB**: Instructions Per Branch
- **MPKI**: Miss-predicts per Kilo Instructions = Miss-predicted Branches Committed / (Commits/1000)

- **Commits:** Total Number of Committed Instructions
- **Issued:** Total Number of Issued Instructions
- **Miss-predicted F_CTRL Committed:** Total Number of Miss-predicted F_CTRL Committed Instructions
- **Miss-predicted F_CTRL Resolved:** Total Number of Miss-predicted F_CTRL Resolved Instructions
- **Miss-predicted F_CTRL in the Shadow of Traps :** Total Number of Miss-predicted F_CTRL Resolved Instructions in the Shadow of Traps

- **wb_load_replaytrap**: total number of load replay traps
- **wb_store_replaytrap**: total number of store replay traps
- **wb_diffsize_replaytrap**: total number of different size replay traps
- **cm_load_replaytrap**: total number of load replay traps at commit stage
- **cm_store_replaytrap**: total number of store replay traps at commit stage
- **cm_diffsize_replaytrap**: total number of different size replay traps at commit

- **avg_br_correct_latency_sum**: for the correct predicted branches that COMMIT I calculate the difference from (COMMIT cycle - fetch cycle)
- **avg_br_correct_latency_num**: number of correct predicted branches that COMMIT
- **avg_br_misspredicted_latency_sum**: for the miss predicted branches that COMMIT I calculate the difference from (COMMIT cycle - fetch cycle)
- **avg_br_misspredicted_latency_num**: number of miss predicted branches that COMMIT

- **avg_br_correct_latency_resolution_sum**: for the correct predicted branches that COMMIT I calculate the difference from (resolution cycle - fetch cycle)
- **avg_br_misspredicted_latency_resolution_sum:** for the miss predicted branches that COMMIT I calculate the difference from (resolution cycle - fetch cycle)

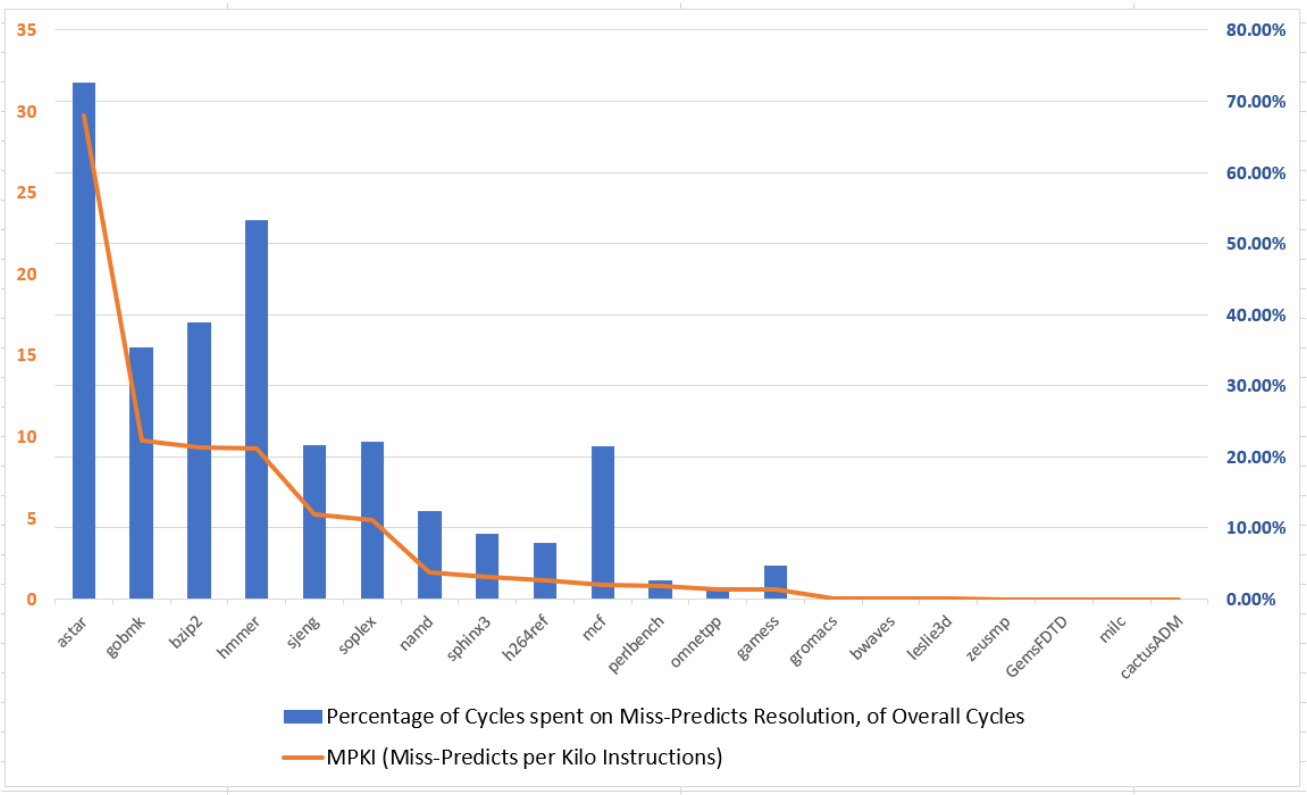
- **indirect_branches:** total number of committed indirect jmp
- **indirect_mispr_branches:** total number of committed mispredicted indirect jmp
- **ret_branches:** total number of committed return branches
- **ret_mispr_branches:** total number of committed mispredicted return branches
- **indirect_calls+jumps-rets(correct):** total number of correct committed indirect jmp, calls but not returns
- **indirect_calls+jumps-rets(misspredicted):** total number of committed misspredicted indirect jmp, calls but not returns

- **myOpportunityWindow_num**: Number of Miss-Predicted Conditional Branches that Commit and had an Oppurtunity Window at the time they were Resolved.
- **myOpportunityWindow_sum**: Sum of the Oppurtunity Windows of the Miss-Predicted Conditional Branches that Commit

Κεφάλαιο 7

Αποτελέσματα και Ανάλυση

7.1 Miss-Predicts Impact on Performance



Στο πιο πάνω γράφημα βλέπουμε να αναπαριστούντε για κάθε ένα από τα Benchmarks της σουίτας Spec2006, στον αριστερό άξονα, το πλήθος των MPKI (Miss-Predicts Per Kilo Instructions). Αυτό το στατιστικό μας αποκαλύπτει την αναλογία των Miss-Predict εντολών έναντι του συνολικού αριθμού εντολών του εκάστοτε Benchmark .
Στον δεξιό άξονα, παρατηρούμε το ποσοστό των κύκλων των οποίω σπαταλούνται για την λανθασμένη κερδοσκοπική εκτέλεση αυτών των Miss-Predicts από τον συνολικό αριθμό κύκλων εκτέλεσης κάθε Benchmark.

Παρατηρούμε ότι υπάρχουν διακυμάνσεις μεταξύ των Benchmarks όσων αφορά στην επίδραση που έχουν τα Miss-Predicts στην επίδοση τους. Η πλειοψηφία των Benchmarks, όπως στο astar, φαίνεται να επηρεάζονται πολύ έντονα στην επίδοση τους από τα Miss-Predicts. Αντιθέτως, κάποια Benchmarks, όπως το bwaves, φαίνεται να μην επηρεάζονται καθόλου από τα Miss-Predicts τους, αν και αυτά τα Benchmarks αποτελούν μειονότητα. Γενικότερα βλέπουμε ότι το penalty από τα Miss-Predicts λαμβάνει σημαντικό μέρος του χρόνου εκτέλεσης.

7.2 Opportunity Window Analysis

Στο πιο πάνω γράφημα βλέπουμε να αναπαριστούντε για 13 από τα Benchmarks της σουίτας Spec2006, με μπλε χρώμα, το ποσοστό των Miss-Predicts που έχουν Opportunity Window, και με πορτοκαλί χρώμα, το ποσοστό των κύκλων που λαμβάνουν συνολικά τα Opportunity Windows από τον συνολικό αριθμό κύκλων εκτέλεσης των Miss-Predicts που έχουν Opportunity Window.

Στο γράφημα συμπεριλαμβάνονται μόνο τα Benchmarks που φάνηκαν να βιώνουν σημαντική επίπτωση στο χρόνο εκτέλεσης τους από τα Miss-Predicts (13 από τα 20 Benchmarks).

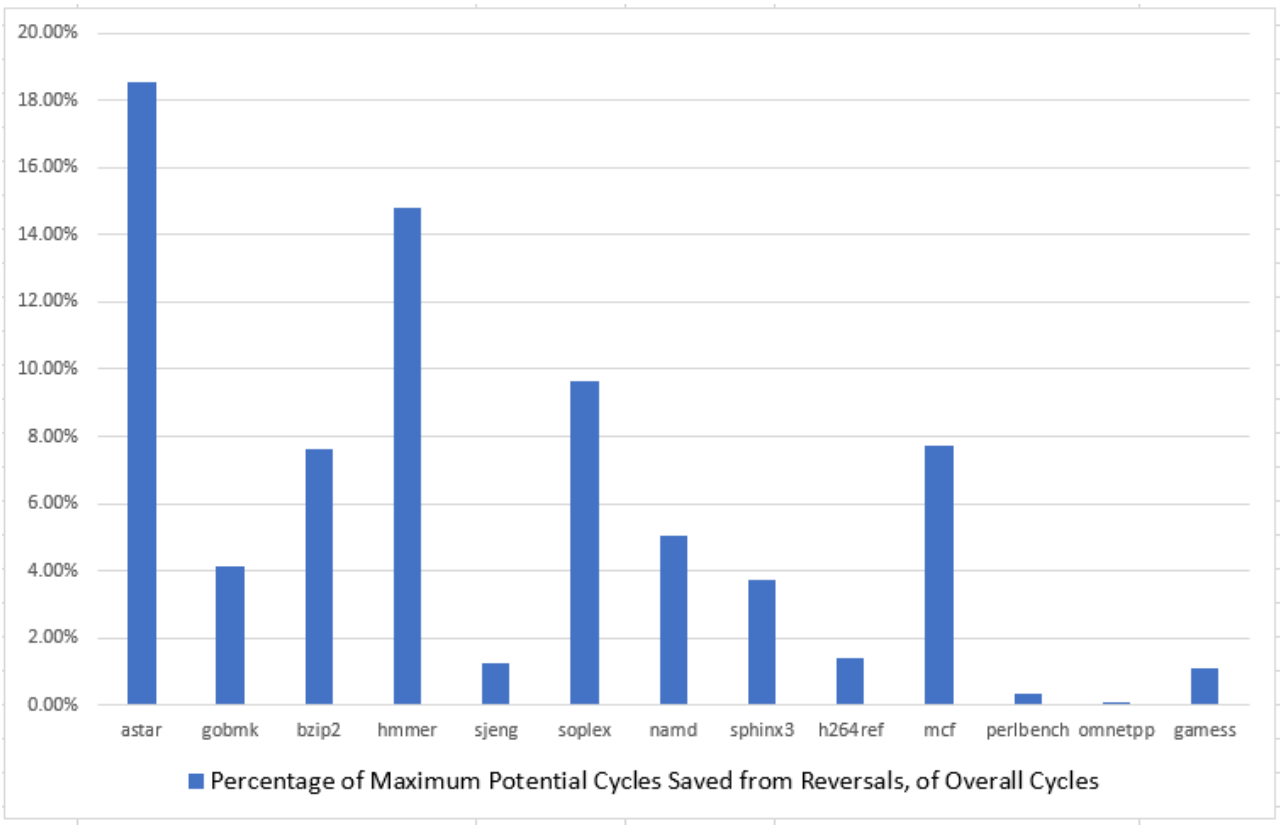
Όπως παρατηρούμε το πλήθος των Opportunity Windows αλλά και το μέγεθος τους σε κύκλους, λαμβάνει αρκετά μεγάλο ποσοστό των Miss-Predicts, γεγονός που αποδεικνύει το δυνητική ωφελιμότητα που μπορεί να προκύψει από τον μηχανισμό των reversals βάση των Wrong Path Events.

Ο μέσος ορός των ποσοστών του γραφήματος είναι 53% για το ποσοστό των Miss-Predicts που έχουν Opportunity Window, και 44% για το ποσοστό των κύκλων που λαμβάνουν συνολικά τα Opportunity Windows από τον συνολικό αριθμό κύκλων εκτέλεσης των Miss-Predicts που έχουν Opportunity Window. Αυτό επιβεβαιώνει το συμπέρασμα ότι τα Miss-Predicts επιβαρύνουν σημαντικά την επίδοση αυτών των Benchmarks

(Υπενθύμιση: Opportunity Window ορίζουμε το μέγιστο αριθμό κύκλων από τη διαφορά μεταξύ μιας εντολής που κάνει Resolve και όλων των νεότερων εντολών σε program order που έχουν κάνει Resolve και βρίσκονται στο Reorder Buffer. Δηλαδή είναι η μέγιστη διαφορά σε κύκλους από τον κύκλο που έγινε Resolve η εντολή που εξετάζουμε σε σχέση με όλες τις νεότερες εντολές που έκαναν Resolve νωρίτερα. Οι νεότερες εντολές θα πρέπει να είναι είτε F_CTRL εντολές, είτε να αποτελούν Replay Traps

Το στατιστικό του Opportunity Window είναι ιδιαίτερα χρήσιμο, εφόσον χρησιμοποιείται στο Performance Model για να υπολογιστεί το περιθώριο βελτίωσης από τον μηχανισμό ανάκαμψής που προτείνουμε σε αυτή την μελέτη.)

7.3 Potential Improvement from Reversals



Στο πιο πάνω γράφημα βλέπουμε να αναπαριστούντε για 13 από τα Benchmarks της σουίτας Spec2006, το μέγιστο ποσοστό των κύκλων που θα μπορούσαν δυνητικά να εξοικονομηθούν από τον μηχανισμό των reversals βάση των Wrong Path Events.

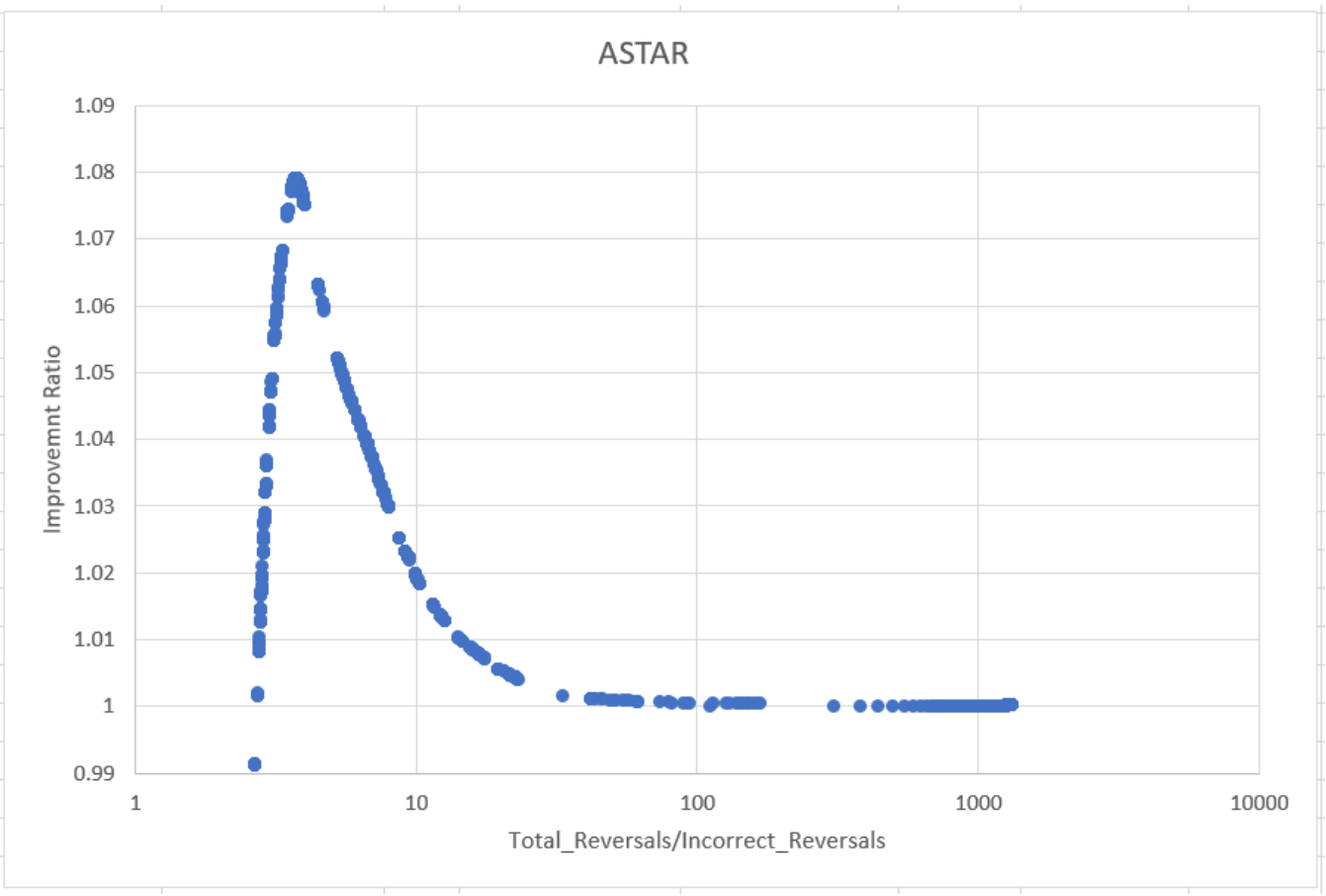
Αυτό το ποσοστό προέκυψε από το γινόμενο 3 κλασμάτων, το ποσοστό των κύκλων των οποίω σπαταλούνται για την λανθασμένη κερδοσκοπική εκτέλεση αυτών των Miss-Predicts από τον συνολικό αριθμό κύκλων εκτέλεσης κάθε Benchmark, το ποσοστό των Miss-Predicts που έχουν Opportunity Window και το ποσοστό των κύκλων που λαμβάνουν συνολικά τα Opportunity Windows από τον συνολικό αριθμό κύκλων εκτέλεσης των Miss-Predicts που έχουν Opportunity Window.

Το πιο πάνω στατιστικό αποτελεί άνω φράγμα για την ανάλυση της δυνητικής ωφελιμότητας του μηχανισμού των reversals βάση των Wrong Path Events που θέσαμε. Ο μέσος ορός της δυνητικής ωφελιμότητας του μηχανισμού για τα 13 Benchmarks του γραφήματος είναι 7%. Κατά συνέπεια, αποδεικνύεται ότι ο συγκεκριμένος μηχανισμός δίνατε να προσφέρει σημαντική βελτίωση στην μείωση των κύκλων που σπαταλούνται σε ένα πρόγραμμα από τα Miss-Predicts και εν τέλη να προσφέρει βελτίωση στον συνολικό χρόνο εκτέλεσης του προγράμματος.

7.5 Performance Boost Analysis

Βάση του Performance Model που εισάχθηκε νωρίτερα, έγινε μια λεπτομερής ανάλυση για ένα από τα Benchmarks τα οποία παρουσίασαν μεγάλο αριθμό MPKI (Miss-Predicts Per Kilo Instructions), το astar Benchmark.
Για το Benchmark αυτό ταξινομήσαμε τα Wrong Path Events που εντοπίσαμε με τη βοήθεια των 8 Features που αναλύσαμε προηγουμένως, σε φθίνουσα σειρά βάση της αναλογίας εμφάνισης τους σε Miss-Predicts και Correct-Predicts. (Δηλαδή πρώτο σε σειρά είναι το event με μεγαλύτερο ratio όσων Total_Reversals/Incorrect_Reversals)

Σκοπός της ανάλυσης αυτής είναι να παρακολουθήσουμε την επίδραση του μηχανισμού ανάκαμψής σε σχέση με την συνολική επίδοση του προγράμματος και το ποσοστό των επιτυχημένων Reversals που έχουμε εφαρμόσει.



Στο πιο πάνω γράφημα βλέπουμε να αναπαριστάτε το ποσοστό βελτίωσης (Improvement Ratio) στην επίδοση του astar Benchmark, σε σχέση με την ανεκτικότητα σε Incorrect_Reversals (Total_Reversals/Incorrect_Reversals).

Από το γράφημα παρατηρούμε ότι με την ιδανικότερη ανεκτικότητα σε Incorrect_Reversals, δηλαδή με ratio 3.75 για το Total_Reversals/Incorrect_Reversals , πετυχαίνουμε συνολική βελτίωση 108% στην επίδοση του χρόνου εκτέλεσης του astar.

Αυτά τα αποτελέσματα είναι άκρως ενθαρρυντικά και απόδειξη ότι η μηχανισμός ανάκαμψής με βάση τα Wrong Path Events που ορίσαμε μπορεί να προσφέρει σημαντική βελτίωση στην επίδοση του προγράμματος, μειώνοντας το Miss-Predicts Penalty.

Κεφάλαιο 8

Συμπεράσματα και Μελλοντικό Έργο

8.1 Συμπεράσματα

Μέσα από τη μελέτη της Διπλωματικής μου Εργασίας , έχοντας αναλύσει την συμπεριφορά των Benchmarks της σουίτας Spec2006, καταφέραμε να καταμετρήσουμε την επίδραση που έχουν τα Miss-Predicts στην συνολική επίδοση αυτών των προγραμμάτων. Φάνηκε λοιπόν, πως τα Miss-Predicts επιβαρύνουν αρκετά την συνολική επίδοση, προσθέτοντας μεγάλο αριθμό κύκλων στο χρόνο εκτέλεσης λόγω του Miss-Predict Penalty.
Ακολούθως, προχωρήσαμε στην αναζήτηση γεγονότων που θα αναγνώριζαν εγκαίρως τα Miss-Predicts στην κερδοσκοπική εκτέλεση (Wrong Path Events) και αφού προτείναμε το σύστημα με τα 8 Features, το αξιολογήσαμε μελετώντας το πόσο μπορούσε να διαφοροποίηση τα Correct-Predicts από τα Miss-Predicts . Από την αξιολόγησή αυτή μαζί με την αξιολόγησή για το περιθώριο χρόνου (Opportunity Window) που μας επέτρεπαν τα WPE να εκμεταλλευτούμε, αποδείξαμε ότι ο συγκεκριμένος μηχανισμός θα μπορούσε να προσφέρει σημαντική βελτίωση στο χρόνο εκτέλεσης των προγραμμάτων μειώνοντας το Miss-Predict Penalty .
Τέλος, χρησιμοποιώντας το Performance Model που προτείναμε κατά τη διάρκεια αυτής της μελέτης, αξιολογήσαμε σε βάθος τη βελτίωση του Benchmark astar, ενός Benchmark με πολλά MPKI (Miss-Predicts Per Kilo Instructions), και αποδείξαμε ότι για τις κατάλληλες παραμέτρους, που ορίσαμε στο Performance Model , ο μηχανισμός ανάκαμψής από Wrong Path στην κερδοσκοπική εκτέλεση χρησιμοποιώντας τα WPE που ορίσαμε, μπορεί να βελτιώσει σημαντικά την επίδοση του προγράμματος.

82 Μελλοντικό Έργο

- Ανάλυση όλων των Benchmarks της σουίτας Spec2006, χρησιμοποιώντας το Performance Model
- Αύξηση των ιδιοτήτων που χρησιμοποιούμε για να εντοπίσουμε τα Wrong Path Events, από 8 που είναι τα υφιστάμενα, έτσι ώστε να επιχειρήσουμε να διαφοροποιήσουμε ακόμα περισσότερο τα Correct-Predicts από τα Miss-Predicts και έτσι να αυξήσουμε το ratio Correct_Reversals/Incorrect_Reversals
- Υλοποίηση του μηχανισμού ανάκαμψής από Miss-Predicts στην κερδοσκοπική εκτέλεση χρησιμοποιώντας τα WPE που ορίσαμε, στον προσομοιωτή Sim-Alpha, έτσι ώστε να δούμε πρακτικά την ωφελιμότητα που προσφέρει στα προγράμματα

Βιβλιογραφία

[1] J. Smith and G. Sohi, Superscalar Processors Microarchitecture, TR UW 1995

[2] S. McFarling. Combining branch predictors. Technical Report TN-36, Digital Western Research Laboratory, June 1993

[3] David N. Armstrong Hyesoon Kim Onur Mutlu Yale N. Patt, Wrong Path Events: Exploiting Unusual and Illegal Program Behavior for Early Misprediction Detection and Recovery, MICRO-37 2004

[4] Moritz Lipp and Michael Schwarz, Meltdown: Reading Kernel Memory from User Space, 2018

[5] Paul Kocher and Jann Horn, Spectre Attacks: Exploiting Speculative Execution, 2019