
ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών
Εργαστήριο Υπολογιστικών Συστημάτων


# Ανάλυση και Μοντελοποίηση Ετερογενών Επεξεργαστών ARM

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του


**Γεώργιου Χρήστου Τσιατσιάνη**

**Επιβλέπων**: Νεκτάριος Κοζύρης
Καθηγητής Ε.Μ.Π.



Αθήνα, Μάρτιος 2019

**Εθνικό Μετσόβιο Πολυτεχνείο**
Τμήμα Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και
Υπολογιστών
Εργαστήριο Υπολογιστικών Συστημάτων

# Ανάλυση και Μοντελοποίηση Ετερογενών Επεξεργαστών ARM

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

**Γεώργιου Χρήστου Τσιατσιάνη**

**Επιβλέπων**: Νεκτάριος Κοζύρης
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή στις 20 Μαρτίου 2019.

| ..... | ..... | ..... |
|---|---|---|
| Νεκτάριος Κοζύρης | Γεώργιος Γκούμας | Νικόλαος Παπασπύρου |
| Καθηγητής Ε.Μ.Π. | Επίκουρος Καθηγητής Ε.Μ.Π. | Καθηγητής Ε.Μ.Π. |

Αθήνα, Μάρτιος 2019

.....
Γεώργιος Χρήστος Τσιατσιάνης
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών
Ε.Μ.Π.

## Περίληψη

Τα τελευταία χρόνια η ραγδαία αύξηση της υπολογιστικής ισχύος αναπόδραστα έχει προκαλέσει ζητήματα χειρισμού της ενεργειακής κατανάλωσης, τόσο σε φορητές συσκευές (tablet, κινητά) όσο και σε εξυπηρετητές και υπερυπολογιστές. Μία από τις μεθόδους που χρησιμοποιείται για τη ρύθμιση της ενεργειακής κατανάλωσης είναι η χρήση ετερογενών υπολογιστικών συστημάτων. Τέτοιου είδους συστήματα είναι οι επεξεργαστές big.LITTLE της εταιρίας ARM, οι οποίοι περιέχουν out-of-order big πυρήνες και in-order little πυρήνες σε διαφορετικά εύρη συχνοτήτων, με στόχο την εξοικονόμηση ενέργειας δίχως αισθητή διαφορά στην απόδοση.

Στην παρούσα διπλωματική εργασία, αναλύσαμε εφαρμογές από τις σουίτες μετροπρογραμμάτων Spec2006 και Parsec. Πραγματοποιήσαμε κατάλληλες μετρήσεις με στόχο να βρούμε τους χρόνους εκτέλεσης των προγραμμάτων στα διαφορετικά είδη πυρήνων, το instruction mix τους, τη συμπεριφορά των cache μνημών τους, καθώς και την ενεργειακή τους κατανάλωση. Με βάση αυτές τις μετρήσεις, επιχειρήσαμε να κατανοήσουμε ποιοι παράγοντες καθορίζουν την επίδοση και την ενεργειακή απόδοση ενός προγράμματος στο κάθε είδος πυρήνα και να την μοντελοποιήσουμε.

Για τη μοντελοποίηση των επεξεργαστών χρησιμοποιήσαμε πέντε μοντέλα μηχανικής μάθησης. Πιο συγκεκριμένα, χρησιμοποιήσαμε τη μέθοδο Logistic Regression με Stochastic Gradient Descent, τη μέθοδο Decision Tree, τη μέθοδο Random Forest, τη μέθοδο kNN(k Nearest Neighbours) και τη μέθοδο Multilayer Perceptron. Εισάγαμε στις εισόδους των μοντέλων μηχανικής μάθησης το instruction mix και τη συμπεριφορά των caches των προγραμμάτων με στόχο να μπορούμε να προβλέψουμε το βέλτιστο συνδυασμό ενεργειακής κατανάλωσης (Energy Delay Product και Energy Delay$^2$ Product) για το υπολογιστικό σύστημα.



**Λέξεις κλειδιά**

Ετερογενείς αρχιτεκτονικές, ARM, big.LITTLE, ανάλυση, μοντελοποίηση, μηχανική μάθηση

**Abstract**

Nowadays the rampant increase of computing performance has caused important issues in energy consumption. Current computing systems that target different domains, such as portable devices (tablets and mobiles), servers, and supercomputers, exhibit energy consumption deficiencies. Heterogeneous architectures try to address that issue. ARM big.LITTLE is a heterogeneous computing architecture developed by ARM Holdings, that couples battery-saving and slower processor cores (LITTLE) with more powerful and power-hungry ones (big).

In this diploma thesis, we analyzed the behavior of various applications from both Spec2006 and Parsec benchmark suites on an ARM big.LITTLE heterogeneous system. We made extensive experiments to measure the performance of the benchmarks on the different cores, the instruction mix, the behavior of the caches, and the energy consumption. We used these measurements, together with findings from prior works, to understand which factors affect the behavior of the programs and to model that behavior.

For the modeling approach, we used five machine learning models to predict on which core (big or LITTLE) each workload should be executed when considering the Energy Delay Product (EDP) και Energy Delay$^2$ Product (ED$^2$P) metrics. More specifically, we used the following machine learning models: Logistic Regression with Stochastic Gradient Descent, Decision Trees, Random Forests, k Nearest Neighbours (kNN) and Multilayer Perceptron.

**Keywords**

Heterogeneous architectures, ARM, big.LITTLE, analysis, modeling, machine learning

## Ευχαριστίες

Κατ' αρχάς θα ήθελα να ευχαριστήσω όλο το προσωπικό του CSLab για το υψηλό επίπεδο γνώσεων, το οποίο παρέχει. Θέλω να ευχαριστήσω θερμά τον κ. Νεκτάριο Κοζύρη για την υποστήριξή του καθ' όλη τη διάρκεια εκπόνησης της παρούσας διπλωματικής εργασίας. Επίσης ευχαριστώ τους κ. Νεκτάριο Κοζύρη και κ. Γεώργιο Γκούμα για τις πολύτιμες γνώσεις που μου προσέφεραν στις διαλέξεις τους. Οφείλω να "πω", επίσης, ένα μεγάλο ευχαριστώ σε όλο το εκπαιδευτικό προσωπικό της σχολής για όσα διδάχτηκα τα τελευταία χρόνια. Ιδιαίτερες ευχαριστίες οφείλω για την υλοποίηση της παρούσας διπλωματικής, στο εργαστήριο ευφυών υπολογιστικών συστημάτων.

Ευχαριστώ ιδιαίτερα τον διδάκτορα Βασίλη Καρακώστα για την αμέριστη βοήθεια και καθοδήγηση που μου προσέφερε, καθώς και για τις ατέλειωτες ώρες ενασχόλησης κατά την εκπόνηση της παρούσας διπλωματικής. Χωρίς τη συστηματική καθοδήσή του δε θα μπορούσε, σε καμία περίπτωση, να πραγματοποιηθεί η συγκεκριμένη διπλωματική.

Ευχαριστώ θερμά τον Guillermo Callaghan για τη βοήθειά του σχετικά με το mambo instrumentation binary tool.

Τέλος, οφείλω ένα μεγάλο ευχαριστώ στην οικογένειά μου και στους γονείς μου ιδιαίτερα, για τη στήριξή τους καθ' όλη τη διάρκεια των σπουδών μου.

# Περιεχόμενα

# Ευρετήριο Πινάκων

# Ευρετήριο Εικόνων

# Κεφάλαιο 1

# Εισαγωγή

## 1.1 Κίνητρο και έρευνα

Τα τελευταία χρόνια η ραγδαία αύξηση της υπολογιστικής ισχύος, αναπόδραστα έχει προκαλέσει ζητήματα χειρισμού της ενεργειακής κατανάλωσης τόσο σε φορητές συσκευές (tablet, κινητά), όσο και σε υπερυπολογιστές. Οι σημερινές συσκευές καλούνται να εξυπηρετήσουν "έξυπνες" και πιο σύνθετες εργασίες, όπως ο έλεγχος της ανθρώπινης φωνής και κίνησης, σε συνδυασμό με τη διαρκή μεταφορά, λήψη και διαχείριση δεδομένων. Η πολυπλοκότητα της διεπαφής χρήστη έχει αυξηθεί σημαντικά στις φορητές συσκευές, όπως, επίσης και οι υπολογιστικές ανάγκες για ηλεκτρονικά παιχνίδια σε τέτοιου είδους συσκευές. Κινητά και tablet χρησιμοποιούνται όλο και περισσότερο σαν κονσόλες παιχνιδιών.

Μία από τις προτεινόμενες λύσεις για τη ρύθμιση της ενεργειακής κατανάλωσης είναι η χρήση ετερογενών υπολογιστικών συστημάτων. Στις ετερογενείς αρχιτεκτονικές οι πυρήνες παρουσιάζουν διαφορετικότητα. Ο στόχος των ετερογενών αρχιτεκτονικών είναι η βέλτιστη χρησιμοποίηση των υπολογιστικών κόμβων, ώστε να επιτευχθεί μεγιστοποίηση της απόδοσης και η ταυτόχρονη ελαχιστοποίηση της ενεργειακής κατανάλωσης. Η ετερογένεια των πυρήνων μπορεί μπορεί να αφορά τη συχνότητα λειτουργίας των υπολογιστικών κόμβων, τις δομικές τους δυνατότητες ή μπορεί να αναφέρεται ακόμα και σε υπολογιστικούς κόμβους που υποστηρίζουν διαφορετικά μοντέλα εκτέλεσης εντολών.

Η τεχνολογία ARM big.LITTLE αποτελεί είδος ετερογενούς αρχιτεκτονικής. Αναπτύχθηκε από την εταιρεία ARM Holdings και επιχειρεί να συνδυάσει πιο αργούς little πυρήνες με τους ταχύτερους, αλλά και ενεργειακά πιο κοστοβόρους, big πυρήνες, με στόχο την εξοικονόμηση ενέργειας, δίχως να υπάρχει αισθητή διαφορά στην απόδοση. Ο στόχος είναι να δημιουργηθούν πολυπύρηνα συστήματα, τα οποία προσαρμόζουν δυναμικά τις υπολογιστικές

τους δυνατότητες, ώστε να είναι ενεργειακά πιο αποδοτικά.

Πρόκληση αποτελεί σήμερα η μοντελοποίηση και χρονοδρομολόγηση τέτοιου είδους ετερογενών υπολογιστικών συστημάτων. Στόχος είναι κάθε εφαρμογή να μπορεί να εκτελεστεί στο κατάλληλο είδος πυρήνα, με τον κατάλληλο αριθμό threads στην κατάλληλη συχνότητα. Συνεπώς χρειαζόμαστε μία μεθοδολογία αξιολόγησης και πρόβλεψης της συμπεριφοράς των προγραμμάτων. Επιδιώκεται, μέσω της συγκεκριμένης μεθοδολογίας, κάθε πρόγραμμα να εκτελείται στον κατάλληλο συνδυασμό είδος πυρήνα-αριθμός threads-συχνότητα λειτουργίας, ώστε να επιτυγχάνεται εξοικονόμηση ενέργειας, δίχως σημαντική υποβάθμιση της απόδοσης.

Στην παρούσα διπλωματική επιχειρούμε να αναλύσουμε τη συμπεριφορά των προγραμμάτων κατά την εκτέλεσή τους σε ARM big.LITTLE σύστημα και στη συνέχεια να μοντελοποιήσουμε τη συμπεριφορά τους. Στόχος μας είναι να μπορούμε να προβλέψουμε σε ποιο είδος πυρήνα πρέπει να εκτελεστεί κάθε εφαρμογή ώστε να θεωρείται ενεργειακά αποδοτική, ενώ ταυτόχρονα να μην παρατηρείται αισθητή διαφορά στην απόδοση. Για να πετύχουμε το στόχο μας ακολουθήσαμε την μεθοδολογία που περιγράφουμε παρακάτω.

Σε πρώτο στάδιο πραγματοποιήσαμε μετρήσεις πάνω σε εφαρμογές από τις σουίτες μετροπρογραμμάτων Spec2006 και Parsec. Μετρήσαμε τους χρόνους εκτέλεσης των προγραμμάτων, το instruction mix τους, τη συμπεριφορά των μνημών caches τους, την ενεργειακή τους κατανάλωση και τη συμπεριφορά τους σε παράλληλες εκτελέσεις. Στη συνέχεια με βάση τα αποτελέσματα των μετρήσεων, αλλά και άλλες έρευνες επιχειρήσαμε να αναλύσουμε τη συμπεριφορά των προγραμμάτων. Εξετάσαμε ποιοι παράγοντες ευνοούν την εκτέλεση προγραμμάτων σε big και ποιοι σε little πυρήνες, τόσο σε επίπεδο απόδοσης, όσο και σε επίπεδο ενεργειακής κατανάλωσης.

Σε επόμενο στάδιο επιχειρήσαμε να μοντελοποιήσουμε τη συμπεριφορά των προγραμμάτων, δηλαδή σε ποιο είδος πυρήνα έπρεπε να εκτελεστούν για να ικανοποιούνται συγκεκριμένα κριτήρια και στόχοι, όπως οι συντελεστές EDP (Energy Delay Product), $ED^2P$ (Energy Delay Delay Product). Για το σκοπό αυτό χρησιμοποιήσαμε μοντέλα μηχανικής μάθησης. Θεωρούμε ότι η τεχνητή νοημοσύνη (artificial intelligence) και ιδιαίτερα η τεχνητή μάθηση (machine learning) μπορεί να δώσει τη βέλτιστη λύση σε ένα ευρύ φάσμα προβλημάτων, όπως και εκείνο της μοντελοποίησης ετερογενών επεξεργαστών. Χρησιμοποιήσαμε τη μέθοδο Logistic Regression using Stochastic Gradient Descent, τη μέθοδο Decision Tree, τη μέθοδο Random Forest, τη μέθοδο kNN (k Nearest Neighbours) και τη μέθοδο Multilayer Perceptron. Επιλέξαμε τα παραπάνω μοντέλα για δύο λόγους. Πρώτον, αποτελούν διαφορετικές μεθόδους machine learning και καλύπτουν διαφορετικά εύρη learning αλγορίθμων είτε για γραμμικά είτε για μη γραμμικά προβλήματα. Δεύτερον οι συγκεκριμένες μέθοδοι παράγουν ντετερμινιστικές προβλέψεις, οι οποίες είναι κατάλληλες στο

να προβλέψουν τους ζητούμενους συντελεστές.

Χρησιμοποιήσαμε τα παραπάνω μοντέλα μηχανικής μάθησης για την πρόβλεψη των συντελεστών EDP (Energy Delay Product), $ED^2P$ (Energy Delay Delay Product) και παράλληλης επίδοσης εφαρμογών μέσω της open-source διανομής anaconda και της βιβλιοθήκης scikit-learn της python. Ως εισόδους, για την εκπαίδευση των μοντέλων, χρησιμοποιήσαμε ομαλοποιημένο τον αριθμό branches, load, store, integer, floating-point, simd και other εντολών, ως ποσοστό επί του συνόλου των εντολών του κάθε προγράμματος. Επιπλέον, ως παραμέτρους εισάγαμε και τα ποσοστά των misses των L1 και L2 caches των δύο ειδών πυρήνων. Συνολικά, δηλαδή, είχαμε έντεκα παραμέτρους εισόδου.

Προβλέψαμε με υψηλά ποσοστά επιτυχίας σε ποιον πυρήνα θα εκτελεστούν σύνολα Spec2006 προγραμμάτων, μονονηματικών Parsec προγραμμάτων, καθώς και σύνολα που περιείχαν και τα δύο είδη προγραμμάτων. Αντίθετα δεν προβλέψαμε με επιτυχία τη συμπεριφορά Spec2006 και μονοπύρηνων Parsec προγραμμάτων, όταν συνεκτελούνται με άλλο πρόγραμμα ταυτόχρονα στο ίδιο σύμπλεγμα πυρήνων. Θεωρούμε ότι γι' αυτόν το λόγο ευθύνονται τα είδη των εισόδων στα μοντέλα μηχανικής μάθησης, όπως περιγράφουμε αναλυτικά στο υποκεφάλαιο 6.3.3 της παρούσας διπλωματικής.

## 1.2 Στόχοι της εργασίας

Οι στόχοι της παρούσας διπλωματικής εργασίας ήταν οι εξής:

- Να περιγράψουμε τι είναι ετερογένεια, καθώς και την τεχνολογία ARM big.LITTLE.

- Να πραγματοποιήσουμε μετρήσεις χρησιμοποιώντας διάφορα benchmarks από Spec2006 και Parsec. Να μετρήσουμε τους χρόνους εκτέλεσης, τα ποσοστά των cache misses, το instruction mix, την συμπεριφορά όταν συνεκτελούνται με άλλες εφαρμογές και την ενεργειακή κατανάλωση αυτών των προγραμμάτων.

- Να αναλύσουμε τη συμπεριφορά των benchmarks και να κατανοήσουμε ποιοι παράγοντες επηρεάζουν την επίδοση των προγραμμάτων στα δύο είδη πυρήνων.

- Να κατανοήσουμε και να περιγράψουμε μοντέλα μηχανικής μάθησης. Συγκεκριμένα περιγράφουμε τη μέθοδο Logistic Regression using Stochastic Gradient Descent, τη μέθοδο Decision Tree, τη μέθοδο Random Forest, τη μέθοδο kNN (k Nearest Neighbours) και το Multilayer Perceptron.

- Να μοντελοποιήσουμε τους επεξεργαστές ARM big.LITTLE, και γενικά ετερογενείς επεξεργαστές, με τη χρήση μοντέλων μηχανικής μάθησης ώστε να μπορούμε να προβλέψουμε σε ποιο είδος πυρήνα θα μπορεί να τρέξει μία μονονηματική εφαρμογή.

- Να υλοποιήσουμε τη μοντελοποίηση με χρήση μοντέλων μηχανικής μάθησης και να αναλύσουμε τα ποσοστά επιτυχημένων προβλέψεων κάθε μεθόδου.

## 1.3 Άλλες εργασίες

Βασιστήκαμε στα official documentations για τη περιγραφή των Spec2006 και των Parsec προγραμμάτων. Για την ανάλυση τους βασιστήκαμε κυρίως σε δύο papers. Για την ανάλυση των Spec2006 προγραμμάτων βασιστήκαμε στο paper [18]. Για την ανάλυση των Parsec προγραμμάτων στηριχτήκαμε στην έρευνα [2].

Όσον αφορά τα μοντέλα μηχανικής μάθησης, η έρευνά μας στηρίχτηκε κυρίως σε δύο papers. Χρησιμοποιήσαμε τα ίδια μοντέλα μηχανικής μάθησης με την έρευνα [13]. Πραγματοποιήσαμε τροποποιήσεις πάνω σε αυτά τα μοντέλα και στη συνέχεια τα επεκτείναμε με στόχο να καλύψουμε, τουλάχιστον σε θεωρητικό επίπεδο, προβλήματα με πολλαπλές τιμές στόχου (multinomial targets). Αξίζει να σημειώσουμε ότι στο κομμάτι των μεθόδων μηχανικής μάθησης ιδιαίτερα ενδιαφέρουσα είναι η έρευνα [17]. Στη συγκεκριμένη έρευνα θα μπορούσαμε να βασιστούμε ώστε να υλοποιήσουμε ένα πρόγραμμα χρονοδρομολόγησης πάνω στους επεξεργαστές ARM big.LITTLE με βάση, πάντα, τη δική μας μοντελοποίηση. Ασχοληθήκαμε και με άλλες εργασίες χρονοδρομολόγησης εργασιών, αλλά η τελική εργασία μας στηρίχτηκε περισσότερο σε αυτές που αναφέραμε.

# Κεφάλαιο 2

# Θεωρητικό Υπόβαθρο και κίνητρο

## 2.1 Νόμος του Moore

Ο συνιδρυτής της Intel και της Fairchild Semiconductor Gordon Moore περιέγραψε σε μία δημοσίευσή του το 1965 ότι ο αριθμός των τρανζίστορ σε ένα ολοκληρωμένο κύκλωμα διπλασιάζεται κάθε δύο χρόνια. Το 1975, κοιτάζοντας ξανά τα δεδομένα για την επόμενη δεκαετία, αναθεώρησε την "πρόβλεψή" του θέτοντας το διάστημα που απαιτείται για τον διπλασιασμό των τρανζίστορ ενός πυκνού ολοκληρωμένου κυκλώματος στα δύο έτη. Η πρόβλεψη επαληθεύτηκε από την πραγματικότητα, καθώς έκτοτε ο αριθμός των τρανζίστορ ενός ολοκληρωμένου κυκλώματος διπλασιάζεται κάθε δύο χρόνια. Η πρόβλεψη του Moore, ύστερα από την πρακτική επαλήθευσή της, ονομάστηκε "Νόμος του Moore".

Συχνά, ως χρόνος για την επιβεβαίωση του "Νόμος του Moore" θεωρούνται οι 18 μήνες, καθώς ο τεχνικός της Intel Ντέιβιντ Χάουζ παρατήρησε πως η απόδοση των μικροεπεξεργαστών θα διπλασιάζεται μετά το πέρας αυτού του διαστήματος, ως συνδυασμός της αύξησης του αριθμού των τρανζίστορ των μικροεπεξεργαστών και της αύξησης της ταχύτητάς τους.

Δυστυχώς με την πάροδο των δεκαετιών, ο διπλασιασμός της πυκνότητας των τρανζίστορ δεν παρέχει ανάλογα αποτελέσματα. Οι λόγοι είναι κυρίως τρεις. Πρώτον, η δημιουργία μονοπύρηνων επεξεργαστών υψηλής πυκνότητας τρανζίστορ αναπόφευκτα οδηγεί σε αύξηση της εκλυόμενης θερμότητας και της πολυπλοκότητας του ολοκληρωμένου. Δεύτερον, ο ρυθμός αύξησης της ταχύτητας των μνημών, καθώς και του διαύλου επικοινωνίας είναι πολύ μικρότερος από αυτόν των επεξεργαστών, με αποτέλεσμα οι προσβάσεις στη μνήμη να προκαλούν μεγάλες καθυστερήσεις. Τρίτον τα μεγέθη των τρανζίστορ πλη-

σιάζουν πλέον την τάξη μεγέθους των ατόμων, δηλαδή ένα θεμελιώδες όριο εξέλιξης.

## 2.2 Νόμος του Flynn

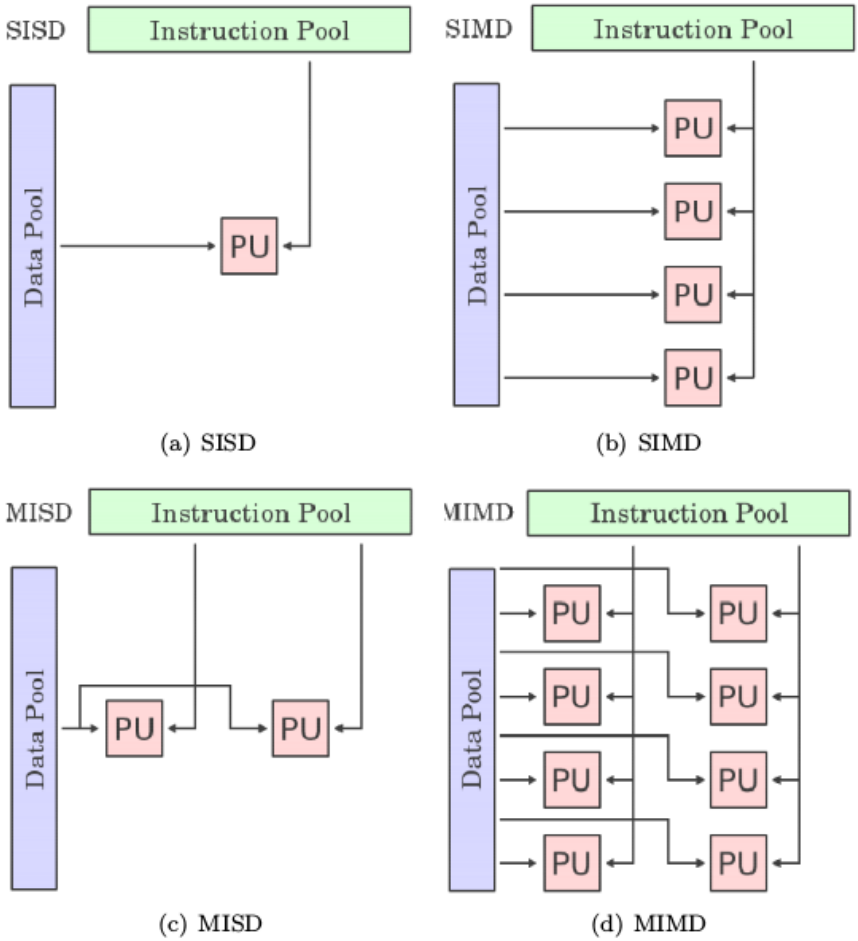
Ο Michael J. Flynn πρότεινε το 1966 την κατηγοριοποίηση των επεξεργαστικών συστημάτων σε τέσσερις κατηγορίες. Τα κριτήρια της παραπάνω ταξινόμησης ήταν ο αριθμός των εντολών τις οποίες επεξεργάζεται ο επεξεργαστής ταυτόχρονα, καθώς και η ροή των δεδομένων. Αναλυτικά παραθέτουμε τις κατηγορίες:

- **Single Instruction, Single Data stream (SISD)**: Οι συγκεκριμένου τύπου πυρήνες δεν εμφανίζουν παραλληλισμό ούτε στις εντολές, ούτε στα δεδομένα. Η μοναδική μονάδα ελέγχου (control unit) φέρνει κάθε εντολή σειριακά σε ξεχωριστές ροές. Η μονάδα ελέγχου γεννά σήματα ελέγχου, ώστε το κατάλληλο στοιχείο επεξεργασίας (processing element) να επεξεργάζεται την μοναδική ροή δεδομένων. Κλασικό παράδειγμα SISD αποτελούν οι μονοπύρηνοι επεξεργαστές των υπολογιστών.

- **Single Instruction, Multiple Data streams (SIMD)**: Οι συγκεκριμένου τύπου πυρήνες εμφανίζουν παραλληλισμό σε επίπεδο δεδομένων. Με μία εντολή μπορούν να επεξεργαστούν παράλληλα πολλαπλές ανεξάρτητες ροές δεδομένων. Παράδειγμα εφαρμογής της SIMD αρχιτεκτονικής παρατηρούμε στις σύγχρονες GPUs.

- **Multiple Instruction, Single Data stream (MISD)**: Οι συγκεκριμένου τύπου πυρήνες υποστηρίζουν την επεξεργασία μίας ροής δεδομένων από πολλές διαφορετικού τύπου εντολές ταυτόχρονα. Δεν παρουσιάζονται σε πολλές αρχιτεκτονικές, και χρησιμοποιούνται κυρίως σε ελέγχους ανοχής σφαλμάτων.

- **Multiple Instruction, Multiple Data streams (MIMD)**: Πολλαπλοί αυτόνομοι πυρήνες εκτελούν διαφορετικές εντολές πάνω σε διαφορετικά δεδομένα ταυτόχρονα. Τα κατανεμημένα συστήματα συνηθίζεται να είναι MIMD, εμφανίζοντας είτε κοινή μοιραζόμενη μνήμη είτε κατανεμημένο χώρο μνήμης.




Εικόνα 2.1: Flynn's Taxonomy.

## 2.3 Ετερογενή Συστήματα

Οι ετερογενείς αρχιτεκτονικές αποτελούν σχεδιάσεις, στις οποίες διαφορετικοί υπολογιστικοί κόμβοι διαθέτουν διαφορετικές δυνατότητες ή και διαφορετικούς τρόπους εκτέλεσης εντολών. Όταν ξεκίνησε η δημιουργία πολυπύρηνων συστημάτων, οι υπολογιστικοί πυρήνες ήταν ομοιόμορφοι. Στις ετερογενείς αρχιτεκτονικές οι πυρήνες παρουσιάζουν διαφορετικότητα. Ο στόχος των ετερογενών αρχιτεκτονικών είναι η βέλτιστη χρησιμοποίηση των υπολογιστικών κόμβων, ώστε να επιτευχθεί μεγιστοποίηση της απόδοσης και η ταυτόχρονη ελαχιστοποίηση της ενεργειακής κατανάλωσης.

Οι πυρήνες ενός επεξεργαστή μπορεί να έχουν τις ίδιες υπολογιστικές δυνατότητες. Για παράδειγμα μπορεί να εμφανίζουν τις ίδιες δυνατότητες παραλληλισμού νημάτων, το ίδιο superscalar width, καθώς και τις ίδιες υποδομές επεξεργασίας διανυσμάτων. Παρόλα αυτά κάθε πυρήνας διαθέτει το δικό του DVFS (dynamic voltage and frequency scaling) μηχανισμό. Ανάλογα με τη συχνότητα που χρησιμοποιεί ο κάθε επεξεργαστής παρουσιάζει διαφορετική απόδοση. Παρατηρούμε ότι ακόμα και πυρήνες με τις ίδιες προδιαγραφές εμφανίζουν ετερογένεια, λόγω ρυθμίσεων στο DVFS. Οι συγκεκριμένου τύπου πολυπύρηνοι επεξεργαστές αποτελούν τη πρώτη κατηγορία ετερογένειας.

Η δεύτερη κατηγορία ετερογενών αρχιτεκτονικών αποτελείται από πυρήνες επεξεργαστή με διαφορετικές δομικές δυνατότητες. Για παράδειγμα, ένας επεξεργαστής μπορεί να περιέχει αρκετούς απλούς πυρήνες με single issue in-order αρχιτεκτονική, μαζί με πιο πολύπλοκους πυρήνες, οι οποίοι υποστηρίζουν SMT τεχνολογία, είναι superscalar και υποστηρίζουν out-of-order speculative εκτέλεση εντολών.

Παρατηρούμε ότι οι πρώτες δύο κατηγορίες ετερογενών αρχιτεκτονικών υποστηρίζουν την εκτέλεση σειριακού φαινομενικά κώδικα, στο εσωτερικό κάθε παράλληλα εκτελούμενου thread. Η τρίτη κατηγορία ετερογενών αρχιτεκτονικών αποτελείται από υπολογιστικούς κόμβους με διαφορετικά μοντέλα εκτέλεσης. Πολλά διαφορετικά είδη κόμβων βρίσκονται στο υπολογιστικό σύστημα. Το πιο χαρακτηριστικό παράδειγμα αυτής της κατηγορίας είναι οι GPUs. Οι GPUs υποστηρίζουν το αρχιτεκτονικό μοντέλο SIMD, όπως αναφέραμε στο προηγούμενο υποκεφάλαιο. Τη συγκεκριμένη μορφή ετερογένειας μπορεί να προκαλέσει και η εισαγωγή FPGA (field-programmable gate array) κόμβου. Είναι ένα ολοκληρωμένο, το οποίο μπορεί να προγραμματιστεί με τη χρήση γλώσσα περιγραφής υλικού, ώστε να επιτελεί μία συγκεκριμένη εργασία. Ας υποθέσουμε ότι θέλουμε να εκτελέσουμε μία πολυωνυμική συνάρτηση σε πληθώρα στοιχείων. Αν ο αριθμός των assembly εντολών δεν είναι τόσο μεγάλος ώστε να χρειάζεται GPU, αλλά ούτε και τόσο μικρός ώστε να εκτελεστεί αποδοτικά από τον επεξεργαστή τότε τα FPGA φαντάζουν η ιδανική λύση. Άλλα είδη κόμβων είναι τα automata processor (AP) από την Micron, τα DSP

(digital signal processor) και τα ASIC (application-specific integrated circuit), τα οποία μπορούν να χρησιμοποιηθούν σε ετερογενή συστήματα.

## 2.4 Τεχνολογία ARM big.LITTLE

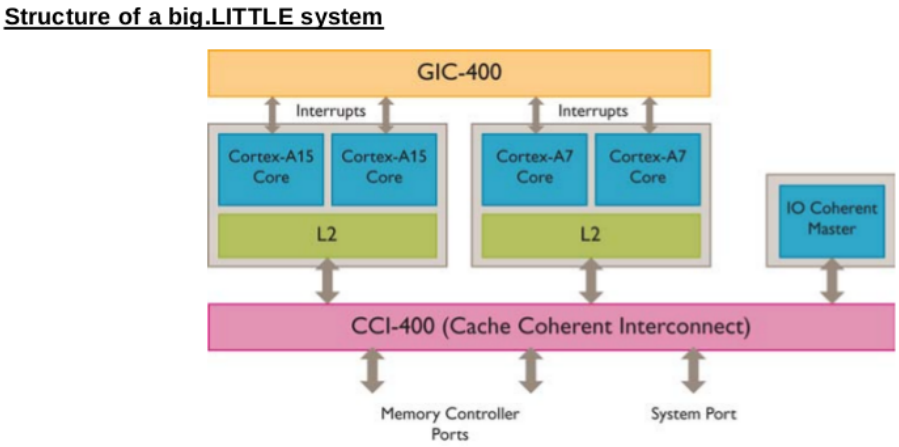
Αφού πραγματοποιήσαμε την εισαγωγή στις ετερογενείς αρχιτεκτονικές, στη συνέχεια θα αναφερθούμε στους πολυπύρηνους επεξεργαστές big.LITTLE. Οι συγκεκριμένοι επεξεργαστές εμφανίζουν την πρώτη και τη δεύτερη μορφή ετερογένειας. Μπορεί να μεταβληθεί το DVFS σε κάθε πυρήνα, ενώ οι out-of-oder big πυρήνες έχουν διαφορετική δομή σε σχέση με τους απλούς in-order little πυρήνες.

Η τεχνολογία ARM big.LITTLE αποτελεί είδος ετερογενούς αρχιτεκτονικής. Αναπτύχθηκε από την εταιρεία ARM Holdings και επιχειρεί να συνδυάσει πιο αργούς little πυρήνες με τους ταχύτερους, αλλά και ενεργειακά πιο κοστοβόρους, big πυρήνες, με στόχο την εξοικονόμηση ενέργειας, δίχως αισθητή διαφορά στην απόδοση. Η τεχνολογία big.LITTLE προέρχεται από μία απλή παρατήρηση: για να δημιουργηθούν ταχύτεροι πυρήνες πρέπει να αυξηθούν οι μονάδες επεξεργασίας (excecution units), οι αποκωδικοποιητές εντολών η χωρητικότητα της μνήμης cache, καθώς και τα υπόλοιπα συστατικά στοιχεία των πυρήνων. Αναπόφευκτα η συγκεκριμένη σχεδιαστική επιλογή αυξάνει την κατανάλωση ενέργειας και σε τομείς όπως η κινητή τεχνολογία εμποδίζεται καθοριστικά η λειτουργικότητά τους. Ο στόχος είναι να δημιουργηθούν πολυπύρηνοι επεξεργαστές, οι οποίοι προσαρμόζονται στις δυναμικές υπολογιστικές ανάγκες του συστήματος και καταναλώνουν λιγότερη ενέργεια.

### 2.4.1 Γενικά χαρακτηριστικά

Η τεχνολογία ARM big.LITTLE είναι μία από τις ετερογενείς αρχιτεκτονικές που υποστηρίζει διαφορετικούς πυρήνες με ίδιο ISA (instruction set architecture), με αποτέλεσμα να μειώνεται η πολυπλοκότητα του συστήματος και η κατανάλωση ενέργειας. Υποστηρίζει διάφορα ζεύγη συμβατών "μεγάλων" και "μικρών" πυρήνων. Ένα από αυτά είναι ο Cortex-A7 (little) και ο Cortex-A15 (big). Ο Cortex-A7 είναι σχεδιασμένος για εξοικονόμηση ενέργειας είναι double issue επεξεργαστικός πυρήνας και υποστηρίζει in-order εκτέλεση εντολών με 8-10 pipeline στάδια. Αντίθετα, ο Cortex-A15 είναι triple issue επεξεργαστικός πυρήνας, επιτρέπει την εκτέλεση out-of-order εντολών με 15-24 pipeline στάδια και είναι ενεργειακά περισσότερο κοστοβόρος ακόμα και στη χαμηλότερη δυνατή κατάσταση λειτουργίας του. Μπορούν, με αυτόν τον τρόπο να εκτελούνται πολλές εντολές παράλληλα σε ένα πιο σύνθετο pipeline.

Με βάση την παραπάνω δομή παραθέτουμε ένα τυπικό σύστημα big.LITTLE,




Εικόνα 2.2: ARM big.LITTLE system structure.

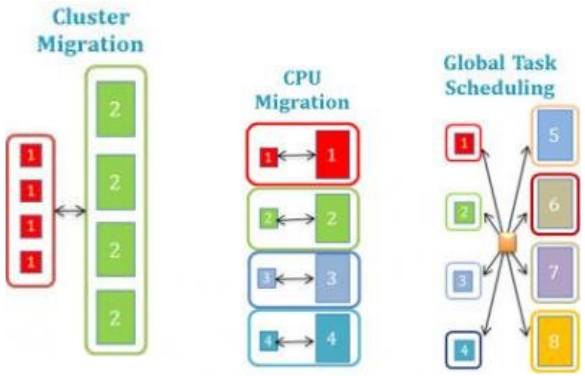
με τους πυρήνες επξεργασίας που περιγράψαμε παραπάνω. Όπως φαίνεται και στην εικόνα υπάρχει μία L2 cache κοινή για κάθε σύμπλεγμα επεξεργαστών, την οποία μοιράζονται οι πυρήνες. Τα δεδομένα που μεταφέρονται, μεταξύ των δύο συμπλεγμάτων πυρήνων εμφανίζουν συνέπεια μέσω του διαδρόμου CCI-400 (Cache Coherent Interconnect). Η ίδια δομή επικοινωνίας επιτρέπει τη μεταφορά πληροφορίας και με άλλα τμήματα, όπως η GPU. Το κανάλι επικοινωνίας GIC-400 χρησιμοποιείται για τη μεταφορά των σημάτων διακοπών σε κάθε επεξεργαστικό πυρήνα του συστήματος.

### 2.4.2 Τρόποι διάταξης των πυρήνων στην αρχιτεκτονική ARM big.LITTLE

Η διάταξη και χρησιμοποίηση των πυρήνων εξαρτάται τόσο από την υλοποίηση του συστήματος όσο και από τον χρονοδρομολογητή που βρίσκεται στον πυρήνα του λειτουργικού συστήματος.

Η πρώτη και πιο απλή προσέγγιση είναι ο διαχωρισμός των πυρήνων σε δύο συμπλέγματα, ανάλογα με το είδος τους big ή little. Ο χρονοδρομολογητής του λειτουργικού συστήματος βλέπει κάθε φορά ένα συμπλεγμα πυρήνων, τους big ή τους little πυρήνες. Όταν κριθεί αναγκαίο το σύστημα εναλλάσσεται από το ένα σύστημα στο άλλο και τα σχετικά δεδομένα μεταφέρονται μέσω της κοινής L2 cache. Το σύμπλεγμα πυρήνων που δεν χρησιμοποιείται τίθεται ανενεργό.

Μία άλλη προσέγγιση χρησιμοποίησης των πυρήνων αποτελεί η δημιουργία ζευγαριών πυρήνων big και little, τα οποία ο χρονοδρομολογητής τα αντιμετωπίζει ως έναν πυρήνα. Βέβαια σε περίπτωση που ο αριθμός πυρήνων του ενός




(1) Cluster migration (2) CPU migration (3) Global task scheduling

Εικόνα 2.3: Τρόποι διάταξης των big.LITTLE πυρήνων.

είδους είναι μεγαλύτερος από εκείνους του άλλου είδους, μπορεί να υπάρχει αντιστοίχιση ενός big ή little πυρήνα σε περισσότερους από έναν little ή big πυρήνες αντίστοιχα. Ανάλογα με τις επεξεργαστικές ανάγκες επιλέγεται δυναμικά το big ή το little τμήμα κάθε εικονικού πυρήνα, η συχνότητα επεξεργασίας (DVFS) και η τάση τροφοδότησης των πυρήνων.

Η πιο αποδοτική, όμως, αξιοποίηση των πυρήνων φαίνεται ότι επιτυγχάνεται με την γενική χρονοδρομολόγηση των εργασιών (global task scheduling), όπου μπορούν να χρησιμοποιηθούν ταυτόχρονα όλοι οι πυρήνες του συστήματος. Ο χρονοδρομολογητής είναι ενήμερος για τις επιδόσεις και τη χωρητικότητα κάθε πυρήνα και δρομολογεί δυναμικά τις εργασίες. Οι αχρησιμοποίητοι πυρήνες απενεργοποιούνται, ενώ αν όλοι οι πυρήνες ενός συμπλέγματος δεν χρησιμοποιούνται, τότε και εκείνο με τη σειρά του απενεργοποιείται.

## 2.5 Κίνητρο Εργασίας

Η γενική χρονοδρομολόγηση εργασιών εμφανίζει σημαντικά πλεονεκτήματα και γι' αυτόν το λόγο επιλέγεται. Κατ' αρχάς επιτρέπει τον λεπτομερή έλεγχο των εργασιών που μετακινούνται μεταξύ των πυρήνων. Είναι εφικτό να χρησιμοποιούνται οι big πυρήνες μόνο για τις υπολογιστικά απαιτητικές εργασίες, ενώ οι little πυρήνες να εξυπηρετούν αποκλειστικά εφαρμογές παρασκηνίου, αλλά και μη απαιτητικές εφαρμογές. Επιπλέον όταν κάθε πυρήνας αντιμετωπίζεται σαν ξεχωριστή οντότητα είναι πιο εύκολο να πραγματοποιηθεί η χρονο-

δρομολόγηση και να καθοριστεί η συχνότητα του επεξεργαστή.

Η συγκεκριμένη προσέγγιση χρονοδρομολόγησης αποτελεί τη μόνη μέθοδο που εκμεταλλεύεται στο έπακρο την υπολογιστική ισχύ, δηλαδή μπορούν να λειτουργούν όλοι οι πυρήνες ταυτόχρονα. Ακόμα, υποστηρίζει με μεγάλη ευκολία μη συμμετρικά συστήματα, όπου ο αριθμός των big πυρήνων δεν είναι ίδιος με αυτόν των little. Οποιοσδήποτε αριθμός πυρήνων κάθε στιγμή μπορεί να είναι ενεργός, ανάλογα με τις ανάγκες του συστήματος. Για όλους αυτούς τους λόγους τα οφέλη της global task χρονοδρομολόγησης αντισταθμίζουν με το παραπάνω τις δυσκολίες υλοποίησής της.

Στην παρούσα διπλωματική εργασία μοντελοποιήσαμε, με χρήση μοντέλων μηχανικής μάθησης, ποιο είδος πυρήνα θα επιλέγεται για την εκτέλεση μίας μονονηματικής εφαρμογής από τις σουίτες benchmark προγραμμάτων Spec2006 και Parsec. Μοντελοποίηση των προγραμμάτων πραγματοποιήσαμε με χρήση των συντελεστών EDP (Energy Delay Product), $ED^2P$ (Energy Delay Delay Product). Επιπλέον προσπαθήσαμε να μοντελοποιήσουμε την συμπεριφορά των προγραμμάτων όταν εκτελούνται παράλληλα με άλλο πρόγραμμα στο ίδιο σύμπλεγμα πυρήνων. Στην μοντελοποίηση που πραγματοποιήσαμε δεν μας απασχόλησε άμεσα ο τρόπος διάταξης των πυρήνων. Όμως, η μοντελοποίηση μας έχει σκοπό να υποστηρίξει global task schedulers. Με βάση το συγκεκριμένο είδος χρονοδρομολόγησης έχει ξεκινήσει να "χτίζεται" η μοντελοποίησή μας. Σε μελλοντική επέκταση της παρούσας διπλωματικής θα θέλαμε να μπορούμε να προβλέπουμε τον κατάλληλο συνδυασμό πυρήνα-αριθμό threads-συχνότητας πυρήνων για κάθε εφαρμογή, ώστε να βελτιστοποιείται η τιμή του συντελεστή EDP ή του $ED^2P$. Με αυτήν τη μεθοδολογία θα μπορέσουμε να εκμεταλλευτούμε βέλτιστα τις δυνατότητες του ετερογενούς συστήματος.

# Κεφάλαιο 3

# Μεθοδολογία

## 3.1 Μηχάνημα μετρήσεων

Το μηχάνημα που πραγματοποιήθηκαν οι μετρήσεις ήταν ένα Exynos 7420, το οποίο περιέχει οκταπύρηνους επεξεργαστές ARM big.LITTLE. Κάθε επεξεργαστής έχει 4 big Cortex-A57 πυρήνες και 4 little Cortex-A53 πυρήνες. Αναλυτικά τα χαρακτηριστικά των πυρήνων επεξεργασίας είναι:

Ο Πίνακας 3.1 παρουσιάζει τα σχετικά χαρακτηριστικά. A53 cpu frequency: 400 Mhz-1500Mhz. Κάθε πυρήνας A53 διαθέτει μία ιδιωτική L1-Instruction cache και μία ιδιωτική L1-Data cache. Οι 4 little πυρήνες μοιράζονται την L2 cache.

Ο Πίνακας 3.2 παρουσιάζει τα σχετικά χαρακτηριστικά. A57 cpu frequency: 800 MHz-2100MHz. Κάθε πυρήνας A57 διαθέτει μία ιδιωτική L1-Instruction cache και μία ιδιωτική L1-Data cache. Οι 4 big πυρήνες μοιράζονται την L2 cache.

### 3.1.1 ARM A53 Cores (LITTLE)

Ο επεξεργαστής in-order Cortex-A53 υποστηρίζει την αρχιτεκτονική συνόλου εντολών (instruction set architecture) ARMv8. Μπορεί να εκτελέσει μέχρι και δύο εντολές ταυτόχρονα (issue width), ενώ το μήκος του pipeline του ισούται με οκτώ. Ο επεξεργαστής εμφανίζει τρεις λογικές μονάδες για πράξεις ακεραίων, εκ των οποίων οι δύο πραγματοποιούν πράξεις πρόσθεσης και η εναπομείνουσα πράξεις πολλαπλασιασμού. Επιπλέον κάθε πυρήνας Cortex-A53 περιέχει μία λογική μονάδα για εντολές load-store, μία για branch εντολές και μία μονάδα για 64 bit πράξεις floating-point ή SIMD. Τα μεγέθη των L1 caches που χρησιμοποιούνται στους συγκεκριμένους πυρήνες κυμαίνονται από 8KB-64KB τόσο για L1 instruction caches όσο και για L1 data caches. Τα μεγέθη των L2 caches στους επεξεργαστικούς πυρήνες A53 κυμαίνονται από

| **L1-Instruction cache** | |
|---|---|
| Line size | 64B |
| Associativity | 2 |
| Cache Size | 32KB |
| **L1-Data cache** | |
| Line size | 64B |
| Associativity | 4 |
| Cache Size | 32KB |
| **L2 cache** | |
| Line size | 64B |
| Associativity | 16 |
| Cache Size | 256KB |

Πίνακας 3.1: Χαρακτηριστικά μνημών caches Cortex-A53.

| **L1-Instruction cache** | |
|---|---|
| Line size | 64B |
| Associativity | 3 |
| Cache Size | 48KB |
| **L1-Data cache** | |
| Line size | 64B |
| Associativity | 2 |
| Cache Size | 32KB |
| **L2 cache** | |
| Line size | 64B |
| Associativity | 16 |
| Cache Size | 2MB |

Πίνακας 3.2: Χαρακτηριστικά μνημών caches Cortex-A57.

128KB έως 2MB. Στο δικό μας σύστημα, η L1 και η L2 cache των A53 cores έχουν μέγεθος 32KB και 256KB, αντίστοιχα.

### 3.1.2 ARM A57 Cores (big)

Ο επεξεργαστής out-of-order Cortex-A57 υποστηρίζει και αυτός την αρχιτεκτονική συνόλου εντολών (instruction set architecture) ARMv8. Μπορεί να εκτελέσει μέχρι και τρεις εντολές ταυτόχρονα (issue width), ενώ το pipeline του περιέχει δεκαοκτώ στάδια. Ο επεξεργαστής εμφανίζει τρεις λογικές μονάδες για πράξεις ακεραίων, εκ των οποίων οι δύο πραγματοποιούν πράξεις πρόσθεσης και η τρίτη πράξεις πολλαπλασιασμού. Ακόμα κάθε πυρήνας Cortex-A57 περιέχει μία λογική μονάδα για εντολές load και μία ξεχωριστή για εντολές store, μία για branch εντολές και δύο μονάδες εκτέλεσης floating-point ή SIMD εντολών έως 128 bit. Σημειώνουμε ότι το branch misprediction penalty παρουσιάζει stalls δεκαπέντε κύκλων. Τα μεγέθη των L1 caches που χρησιμοποιούνται στα συγκεκριμένα είδη πυρήνων είναι 48KB στις L1 instruction caches και 32KB στις L1 data caches. Τέλος, τα μεγέθη των L2 caches κυμαίνονται από 512KB έως 2MB. Στο δικό μας σύστημα, η L1 και η L2 cache των A57 cores έχουν μέγεθος 32KB και 2MB, αντίστοιχα.

## 3.2 Benchmarks

Με στόχο την εξαγωγή συμπερασμάτων για τη συμπεριφορά των επεξεργαστών ARM big.LITTLE και ειδικότερα των πυρήνων Cortex-A53 και Cortex-A57 χρησιμοποιήσαμε διάφορα μετροπρογράμματα ή αλλιώς benchmarks. Η πρώτη ομάδα προγραμμάτων που χρησιμοποιήσαμε ήταν τα προγράμματα Spec 2006. Τα προγράμματα Spec2006 θεωρούνται compute intensive και χωρίζονται σε δύο κατηγορίες τα integer και τα floating-point προγράμματα. Όλα τα προγράμματα Spec2006 εκτελούνται με ένα thread, είναι δηλαδή σειριακά, και εστιάζουν στην ανάλυση της επίδοσης των επεξεργαστικών πυρήνων, της μνήμης του επεξεργαστή και του μεταγλωττιστή των προγραμμάτων.

Η δεύτερη ομάδα benchmarks που χρησιμοποιήσαμε είναι τα Parsec με στόχο την εξαγωγή επιπρόσθετων συμπερασμάτων. Τα Parsec benchmarks είναι multithreaded εφαρμογές και χρησιμοποιούνται για την αποτίμηση των δυνατοτήτων πολυπύρηνων εφαρμογών. Τα προγράμματα Parsec περιέχουν πολλές κατηγορίες προγραμμάτων και θεωρούνται αντιπροσωπευτικά του σύγχρονου επιστημονικού κόσμου.

Στην παρούσα διπλωματική μετρήσαμε τους χρόνους εκτέλεσης των προγραμμάτων στα δύο είδη πυρήνων, το instruction mix τους, τα L1 και L2 cache misses τους και τη συμπεριφορά τους κατά την συνεκτέλεσή τους με το

stress πρόγραμμα. Με βάση τις τιμές τους μπορούμε να εξάγουμε σημαντικά συμπεράσματα για τη συμπεριφορά των προγραμμάτων. Επιπλέον, με τη χρήση των Parsec προγραμμάτων εστιάζουμε σε νέες παραμέτρους, πέρα από εκείνες που εστιάζουμε στα Spec2006 προγράαμματα. Αυτές αφορούν τη συμπεριφορά των threads κατά την παραλληλοποίηση και θα αναλυθούν στη συνέχεια.

## 3.3 Κατηγοριοποίηση εντολών

Κατ' αρχάς για τις μετρήσεις χρησιμοποιήσαμε το binary tool mambo [8] για ARM επεξεργαστές με armv8 αρχιτεκτονική. Το mambo είναι ένα Dynamic Binary Modification εργαλείο για ARM αρχιτεκτονικές. Πρέπει να αναφέρουμε ότι Dynamic Binary Modification είναι μία τεχνική σε επίπεδο λογισμικού, μέσω της οποίας τροποποιούνται εφαρμογές σε πραγματικό χρόνο. Εφαρμόζεται πάνω στον πηγαίο κώδικα μηχανής και προκαλεί σχετικά μικρή καθυστέρηση στην εκτέλεση του προγράμματος. Για να πραγματοποιήσουμε μετρήσεις πάνω στο instruction mix των προγραμμάτων, πραγματοποιήσαμε σημαντικές τροποποιήσεις πάνω στον πηγαίο κώδικα του προγράμματος mambo. Με βάση τη δική μας κατηγοριοποίηση, το binary tool mambo αναγνωρίζει τις παρακάτω κατηγορίες εντολών:

Οι load, store εντολές είναι οι εξής:
A64_LDP_STP
A64_LDR_STR_UNSIGNED_IMMED
A64_LDR_STR_IMMED
A64_LDR_LIT
A64_LDR_STR_REG
A64_LDX_STX
A64_LDX_STX_MULTIPLE
A64_LDX_STX_MULTIPLE_POST
A64_LDX_STX_SINGLE
A64_LDX_STX_SINGLE_POST

Οι integer εντολές είναι οι εξής:
A64_ADD_SUB_EXT_REG
A64_ADD_SUB_SHIFT_REG
A64_ADC_SBC
A64_ADD_SUB_IMMED
A64_LOGICAL_IMMED

Οι floating-point εντολές είναι οι εξής:

A64_FCMP
A64_FCCMP
A64_FCSEL
A64_FLOAT_REG1
A64_FLOAT_REG2
A64_FLOAT_REG3
A64_FMOV_IMMED
A64_FLOAT_CVT_FIXED
A64_FLOAT_CVT_INT
64_DATA_PROC_REG2

Οι simd εντολές είναι οι εξής:
A64_SIMD_ACROSS_LANE
A64_SIMD_COPY
A64_SIMD_EXTRACT
A64_SIMD_MODIFIED_IMMED
A64_SIMD_PERMUTE
A64_SIMD_SCALAR_COPY
A64_SIMD_SCALAR_PAIRWISE
A64_SIMD_SCALAR_SHIFT_IMMED
A64_SIMD_SCALAR_THREE_DIFF
A64_SIMD_SCALAR_THREE_SAME
A64_SIMD_SHIFT_IMMED
A64_SIMD_TABLE_LOOKUP
A64_SIMD_THREE_DIFF
A64_SIMD_THREE_SAME
A64_SIMD_SCALAR_TWO_REG
A64_SIMD_SCALAR_X_INDEXED
A64_SIMD_TWO_REG
A64_SIMD_X_INDEXED

Οι branch εντολές είναι οι εξής:
A64_B_COND
A64_CBZ_CBNZ
A64_TBZ_TBNZ
A64_B_BL
A64_BR
A64_BLR
A64_RET

Οι other εντολές είναι οι εξής:

A64_DATA_PROC_REG1
A64_CCMP_CCMN_IMMED
A64_CCMP_CCMN_REG
A64_COND_SELECT
A64_LOGICAL_REG
A64_DATA_PROC_REG3
A64_SVC
A64_HVC
A64_BRK
A64_SYS
A64_MRS_MSR_REG
A64_HINT
A64_CLREX
A64_DSB
A64_DMB
A64_ISB
A64_BFM
A64_ADR
A64_EXTR
A64_MOV_WIDE

Οι κρυπτογραφικές εντολές είναι οι εξής:
A64_CRYPTO_AES
A64_CRYPTO_SHA_REG3
A64_CRYPTO_SHA_REG2

## 3.4 Ενεργειακή κατανάλωση επεξεργαστών

Οι επεξεργαστές είναι πολύπλοκα κυκλώματα, τα οποία αποτελούνται από transistors. Τα τρανζίστορ μπορούν να βρίσκονται αποκλειστικά σε δύο καταστάσεις on και off. Οι λογικές πύλες με διάφορους αριθμούς εισόδων και εξόδων αποτελούνται από κατάλληλους συνδυασμούς τρανζίστορ. Κάθε τρανζίστορ χαρακτηρίζεται από τη συμφυή χωρητικότητά του C. Όταν αλλάζει κατάσταση το τρανζίστορ φορτίζονται ή εκφορτίζονται οι πυκνωτές του ανάλογα με την αλλαγή κατάστασης. Υπάρχουν δύο τρόποι με τους οποίους καταναλώνουν ενέργεια τα τρανζίστορ του επεξεργαστή.

Η στατική κατανάλωση ενέργειας αποτελεί την πρώτη κατηγορία. Ο συγκεκριμένος τύπος κατανάλωσης ενέργειας εξαρτάται από την παρεχόμενη τάση και από χαρακτηριστικά των τρανζίστορ όπως το μέγεθος των λογικών πυλών

και τα υλικά τους. Αυτό σημαίνει ότι ο συγκεκριμένος τύπος κατανάλωσης ενέργειας αποτελεί ένα συμφυές χαρακτηριστικό του κυκλώματος. Ο τύπος της δίνεται από την παρακάτω σχέση:

$$P_{static} = m * V$$

όπου m σταθερά και V η τάση τροφοδοσίας

Η δυναμική κατανάλωση ενέργειας αποτελεί τη δεύτερη κατηγορία ενεργειακής κατανάλωσης των τρανζίστορ και χωρίζεται σε δύο υποκατηγορίες. Η πρώτη υποκατηγορία δυναμικής κατανάλωσης ενέργειας παρατηρείται όταν αλλάζει κατάσταση ένα τρανζίστορ, με αποτέλεσμα να υπάρχει μερική φόρτιση ή εκφόρτιση. Εξαρτάται επίσης από τη συχνότητα λειτουργίας, από τη χωρητικότητα του τρανζίστορ, από την τάση τροφοδοσίας και από ένα συντελεστή δραστηριότητας α. Ο θεωρητικός τύπος της δυναμικής κατανάλωσης ενέργειας ενός τρανζίστορ είναι ο εξής:

$$P_{transistion} = (1/2) * C * V^2 * f * \alpha$$

Παρατηρούμε μία εξίσωση με θεμελιώδη μεγέθη ανάλογα ή τετραγωνικά ανάλογα σε σχέση με την κατανάλωση ενέργειας. Ο συντελεστής δραστηριότητας α περιγράφει το ποσοστό αλλαγής κατάστασης των τρανζίστορ. Όταν δεν υπάρχει εναλλαγή της κατάστασης ενός τρανζίστορ, τότε δεν υπάρχει κατανάλωση δυναμικής ενέργειας. Επιπλέον, όταν φτάνει σε υψηλά επίπεδα η συχνότητα λειτουργίας του επεξεργαστή πρέπει να αυξάνεται και η τάση τροφοδοσίας του, ώστε να διασφαλιστεί η σωστή λειτουργία.

Η παραπάνω εξίσωση χρησιμοποιείται και για τον υπολογισμό της δυναμικής κατανάλωσης ενέργειας ενός επεξεργαστή. Παρόλα αυτά τη χωρητικότητα του τσιπ, καθώς και το συντελεστή δραστηριότητάς του είναι δύσκολο να τα συμπεράνουμε. Γι' αυτό το λόγο δεν μπορούμε να τη χρησιμοποιήσουμε για να εξάγουμε άμεσα την ενεργειακή κατανάλωση ενός επεξεργαστή. Παρόλα θεωρούμε πολύ σημαντική τη συγκεκριμένη σχέση. Μας δείχνει ότι η κατανάλωση ενέργειας των πυρήνων έχει αναλογική σχέση με τη συχνότητα λειτουργίας τους και σχέση ανάλογη του τετραγώνου της τάσης τροφοδοσίας.

Υπάρχει και δεύτερη υποκατηγορία δυναμικής κατανάλωσης ενέργειας και ονομάζεται ενέργεια μικρών κυκλωμάτων. Όταν άγουν ένα ή περισσότερα τρανζίστορ ταυτόχρονα μπορεί να δημιουργηθούν μικρά κυκλώματα αγωγής. Η ενεργειακή τους κατανάλωση εξαρτάται από τη συχνότητα λειτουργίας. Το πρόβλημα είναι έντονο όταν αυξάνεται η συχνότητα λειτουργίας του επεξεργαστή. Ο τύπος που περιγράφει το συγκεκριμένο είδος κατανάλωσης ενέργειας είναι:

$$P_{short-circuit} = a * E_{short-circuit} * f$$

όπου $E_{short-circuit}$ είναι η κατανάλωση ενέργειας ενός μόνο κυκλώματος, α ο συντελεστής ενεργοποίησης, και f η συχνότητα λειτουργίας του επεξεργαστή.

Άρα, μπορούμε να πούμε ότι η προσεγγιστική κατανάλωση ενέργειας ενός επεξεργαστή είναι:

$$P_{cpu} = P_{static} + P_{transition} + P_{short-circuit}$$

### 3.4.1 Ενεργειακές μετρήσεις

Είναι εμφανές ότι η ταχύτερη εκτέλεση ενός thread επιτυγχάνεται όταν εκείνο τρέχει στο μεγάλο πυρήνα με τη μέγιστη δυνατή συχνότητα λειτουργίας, ενώ η ελάχιστη ενεργειακή κατανάλωση επιτυγχάνεται στον μικρό πυρήνα με την ελάχιστη δυνατή συχνότητα λειτουργίας. Στην παρούσα διπλωματική εργασία, πραγματοποιήσαμε τις μετρήσεις μας στις μέγιστες συχνότητες λειτουργίας του κάθε είδους πυρήνα. Συγκεκριμένα, κατά την εκτέλεση των προγραμμάτων, οι big πυρήνες λειτουργούσαν με 2,1 GHz συχνότητα λειτουργίας και οι little με 1,5 GHz συχνότητα λειτουργίας. Επειδή τα προγράμματα τα εκτελέσαμε στις μέγιστες συχνότητες λειτουργίας στα δύο είδη πυρήνων, ο χρόνος εκτέλεσης των προγραμμάτων στους little πυρήνες είναι πολύ μεγαλύτερος σε σχέση με τον αντίστοιχο στους big. Γι' αυτόν το λόγο σε πολλά benchmarks η ενεργειακή κατανάλωση στον big πυρήνα είναι μικρότερη σε σχέση με εκείνη στον little πυρήνα.

Αξίζει να σημειωθεί πως στην έρευνά μας δεν εξετάζουμε τα προγράμματα από πλευράς κατανάλωσης ενέργειας ανά εντολή. Όμως, το instruction mix και τα cache misses μας δίνουν πληροφορίες για την ενεργειακή συμπεριφορά των προγραμμάτων, τις οποίες και αξιοποιούμε σαν εισόδους στα μοντέλα εκπαίδευσης. Επίσης, επειδή λαμβάναμε την ενεργειακή κατανάλωση από ένα σύμπλεγμα Exynos συστημάτων, δεν είχαμε τη δυνατότητα να μετρήσουμε την ακριβή ενεργειακή κατανάλωση των προγραμμάτων. Γι' αυτόν το λόγο θέσαμε ως σταθερά την ενεργειακή κατανάλωση συστήματος που μετράται όταν δεν εκτελείται κάποια εφαρμογή (idle state), και την αφαιρούμε από την εκάστοτε μέτρηση ενέργειας που πραγματοποιούμε όταν εκτελείται κάποιο benchmark. Συνεπώς, η μετρούμενη ενεργειακή κατανάλωση ανά πρόγραμμα είναι αρκετά προσεγγιστική.

# Κεφάλαιο 4

# Ανάλυση Spec2006 Benchmarks

## 4.1 Εισαγωγή

Σε αυτό το κεφάλαιο αναλύσαμε τα Spec2006 Benchmarks [10]. Πραγματοποιήσαμε μετρήσεις σχετικά με τους χρόνους εκτέλεσης των προγραμμάτων, το instruction mix τους, τα cache misses τους, τη συμπεριφορά τους όταν συνεκτελούνται ταυτόχρονα με το stress πρόγραμμα, αλλά και σχετικά με την ενεργειακή τους κατανάλωση. Με βάση τα πειραματικά μας αποτελέσματα, αλλά και άλλες έρευνες, επιχειρήσαμε να αναλύσουμε ποιοι παράγοντες επηρεάζουν την απόδοσή τους και την ενεργειακή τους κατανάλωση στα δύο είδη πυρήνων. Τις μετρήσεις τις πραγματοποιήσαμε στη μέγιστη συχνότητα λειτουργίας του κάθε πυρήνα, με βάση τον default χρονοδρομολογητή του linux. Καθ' όλη τη διάρκεια των μετρήσεων ο little πυρήνας έτρεχε στη συχνότητα 1,5 GHz, ενώ big στη συχνότητα 2,1 GHz. Πριν την ανάλυση των Spec 2006 μετρήσεων παρέχουμε πληροφορίες για τα benchmarks που χρησιμοποιήσαμε από τη συγκεκριμένη σουίτα στον πίνακα 4.1.

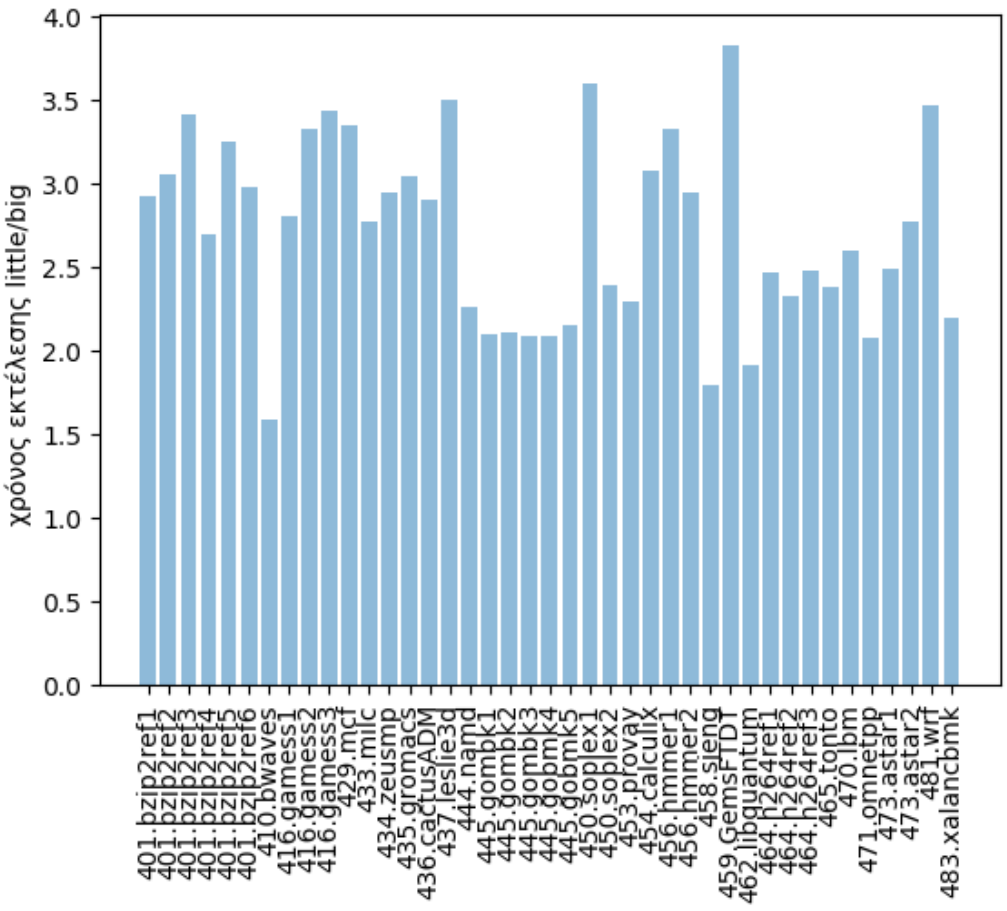
## 4.2 Αντίκτυπος της ετερογένειας big.LITTLE στην απόδοση

Σε πρώτη φάση πραγματοποιήσαμε τις χρονικές των Spec2006 benchmarks. Το κάθε benchmark της κατηγορίας το εκτελέσαμε τόσο σε big όσο και σε little πυρήνα. Επιπλέον εκτελέσαμε τα προγράμματα με reference και με test input sets. Στις εικόνες 4.1 και 4.2 παρουσιάζονται τα αποτελέσματα των χρονικών μας μετρήσεων. Οφείλουμε να σημειώσουμε ότι στα διαγράμματα

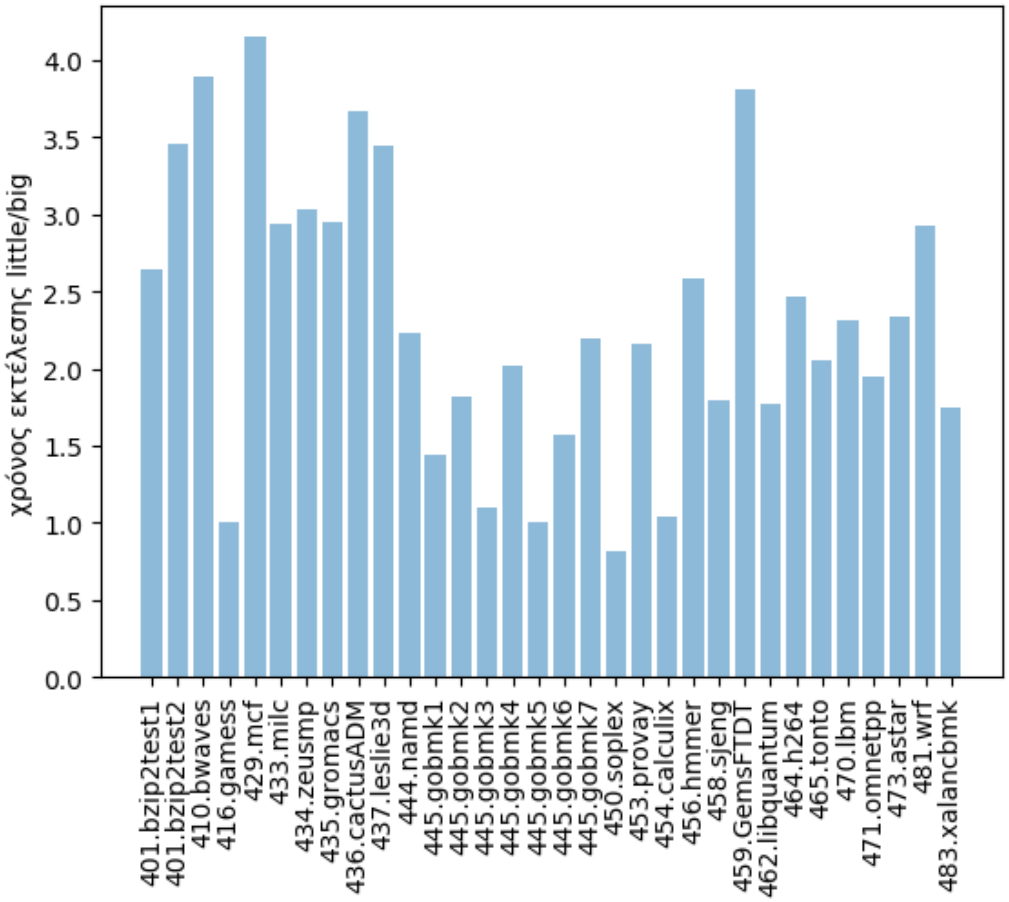
| Ονομασία benchmark | Λειτουργία | Γλώσσα Προγ. | Είδος |
|---|---|---|---|
| 401.bzip2 | Compression | C | int |
| 410.bwaves | Fluid Dynamics | Fortran | fp |
| 416.gamess | Quantum Chemistry | Fortran | fp |
| 429.mcf | Compinatorational Optimazation | C | int |
| 433.milc | Physics: Quantum Chromodynamics | C | fp |
| 434.zeusmp | Physics/CFD | Fortran | fp |
| 435.gromacs | Biochemistry/Molecular Dynamics | Fortran | fp |
| 436.cactusADM | Physics/General Relativity | C/Fortran | fp |
| 437.leslie3d | Fluid Dynamics | Fortran | fp |
| 444.namd | Biology /Molecular Dynamics | C++ | fp |
| 445.gobmk | Artificial Intelligence | C | int |
| 450.soplex | Linear Programming, Optimization | C++ | fp |
| 453.povray | Image Ray-tracing | C++ | fp |
| 454.calculix | Structural Mechanics | C++ | fp |
| 456.hmmer | Search Gene Sequence | C | int |
| 458.sjeng | Artificial Inteligence | C | fp |
| 459.GemsFDTD | Quantum Chemistry | Fortran | int |
| 462.libquantum | Physics: Quantum Computing | C | int |
| 464.h264ref | Video Compression | C | int |
| 465.tonto | Quantum Chemistry | Fortran | fp |
| 470.lbm | Fluid Dynamics | C/Fortran | fp |
| 471.omnetpp | Discrete Event Simulation | C++ | int |
| 473.astar | Path-finding Algorithms | C++ | int |
| 481.wrf | Weather Prediction | C/Fortran | fp |
| 483.xalancbmk | XML Processing | C++ | int |

Πίνακας 4.1: Χαρακτηριστικά των Spec2006 benchmarks.

αυτά συγκρίνουμε το χρόνο εκτέλεσης του κάθε προγράμματος στα δύο είδη πυρήνων σε μορφή ποσοστού. Πριν προχωρήσουμε στην ανάλυση Spec2006 Benchmarks κρίναμε απαραίτητο να παραθέσουμε τις χρονικές μετρήσεις των προγραμμάτων.




Εικόνα 4.1: Αποτελέσματα χρόνων εκτέλεσης των Spec2006 benchmarks με χρήση reference input sets.




Εικόνα 4.2: Αποτελέσματα χρόνων εκτέλεσης των Spec2006 benchmarks με χρήση test input sets.

Κατ' αρχάς θεωρούμε a το συντελεστή χρόνου εκτέλεσης benchmark στον little πυρήνα προς το χρόνο εκτέλεσης του ίδιου benchmark στον big πυρήνα. Χρησιμοποιούμε το συντελεστή a για να κατηγοριοποιήσουμε τα Spec2006 προγράμματα. Έχουμε κατατάξει τα benchmarks με βάση τα reference input sets, τα οποία είναι πιο αντιπροσωπευτικά. Χωρίσαμε τα Spec2006 προγράμματα σε τρεις κατηγορίες με βάση το συντελεστή a. Θεωρήσαμε ως προγράμματα με υψηλό συντελεστή a εκείνα που εμφάνιζαν τιμή a άνω του 3. Στην κατηγορία προγράμματα με μεσαίο συντελεστή a θεωρούμε τα προγράμματα με συντελεστή στο διάστημα από 2,2 έως 3. Τα υπόλοιπα προγράμματα τα κατατάξαμε στην κατηγορία προγραμμάτων με χαμηλό συντελεστή a.

### 4.2.1 Προγράμματα με υψηλό συντελεστή επίδοσης big.LITTLE

. Παρατηρούμε ότι, κατά βάση, computive intensive εφαρμογές, οπως fluid dynamics, quantum dynamics, quantum chemistry εφαρμογές που εμφανίζουν μεγάλο αριθμό floating-point και integer εντολών ανήκουν σε αυτήν την κατηγορία. Συγκεκριμένα οι εφαρμογές 401.bzip2, 416.gamess, 429.mcf, 435.gromacs, 437.leslie3d, 450.soplex, 454.calculix, 459.GemsFDT, 481.wrf εμφανίζουν τον υψηλότερο συντελεστή a.

Αναλυτικά, το benchmark bzip2 ασχολείται με τη συμπίεση αρχείων. Δέχεται ως είσοδο 6 στοιχεία, δύο JPEG εικόνες, ένα δυαδικό πρόγραμμα, έναν πηγαίο κώδικα σε tar αρχείο, ένα HTML αρχείο και ένα συνδυαστικό αρχείο που περιέχει υψηλά συμπιέσιμα και μη υψηλά συμπιέσιμα αρχεία. Με βάση τις μετρήσεις μας με χρήση reference εισόδων, βλέπουμε ότι το πρόγραμμα εμφανίζει σημαντικό ποσοστό integer εντολών. Επιπλέον εμφανίζει υψηλό ποσοστό L2 cache misses στους little πυρήνες, άνω του 40%, ενώ στον big πυρήνα εμφανίζει πολύ χαμηλό ποσοστό L2 cache misses. Αυτός είναι, ίσως, ο πιο σημαντικός λόγος για την πολύ ταχύτερη εκτέλεση του benchmark στους big πυρήνες.

Η εφαρμογή 416.gamess πραγματοποιεί υπολογισμούς στον τομέα της κβαντικής χημείας και ανήκει στην κατηγορία των floating-point προγραμμάτων. Από άλλες έρευνες παρατηρούμε ότι εμφανίζει σημαντικό αριθμό branches εντολών, καθώς και χαμηλό ποσοστό υπολογιστικών εντολών. Βέβαια οι συγκεκριμένες έρευνες αναφέρονται σε επεξεργαστές με διαφορετικό instruction set architecture. Αναφερόμαστε σε άλλες έρευνες, γιατί εμείς λόγω προβλημάτων που αντιμετωπίσαμε με το mambo δεν καταφέραμε να πραγματοποιήσουμε μετρήσεις πάνω στο intruction mix του συγκεκριμένου προγράμματος.

Το 429.mcf είναι πρόγραμμα δρομολόγησης οχημάτων στις δημόσιες συγκοινωνίες. Τα L1 cache misses του είναι υψηλά, άνω του 25% και στα δύο είδη πυρήνων. Με βάση τις μετρήσεις μας με χρήση reference εισόδων, είναι πολύ υψηλά τα L2 cache misses του προγράμματος στον big πυρήνα, άνω του 50%, και αρκετά υψηλότερα είναι εκείνα στον little πυρήνα. Με βάση το instruction mix του προγράμματος με χρήση reference εισόδου, παρατηρούμε ότι εμφανίζει σημαντικό ποσοστό integer και branch εντολών.

Το benchmark 435.gromacs είναι ένα πρόγραμμα προσομοίωσης μοριακών δυνάμεων ανάμεσα σε μία πρωτεΐνη, σε μόρια νερού και σε ιόντα. Το ποσοστό των αριθμητικών πράξεων επί του συνόλου ξεπερνά το 50% στο συγκεκριμένο benchmark, ενώ οι floating-point πράξεις αγγίζουν το 50%. Εμφανίζει ποσοστό 4.89% L2 cache misses στον big πυρήνα και λίγο πάνω από 15% στον little.

Στις floating-point εφαρμογές ανήκει το benchmark 437.leslie3d, το οποίο ασχολείται με την επίλυση φαινομένων ρευστομηχανικής. Λόγω προβλημάτων με το mambo στις μετρήσεις, δε διαθέτουμε πληροφορίες για τα cache misses

και το instruction mix του προγράμματος. Ένα ακόμα floating-point πρόγραμμα είναι το 450.soplex. Επιλύει ένα γραμμικό πρόγραμμα χρησιμοποιώντας τον αλγόριθμο simplex. Θεωρείται υπολογιστικά απαιτητική εφαρμογή με σημαντικό ποσοστό integer εντολών. Όταν χρησιμοποιούμε την πρώτη reference είσοδο το ποσοστό των L2 cache misses είναι 46,01% και 71,98% στον big και στον little πυρήνα αντίστοιχα. Όταν χρησιμοποιούμε την δεύτερη reference είσοδο το ποσοστό των L2 cache misses είναι 63,45% και 67,08% στον big και στον little πυρήνα αντίστοιχα.

Το πρόγραμμα 454.calculix ανήκει στον τομέα της στατικής μηχανικής. Χρησιμοποιεί τη μέθοδο πεπερασμένων στοιχείων για τη μελέτη προβλημάτων στατικής και τα επιλύει. Όταν χρησιμοποιούμε την test είσοδο παρατηρούμε ότι υψηλό ποσοστό integer εντολών και το ποσοστό των L2 cache misses είναι 3.89% και 16.87% στον big και στον little πυρήνα αντίστοιχα.

Στην εφαρμογή 456.hmmer συγκρίνονται αλυσίδες γονιδιώματος και κατατάσσονται με βάση μία συνάρτηση καλύτερης διαλογής. Από τις μετρήσεις μας προκύπτει ότι το συγκεκριμένο πρόγραμμα εμφανίζει σημαντικό αριθμό integer εντολών, τουλάχιστον με βάση την test είσοδό του. Με βάση τις μετρήσεις μας με χρήση reference εισόδων, παρατηρούμε πάρα πολύ υψηλά L2 cache misses, περίπου 70% στο κάθε είδος πυρήνα.

Η βασική λειτουργία του προγράμματος 459.GemsFDT είναι να επιλύει τις εξισώσεις του Maxwell με τη μέθοδο των πεπερασμένων διαφορών στο πεδίο του χρόνου. Με βάση τις μετρήσεις μας με χρήση του reference input set, το πρόγραμμα εμφανίζει άνω του 40% ποσοστό floating-point εντολών και σημαντικό αριθμό integer εντολών. Το ποσοστό των L2 cache misses είναι σχετικά χαμηλό στους big πυρήνες και πολύ υψηλό στους little.

Τέλος, το benchmark 481.wrf είναι ένα πρόγραμμα πρόβλεψης καιρού, κατάλληλο για τις ανάγκες επιχειρήσεων και ερευνητών. Όπως φαίνεται και από τις μετρήσεις με χρήση του test input set του, είναι compute intensive εφαρμογή με υψηλό αριθμό floating-point, integer και simd εντολών. Λόγω προβλημάτων με το mambo στις μετρήσεις δε διαθέτουμε πληροφορίες για τα cache misses του προγράμματος.

Συνοψίζοντας, παρατηρούμε ότι κυρίως floating-point εφαρμογές με μεγάλα ποσοστά εκτέλεσης υπολογιστικών εντολών οδηγούν σε υψηλό συντελεστή a. Οι big πυρήνες A57 διαθέτουν δύο μονάδες εκτέλεσης floating-point εντολών, ενώ οι A53 πυρήνες μόνο μία. Επιπλέον προγράμματα με υψηλό αριθμό εξαρτήσεων μεταξύ των εντολών τους εμφανίζουν καλύτερη συμπεριφορά στους out-of-order big πυρήνες σε σχέση με τους αργούς in-order little πυρήνες. Βέβαια δεν έχουμε ποσοτικοποιήσει τέτοιου είδους μετρικές. Τέλος, τα L2 cache misses παίζουν καθοριστικό ρόλο στην απόδοση ενός προγράμματος. Προγράμματα στα οποία το ποσοστό των L2 cache misses στον little πυρήνα είναι πολύ υψηλότερο σε σχέση με το αντίστοιχο στον big, έχουν περισσότερες

πιθανότητες να εμφανίζουν υψηλό συντελεστή a.

### 4.2.2 Προγράμματα με χαμηλό συντελεστή επίδοσης big.LITTLE

. Αναλύουμε, τώρα, τα benchmarks με το χαμηλότερο συντελεστή a. Παρατηρούμε ότι τα benchmarks με τους χαμηλότερους συντελεστές είναι το 410.bwaves, το 445.gombk, το 458.sjeng, το 462.libquantum και το 471.omnetpp. Το benchmark 410.bwaves επιλύει διαφορικές εξισώσεις με εφαρμογή τα κύματα στο τρισδιάστατο επίπεδο, χρησιμοποιώντας κατάλληλο προσεγγιστικό αλγόριθμο. Εκτελεί σημαντικό, ποσοστιαία, αριθμό integer και floating-point εντολών. Τα L1 cache misses του συγκεκριμένου προγράμματος είναι πάρα πολύ υψηλά, οριακά πάνω από 35%, στα δύο είδη πυρήνων. Τα L2 cache misses του benchmark 410.bwaves θεωρούνται χαμηλά και στα δύο είδη πυρήνων. Τα benchmarks 445.gombk και 458.sjeng είναι εφαρμογές τεχνητής νοημοσύνης. Με βάση τις μετρήσεις μας το πρόγραμμα 445.gombk εμφανίζει μηδενικό ποσοστό L2 cache misses και στα δύο είδη πυρήνων και μεσαίο ποσοστό L1 cache misses, κάτω του 10%, τόσο στον big, όσο και στον little πυρήνα. Όσον αφορά το πρόγραμμα 458.sjeng, παρατηρούμε ότι εμφανίζει υψηλά ποσοστά integer και branches εντολών, τουλάχιστον με βάση το test input set. Με βάση, πάλι, το test input set παρατηρούμε ότι το πρόγραμμα 458.sjeng εμφανίζει χαμηλό ποσοστό L1 και L2 cache misses και στα δύο είδη πυρήνων. Τέλος, τα δύο benchmarks 445.gobmk και 458.sjeng εμφανίζουν σχετικά υψηλά ποσοστά branch misprediction με βάση την έρευνα [18].

Μία βιβλιοθήκη προσομοίωσης κβαντικού υπολογιστή είναι το benchmark 462.libquantum. Όταν χρησιμοποιούμε τη reference είσοδο, το πρόγραμμα εμφανίζει άνω του 40% integer εντολές, μηδενικές floating-point, ενώ παρουσιάζει και σημαντικό αριθμό branches εντολών. Τα L2 cache misses του είναι μηδενικά και στα δύο είδη πυρήνων. Το πρόγραμμα 471.omnetpp προσομοιώνει διακριτά γεγονότα σε ένα δίκτυο Ethernet. Με βάση τις μετρήσεις με χρήση της reference εισόδου το συγκεκριμένο πρόγραμμα έχει υψηλά L2 cache misses στον big πυρήνα και ακόμα υψηλότερα στον little πυρήνα. Εμφανίζει μέτριο ποσοστό integer εντολών και το υψηλότερο ποσοστό simd εντολών, κάτω του 10%.

Παρατηρούμε ότι τα προγράμματα με χαμηλό συντελεστή a περιέχουν σχετικά μικρό αριθμό floating-point εντολών και κάποια από αυτά πρέπει να εμφανίζουν και υψηλό αριθμό branch mispredictions. Εκτός από το 410.bwaves τα υπόλοιπα benchmarks παρουσιάζουν υψηλά ποσοστά πράξεων ακεραίων, όπου απαιτούνται λιγότεροι υπολογιστικοί κύκλοι σε σχέση με τις αντίστοιχες floating-point εντολές. Βασικό χαρακτηριστικό που καθορίζει την επιβράδυνση της εκτέλεσης ενός προγράμματος σε little πυρήνα είναι και τα stalls

λόγω εξαρτήσεων ή και μη διαθέσιμων μονάδων εκτέλεσης εντολών. Επιπλέον παρατηρούμε ότι στα περισσότερα προγράμματα της συγκεκριμένης κατηγορίας δεν παρατηρούνται σημαντικές διαφορές στα L2 cache misses των δύο ειδών πυρήνων.

### 4.2.3 Προγράμματα με κανονικό συντελεστή επίδοσης big.LITTLE

. Τα υπόλοιπα benchmarks παρουσιάζουν ενδιάμεση συμπεριφορά. Ο συντελεστής τους κυμαίνεται από 2,2 έως 3. Σε αυτή την κατηγορία ανήκουν τα benchmarks 433.milc ,434.zeusmp, 436.cactusADM, 444.namd, 453.povray, 464.h264, 465.tonto, 470.lbm, 473.astar, 483.xalancbmk. Κάποια από τα παραπάνω προγράμματα ,ίσως , θα έπρεπε να ανήκουν στην κατηγορία με υψηλό συντελεστή a, αλλά για λόγους απλότητας τα κατηγοριοποιήσαμε έτσι.

Το πρόγραμμα 433.milc προσομοιώνει τη συμπεριφορά στοιχειωδών σωματιδίων. Ο συντελεστής του προγράμματος a είναι σχετικά υψηλός 2,7795 και η εφαρμογή ανήκει στις floating-point. Λόγω σφαλμάτων και σε αυτήν την περίπτωση δεν πραγματοποιήσαμε μετρήσεις με το binary tool mambo πάνω στο instruction mix και στα cache misses του προγράμματος. Το πρόγραμμα 434.zeusmp αποτελεί ένα πρόγραμμα προσομοίωσης αστροφυσικών φαινομένων. Εμφανίζει υψηλό συντελεστή a, με αποτέλεσμα να βρίσκεται οριακά στη συγκεκριμενη κατηγορία. Παρουσιάζει υψηλό αριθμό floating-point εντολών, όταν χρησιμοποιούμε το test input set. Το ποσοστό των L2 cache είναι 22,52% και 27,75% στον big και στον little πυρήνα αντίστοιχα όταν χρησιμοποιούμε το reference input set. Συντελεστή a ίσο με 2,9059 εμφανίζει το πρόγραμμα 436.cactusADM, δηλαδή το συγκεκριμένο benchmark οριακά δεν ανήκει στην πρώτη κατηγορία. Με βάση τις μετρήσεις μας με χρήση της reference εισόδου, τα L2 cache misses του είναι πολύ υψηλά σε big και σε little πυρήνα, ενώ το ποσοστό floating-point εντολών είναι πάνω από το 40% επί του συνόλου.

Το benchmark 444.namd δέχεται ως είσοδο μόρια απολιποπρωτεϊνών και υπολογίζει διάφορα αθροίσματα για τις δυνάμεις που ασκούνται μεταξύ τους. Αναμένονταν κανονικά και το benchmark 444.namd να είχε υψηλό συντελεστή a, λόγω των πολλών υπολογιστικών πράξεων, αλλά πειραματικά δεν αποδεικνύεται. Με βάση τις μετρήσεις μας με χρήση του test input set, το πρόγραμμα εμφανίζει άνω 60% ποσοστό integer και floating-point εντολών μαζί. Τα cache misses του προγράμματος είναι πολύ χαμηλά και στα δύο είδη πυρήνων. Στην κατηγορία της όρασης υπολογιστών ανήκει το πρόγραμμα 453.povray. Εμφανίζει πάνω από 10% floating-point και integer εντολές ξεχωριστά, με βάση τις μετρήσεις μας με χρήση του test input set. Τα L2 cache misses του προγράμματος είναι κάτω του 1% και στα δύο είδη πυρήνων, πάλι με βάση το test

input set.

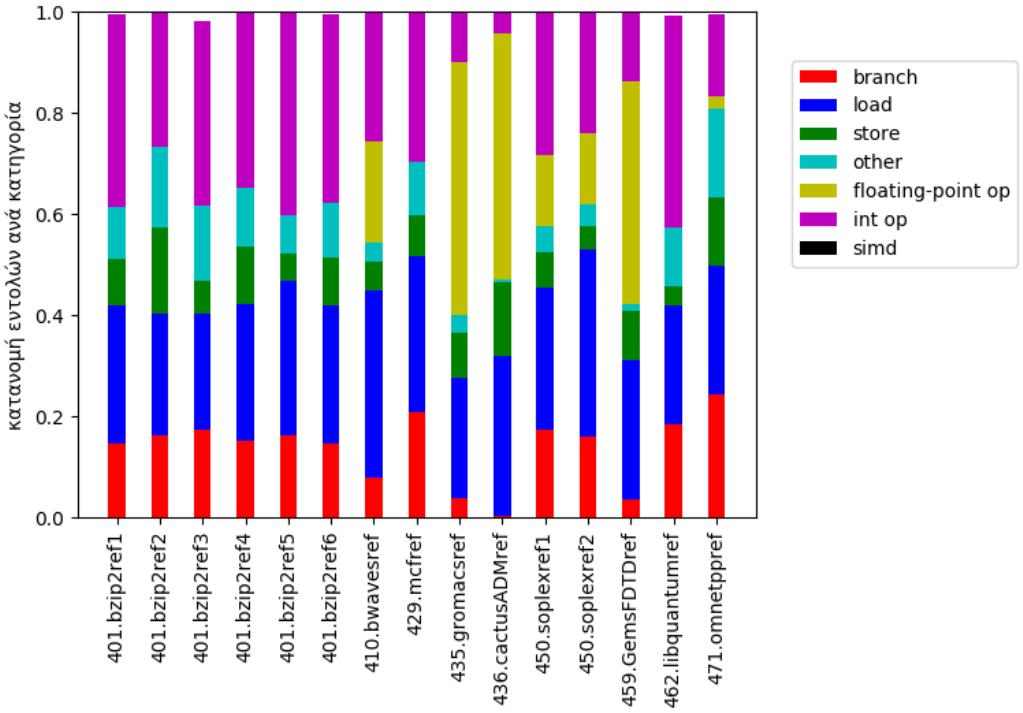
Το πρόγραμμα 464.h264 είναι ένα πρόγραμμα συμπίεσης βίντεο. Λόγω σφαλμάτων κατά την πραγματοποίηση μετρήσεων δεν καταφέραμε να λάβουμε περισσότερες πληροφορίες για το instruction mix και τα cache misses του συγκεκριμένου προγράμματος. Το benchmark 465.tonto ασχολείται με την κβαντική χημεία και είναι γραμμένο σε γλώσσα Fortran 95. Αναμένονταν να εμφανίζει υψηλό συντελεστή a με βάση τα L2 cache misses του στα δύο είδη πυρήνων, αλλά τουλάχιστον πειραματικά δεν επιβεβαιώνεται κάτι τέτοιο. Με βάση τις μετρήσεις μας με χρήση της reference εισόδου, το ποσοστό των L2 cache misses του προγράμματος, όταν "τρέχει" σε big πυρήνα είναι 14,46%, ενώ όταν "τρέχει" σε little αγγίζει το 60%. Ακόμα εμφανίζει υψηλό ποσοστό αριθμητικών εντολών και ιδιαίτερα integer εντολών, όταν χρησιμοποιούμε την test είσοδο. Το πρόγραμμα 470.lbm ανήκει στον τομέα της υπολογιστικής ρευστομηχανικής. Εμφανίζει L2 cache misses κοντά στο 70% και στα δύο είδη πυρήνων. Με βάση το test input set παρατηρούμε ότι το benchmark 470.lbm παρουσιάζει floating-point εντολές σε ποσοστό άνω του 60%.

Το benchmark 473.astar δίνει και εκείνο χαμηλό συντελεστή a, όπως και τα άλλα δύο προγράμματα τεχνητής νοημοσύνης. Το πρόγραμμα προέρχεται από μία συμπληρωματική βιβλιοθήκη εύρεσης δισδιάστατων μονοπατιών και με βάση τις μετρήσεις φαίνεται να παρουσιάζει σημαντικό ποσοστό integer εντολών όταν χρησιμοποιούμε το test input set. Με βάση τα reference input sets, τα L2 cache misses του προγράμματος 473.astar είναι κοντά στο 70% και στα δύο είδη πυρήνων. Τέλος, το πρόγραμμα 483.xalambank μετατρέπει XML αρχεία σε html ή άλλης μορφής αρχεία. Εμφανίζει σημαντικό ποσοστό integer εντολών και τα cache misses του προγράμματος είναι 4,31% και 10,4% στον big και στον little πυρήνα αντίστοιχα, τουλάχιστον όταν χρησιμοποιούμε το test input set.

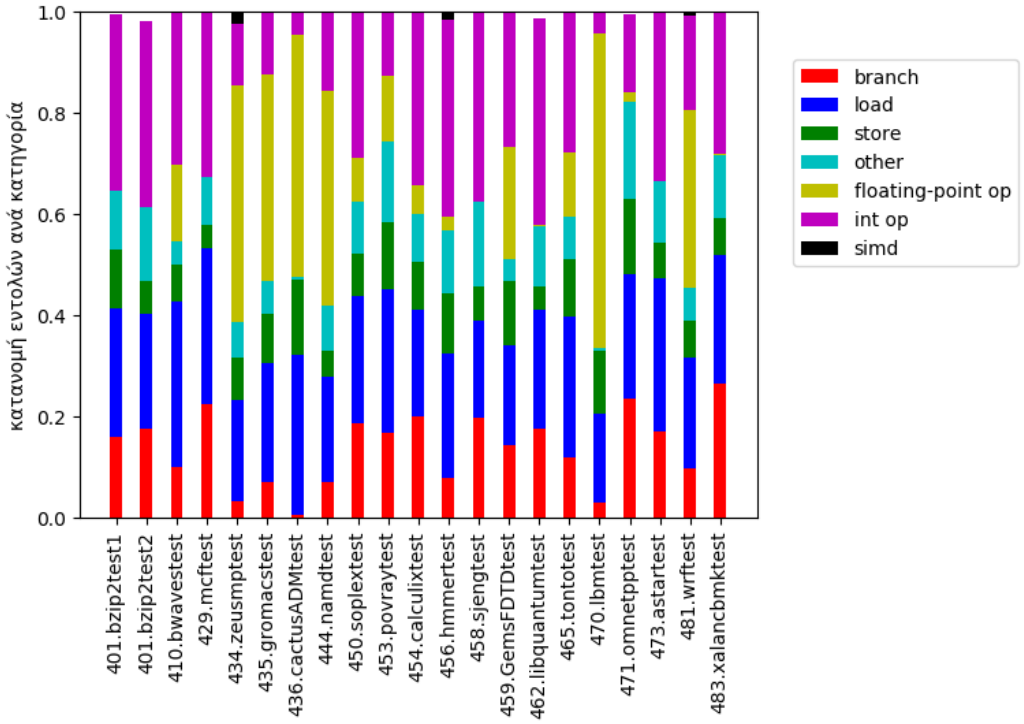
Συμπερασματικά βλέπουμε ότι τα προγράμματα της συγκεκριμένης κατηγορίας εμφανίζουν διαφορετικά χαρακτηριστικά μεταξύ τους. Παρατηρούμε τόσο integer όσο και floating-point εφαρμογές στην κατηγορία εφαρμογών με κανονικό συντελεστή επίδοσης a. Με βάση τα πειραματικά αποτελέσματα πολλά από τα προγράμματα, όπως το 473.astar και το 483.xalambank, σε μία διαφορετικού τύπου κατηγοριοποίηση θα μπορούσαμε να τα κατατάξουμε στην κατηγορία με χαμηλό συντελεστή a. Εμφανίζουν πολλά από τα χαρακτηριστικά των προγραμμάτων χαμηλό συντελεστή a. Αντίθετα το πρόγραμμα 436.cactusADM θα μπορούσαμε να τα κατατάξουμε με βάση το instruction mix του, τα cache misses του και το συντελεστή a στην κατηγορία προγραμμάτων με υψηλό συντελεστή a.

## 4.3 Instruction mix

Αφού περιγράψαμε τις βασικές λειτουργίες των benchamarks, παραθέτουμε στη συνέχεια τα αποτελέσματα από τις επιμέρους μετρήσεις μας. Συγκεκριμένα παραθέτουμε τη συμπεριφορά των δύο επιπέδων της cache και πληροφορίες για το instruction mix των παραπάνω benchmarks. Θα θέλαμε να αναφέρουμε ότι για την πραγματοποίηση των μετρήσεων σε επίπεδο εντολών χρησιμοποιήσαμε το πρόγραμμα mambo binary tool, το οποίο τροποποιήσαμε κατάλληλα. Σε όσα benchamarks δεν υπάρχουν επιμέρους αποτελέσματα είναι λόγω αδυναμίας σωστής εκτέλεσης του mambo με το εκάστοτε benchmark. Στην εικόνα 4.3 παρουσιάζουμε την κατανομή των εντολών για reference input sets και στην εικόνα 4.4 για test input sets.




Εικόνα 4.3: Κατανομή εντολών ανά κατηγορία για τα Spec2006 benchmarks με reference input sets.

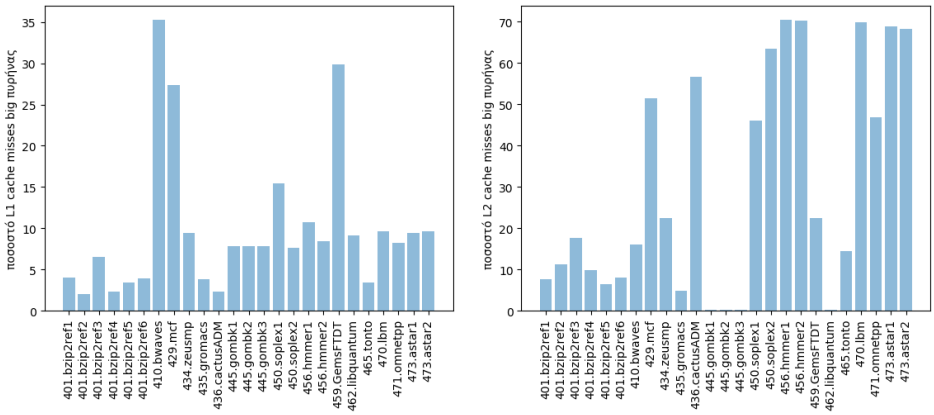



Εικόνα 4.4: Κατανομή εντολών ανά κατηγορία για τα Spec2006 benchmarks με test input sets.

## 4.4 Συμπεριφορά των μνημών caches

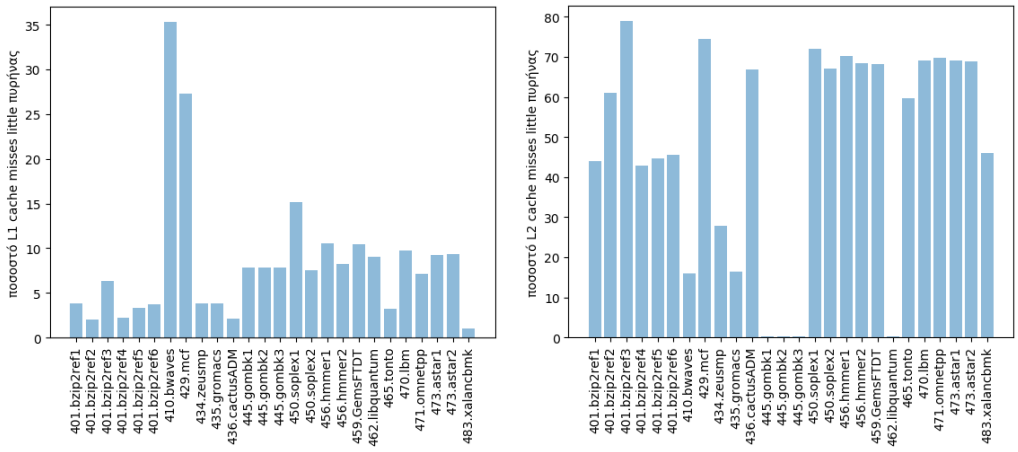
Πραγματοποιήσαμε μετρήσεις πάνω στα cache misses των Spec2006 benchmarks. Ειδικότερα μετρήσαμε τα L1 και L2 cache misses των προγραμμάτων τόσο όταν εκτελέστηκαν σε little όσο και όταν εκτελέστηκαν σε big πυρήνα των big.LITTLE επεξεργαστών. Και εδώ σε όσα benchmarks δεν παρουσιάζουμε επιμέρους αποτελέσματα είναι λόγω σφαλμάτων.

Με βάση τις μετρήσεις μας στις caches, παρατηρούμε ότι τα L2 cache misses είναι σχετικά μικρότερα στους big πυρήνες σε σχέση με τα αντίστοιχα στους little πυρήνες. Αυτήν ακριβώς τη συμπεριφορά αναμέναμε, καθώς η L2 cache του big πυρήνα είναι μεγέθους 2Mbyte, δηλαδή 8 φορές μεγαλύτερη σε σχέση με την L2 cache του little πυρήνα. Αντίθετα, οι big και οι little πυρήνες διαθέτουν ίδιου μεγέθους L1 caches με αποτέλεσμα να μην υπάρχουν σημαντικές αποκλίσεις στα L1 cache misses, μεταξύ των πυρήνων.



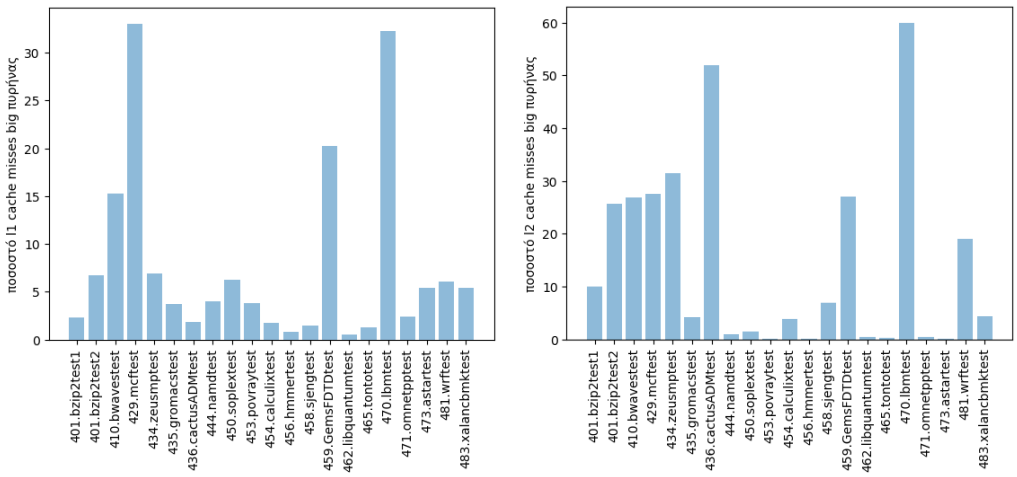

(1) L1 data cache misses. (2) L2 cache misses.

Εικόνα 4.5: Ποσοστό των misses στις L1 και L2 caches του big πυρήνα για τα Spec2006 benchmarks με reference input sets.



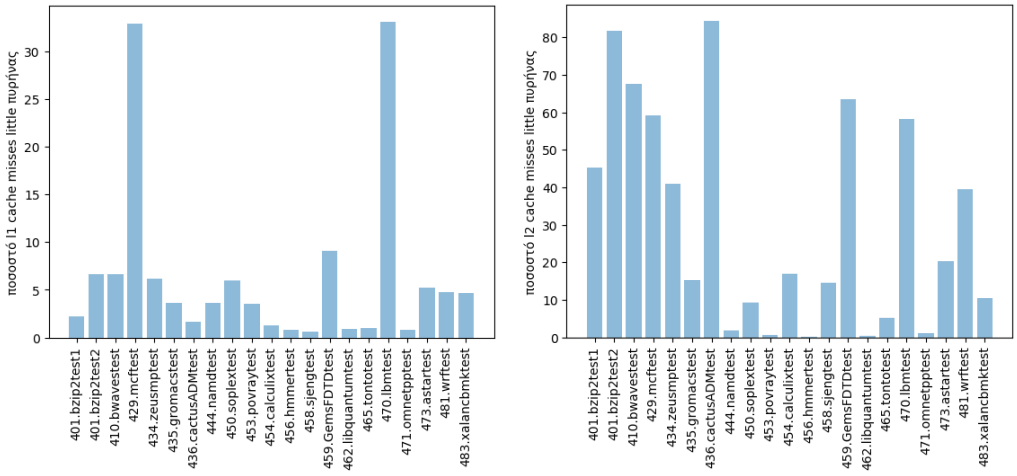

(1) L1 data cache misses. (2) L2 cache misses.

Εικόνα 4.6: Ποσοστό των misses στις L1 και L2 caches του little πυρήνα για τα Spec2006 benchmarks με reference input sets.




(1) L1 data cache misses. (2) L2 cache misses.

Εικόνα 4.7: Ποσοστό των misses στις L1 και L2 caches του big πυρήνα για τα Spec2006 benchmarks με test input sets.

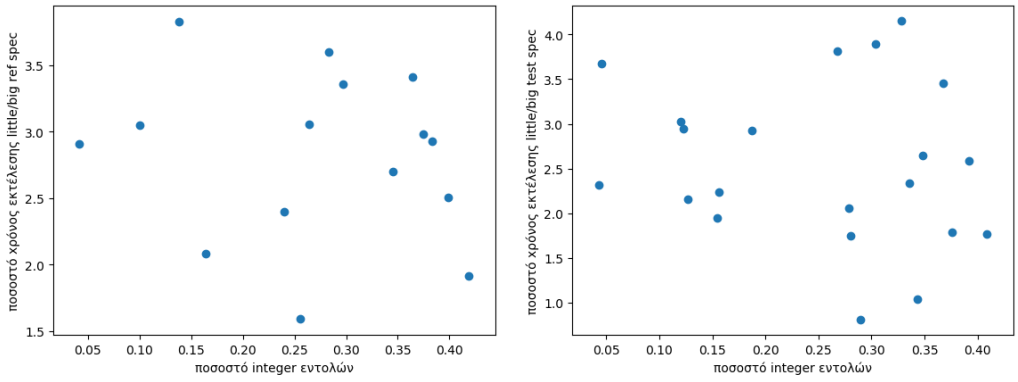



(1) L1 data cache misses. (2) L2 cache misses.

Εικόνα 4.8: Ποσοστό των misses στις L1 και L2 caches του little πυρήνα για τα Spec2006 benchmarks με test input sets.

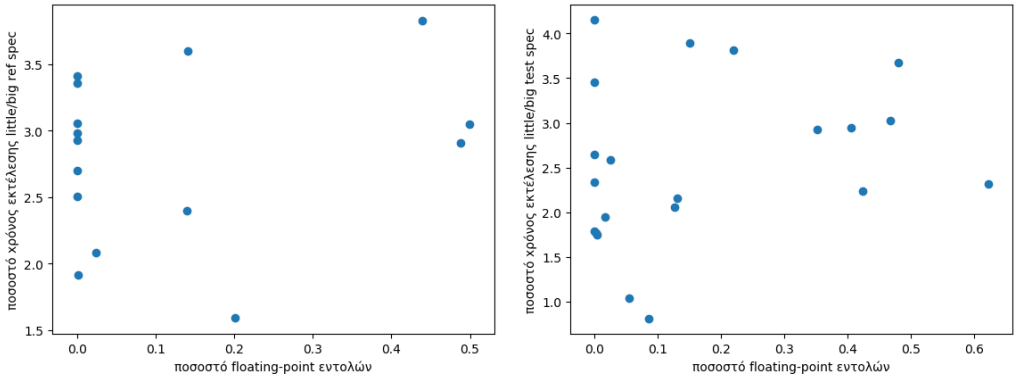
## 4.5 Συσχετιστικά διαγράμματα

Αφού παραθέσαμε τις μετρήσεις μας πάνω στα Spec2006 προγράμματα. Δείχνουμε τα αποτελέσματά των μετρήσεών μας υπό μορφή συσχετιστικών διαγραμμάτων για κάθε παράμετρο ως προς την επίδοση. Σημειώνουμε ότι τα διαγράμματα των reference input sets δεν είναι απόλυτα αντιπροσωπευτικά της πραγματικότητας, αφού καταφέραμε να πραγματοποιήσουμε μετρήσεις με το binary tool mambo μόνο σε δεκαπέντε περιπτώσεις, εκ των οποίων οι έξι αφορούσαν διαφορετικά reference σύνολα εισόδου του benchmark 401.bzip2. Όπως έχουμε αναφέρει το συγκεκριμένο benchmark είναι μία καθαρά integer εφαρμογή με μηδενικά ποσοστά floating-point εντολών. Πιστεύουμε ότι στο συγκεκριμένο δείγμα που αναλύουμε υπάρχει έντονη μεροληψία, η οποία δεν μας επιτρέπει να εξάγουμε συμπεράσματα που αντιπροσωπεύουν την πραγματικότητα. Παρόλα αυτά παρατηρούμε ότι όσο αυξάνονται τα L2 cache misses σε big και σε little πυρήνα, σε πολλές περιπτώσεις αυξάνεται σχεδόν αναλογικά η τιμή του συντελεστή a.



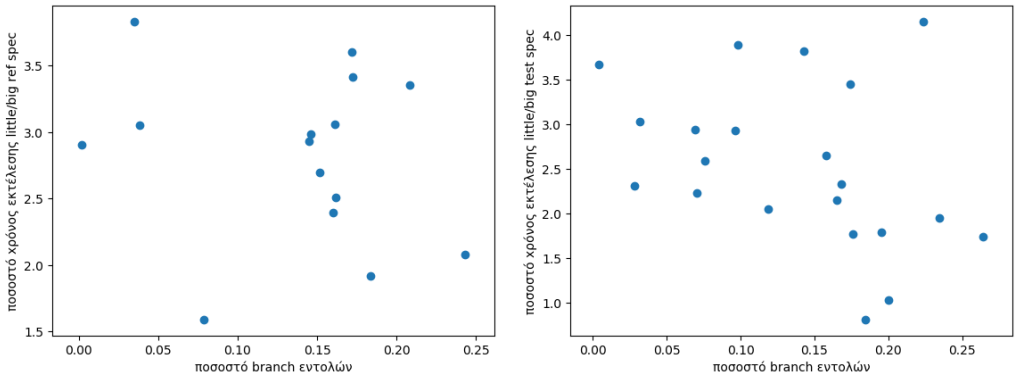

(1) Reference input set. (2) Test input set.

Εικόνα 4.9: Συσχετιστικό διάγραμμα integer εντολών και συντελεστή a για τα Spec2006 benchmarks.



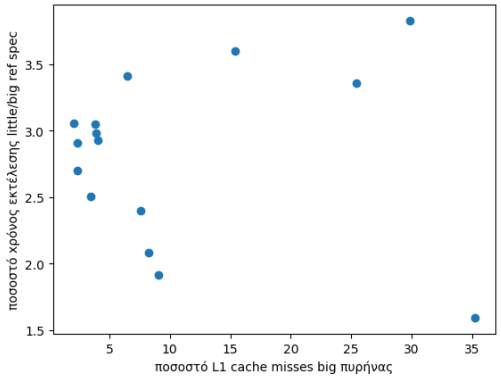

(1) Reference input set. (2) Test input set.

Εικόνα 4.10: Συσχετιστικό διάγραμμα floating-point εντολών και συντελεστή a για τα Spec2006 benchmarks.

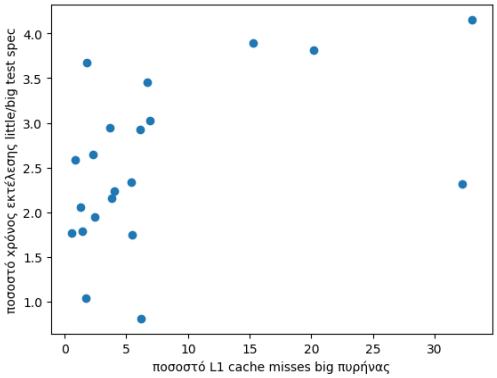



(1) Reference input set. (2) Test input set.

Εικόνα 4.11: Συσχετιστικό διάγραμμα branch εντολών και συντελεστή a για τα Spec2006 benchmarks.

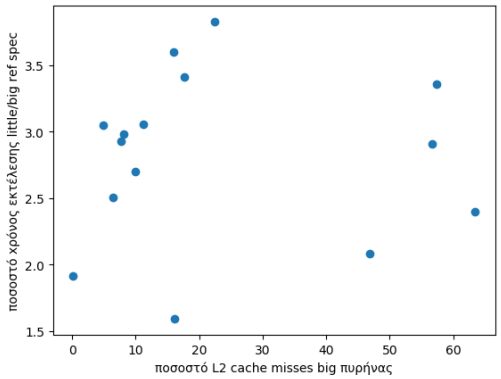



(1) Reference input set.

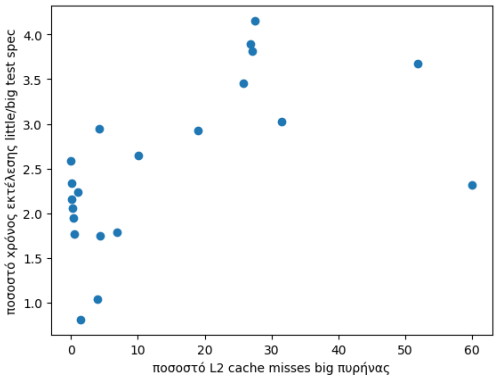



(2) Test input set.

Εικόνα 4.12: Συσχετιστικό διάγραμμα L1 cache misses στον big πυρήνα και συντελεστή a για τα Spec2006 benchmarks.

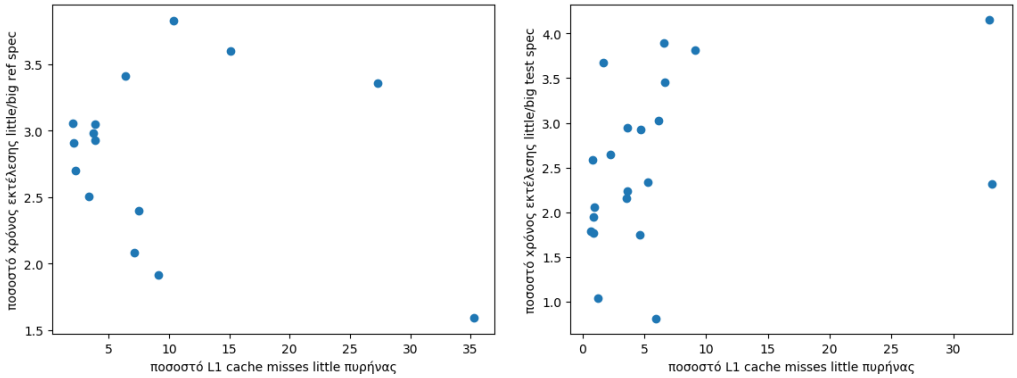



(1) Reference input set.



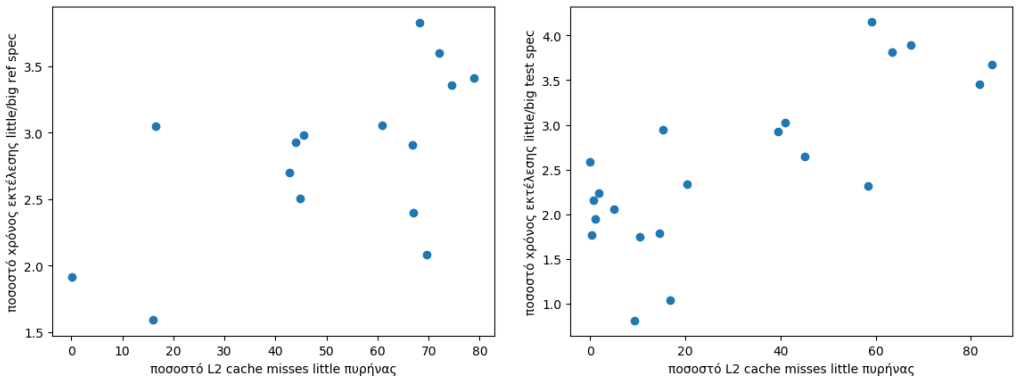

(2) Test input set.

Εικόνα 4.13: Συσχετιστικό διάγραμμα L2 cache misses στον big πυρήνα και συντελεστή a για τα Spec2006 benchmarks.




(1) Reference input set. (2) Test input set.

Εικόνα 4.14: Συσχετιστικό διάγραμμα L1 cache misses στον little πυρήνα και συντελεστή a για τα Spec2006 benchmarks.

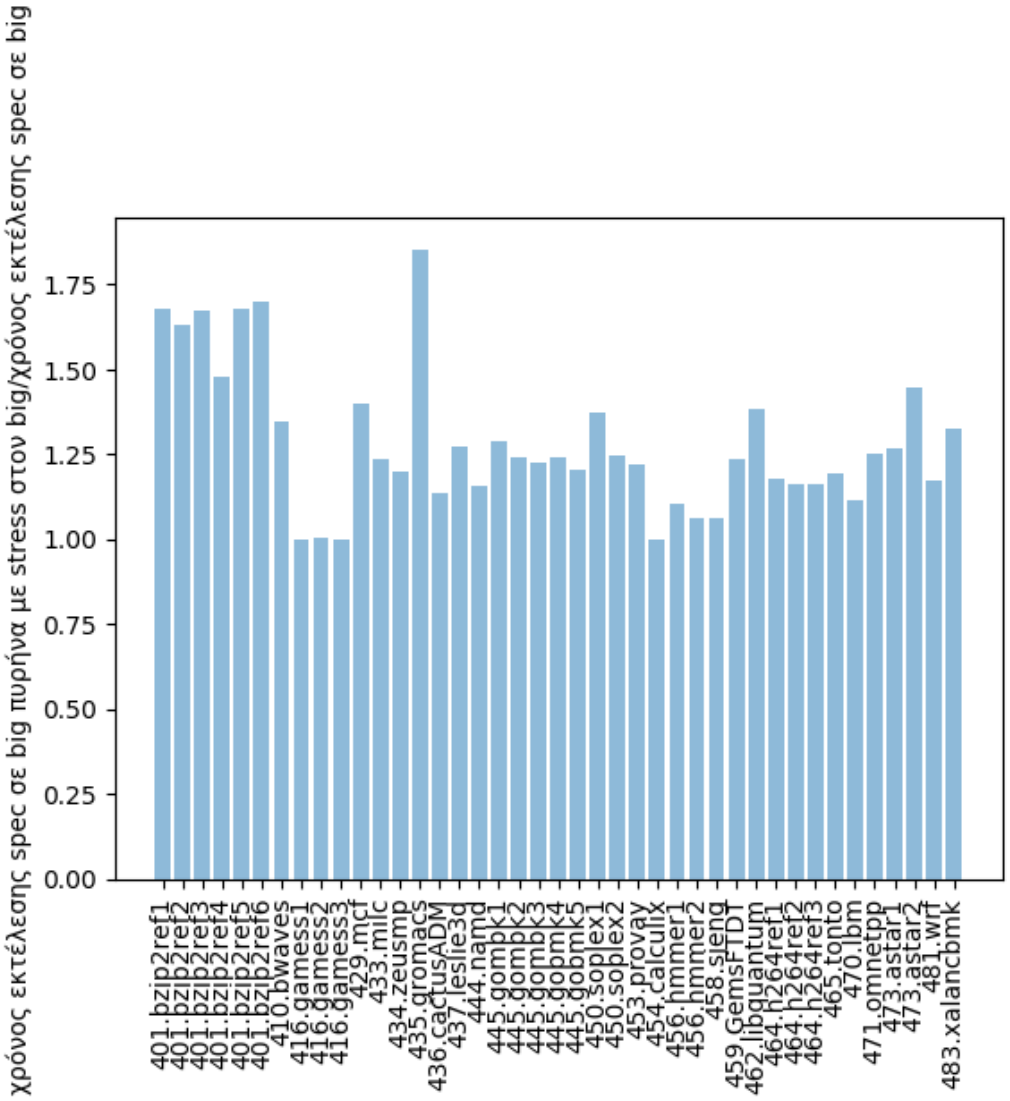



(1) Reference input set. (2) Test input set.

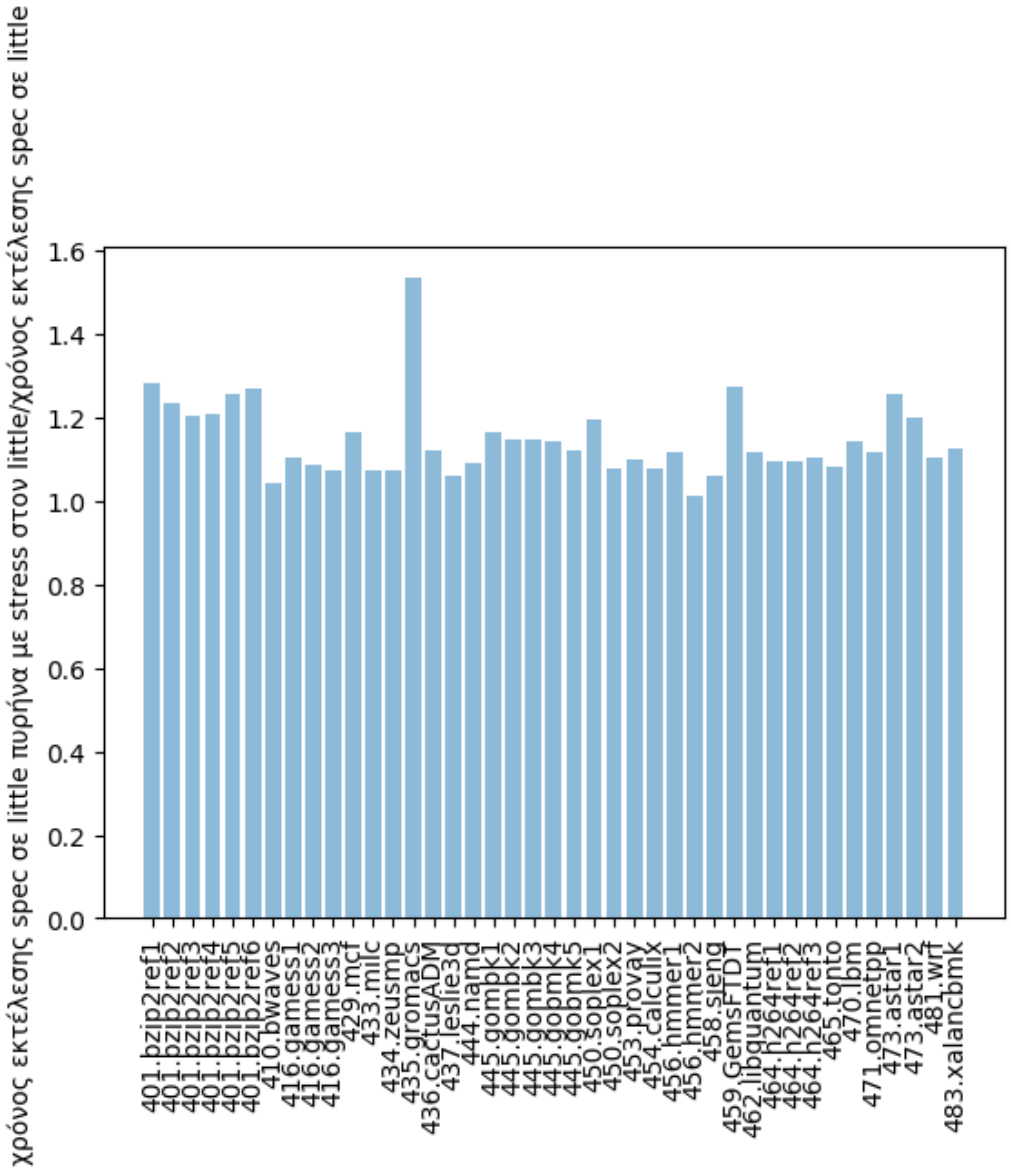
Εικόνα 4.15: Συσχετιστικό διάγραμμα L2 cache misses στον little πυρήνα και συντελεστή a για τα Spec2006 benchmarks.

## 4.6 Συνεκτέλεση Spec2006 και stress προγραμμάτων

Σε αυτήν τη φάση της έρευνάς μας αναθέσαμε την εκτέλεση ενός stress προγράμματος σε έναν big ή little πυρήνα και σε έναν άλλον πυρήνα, ίδιου είδους, εκτελέσαμε διαδοχικά τα Spec2006 προγράμματά μας. Χρησιμοποιήσαμε τόσο τα reference input sets, όσο και τα test input sets, αλλά θα παρουσιά-




Εικόνα 4.16: Ποσοστό χρόνου εκτέλεσης των Spec2006 προγραμμάτων σε big πυρήνα με συνεκτελούμενο stress ως προς το χρόνο εκτέλεσης σε big πυρήνα.



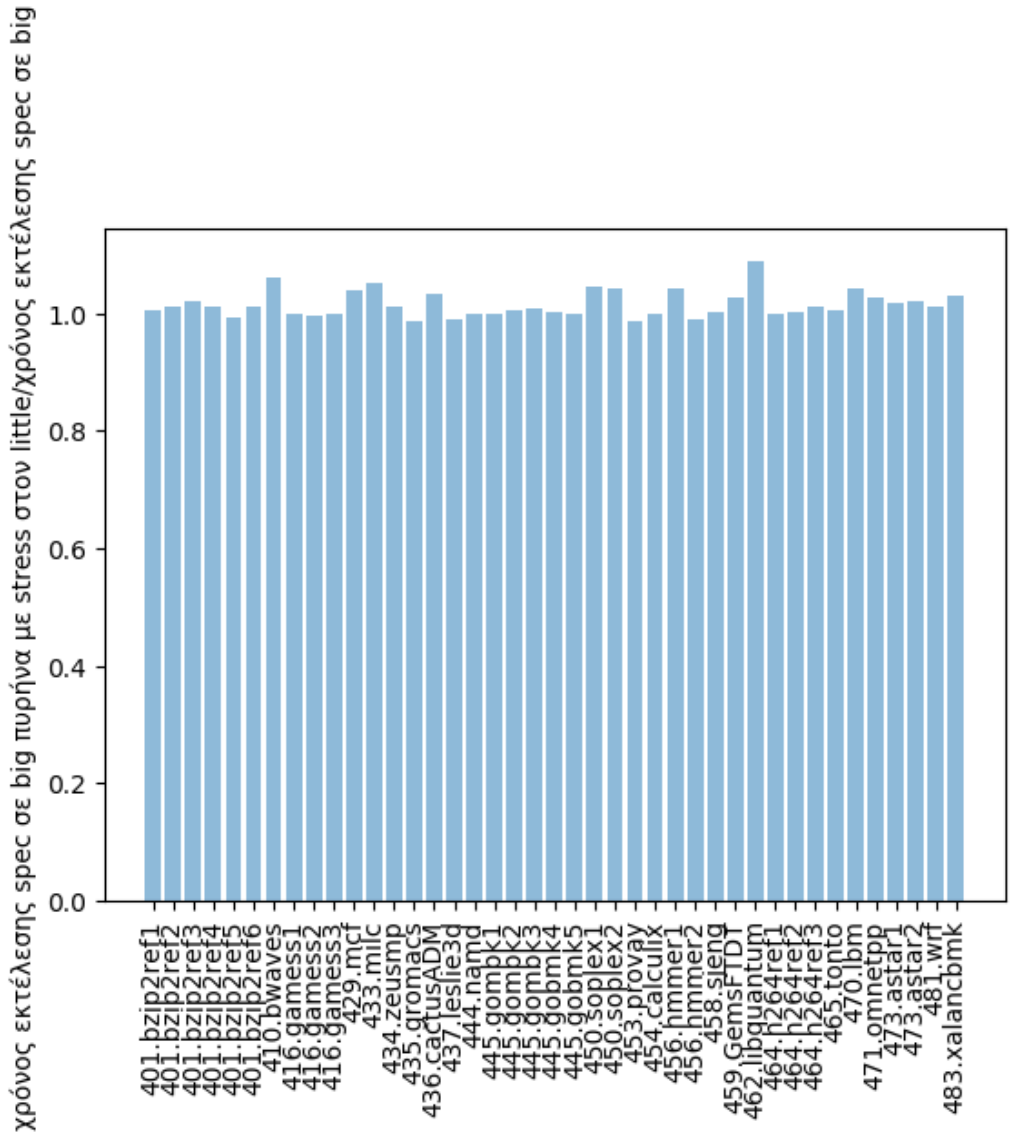

Εικόνα 4.17: Ποσοστό χρόνου εκτέλεσης των Spec2006 προγραμμάτων σε little πυρήνα με συνεκτελούμενο stress ως προς το χρόνο εκτέλεσης σε little πυρήνα.

σουμε μόνο τα αποτελέσματα των reference input sets γιατί τα θεωρούμε αντιπροσωπευτικότερα της πραγματικότητας. Τα αποτελέσματα των μετρήσεων της συγκεκριμένης κατηγορίας βρίσκονται στις εικόνες 4.16 και 4.17.

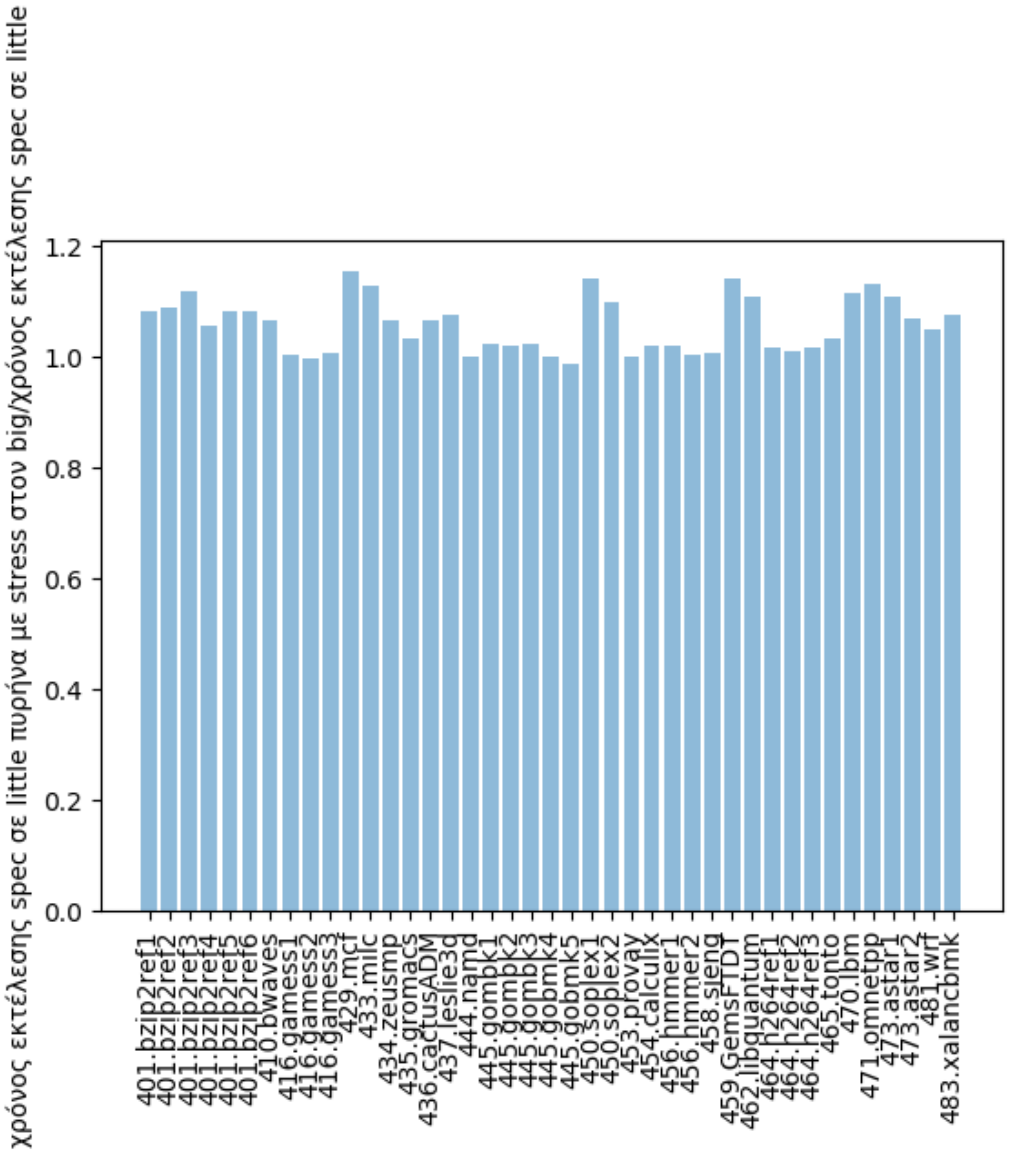
Με βάση τα αποτελέσματα των μετρήσεων στις εικόνες 4.16 και 4.17, παρατηρούμε ότι πολλά από τα προγράμματα Spec2006 παρουσιάζουν σημαντικές αποκλίσεις στους χρόνους εκτέλεσής τους όταν εκτελούνται μόνα τους σε σχέση με την συνεκτέλεσή τους με stress πρόγραμμα στο ίδιο σύμπλεγμα πυρήνων. Μάλιστα παρατηρούμε ότι όταν οι εκτελέσεις της συγκεκριμένης κατηγορίας πραγματοποιούνται στον big πυρήνα τα ποσοτά των αποκλίσεων είναι μεγαλύτερα. Ακόμα αξίζει να σημειώσουμε ότι υπήρχαν προγράμματα με υψηλό συντελεστή a, αλλά και με χαμηλό συντελεστή a που εμφάνιζαν σημαντικές αποκλίσεις στους χρόνους εκτέλεσής τους όταν εκτελούνται μόνα τους σε σχέση με την συνεκτέλεσή τους με stress πρόγραμμα στο ίδιο σύμπλεγμα πυρήνων..

Στη συνέχεια αναθέσαμε την εκτέλεση ενός stress προγράμματος σε έναν big ή little πυρήνα και σε έναν άλλον πυρήνα, διαφορετικού είδους, εκτελούσαμε διαδοχικά τα Spec2006 προγράμματά μας. Ο στόχος μας ήταν να παρατηρήσουμε τη συμπεριφορά των προγραμμάτων όταν χρησιμοποιούν, ταυτόχρονα με το stress πρόγραμμα, τα buses, τους row buffers και τα banks της κύριας μνήμης. Χρησιμοποιήσαμε τόσο τα reference input sets, όσο και τα test input sets, αλλά και σε αυτήν την περίπτωση θα παρουσιάσουμε μόνο τα αποτελέσματα των reference input sets. Τα αποτελέσματα των μετρήσεων της συγκεκριμένης κατηγορίας βρίσκονται στις εικόνες 4.18 και 4.19.

Με βάση τα αποτελέσματα των μετρήσεων στις εικόνες 4.18 και 4.19, παρατηρούμε ότι τα προγράμματα Spec2006 παρουσιάζουν μικρές αποκλίσεις στους χρόνους εκτέλεσής τους όταν εκτελούνται μόνα τους σε σχέση με την συνεκτέλεσή τους με stress πρόγραμμα σε διαφορετικά συμπλέγματα πυρήνων. Θεωρούμε ότι ο ανταγωνισμός για προσβάσεις στην κύρια μνήμη δεν επηρεάζει καθοριστικά τους χρόνους εκτέλεσης των Spec2006 προγραμμάτων. Γι' αυτόν το λόγο δε χρειάστηκε να πραγματοποιήσουμε μοντελοποίηση με βάση τον ανταγωνισμό των προγραμμάτων για την κύρια μνήμη.



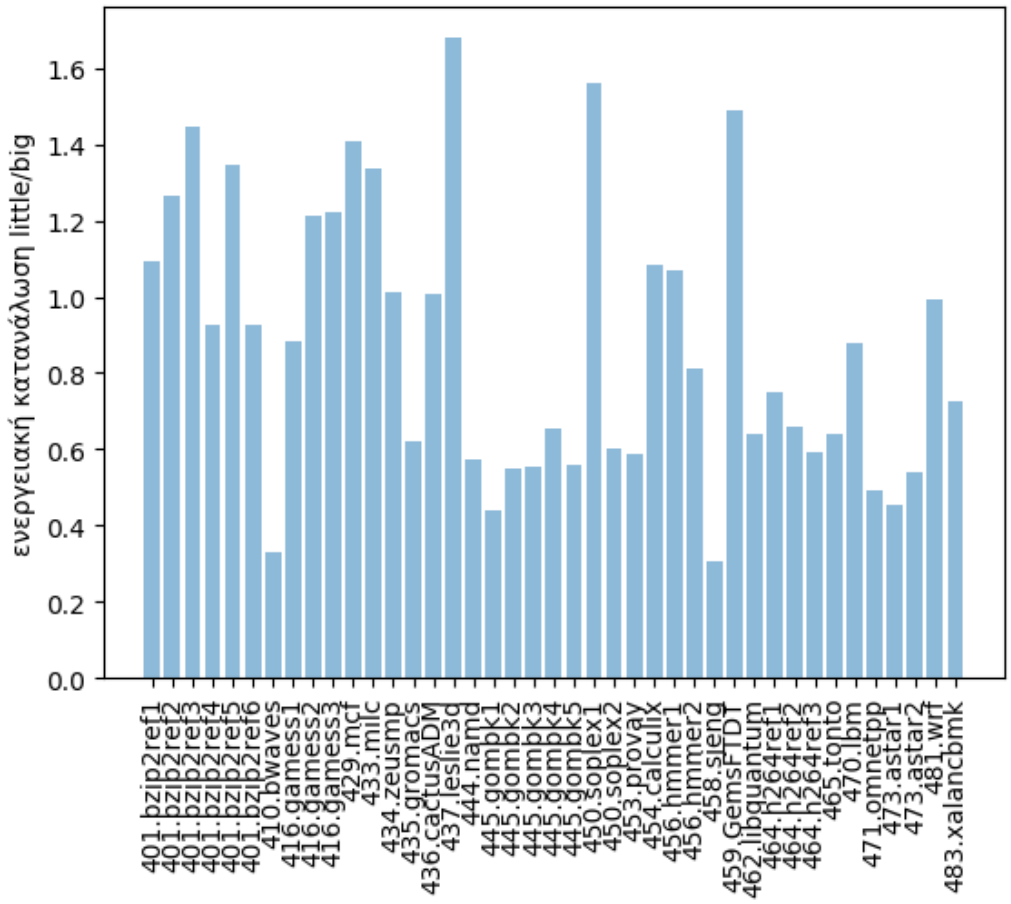

Εικόνα 4.18: Ποσοστό χρόνου εκτέλεσης των Spec2006 προγραμμάτων σε big πυρήνα με συνεκτελούμενο stress στον little ως προς το χρόνο εκτέλεσης σε big πυρήνα.



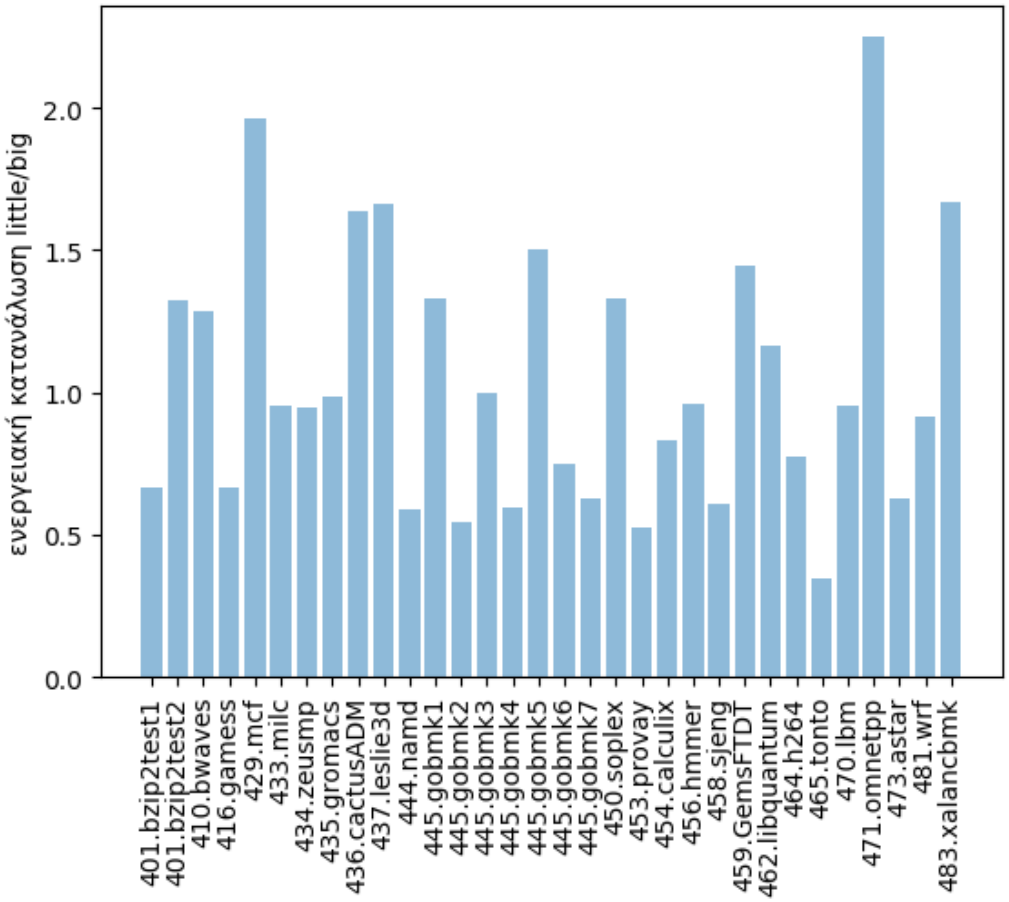

Εικόνα 4.19: Ποσοστό χρόνου εκτέλεσης των Spec2006 προγραμμάτων σε little πυρήνα με συνεκτελούμενο stress στον big ως προς το χρόνο εκτέλεσης σε little πυρήνα.

## 4.7 Ενεργειακές μετρήσεις στα Spec2006 προγράμματα

Μετρήσαμε την ενεργειακή κατανάλωση των Spec 2006 benchmarks. Το κάθε benchmark το τρέξαμε τόσο σε big όσο και σε little πυρήνα. Μετρήσαμε την ενεργειακή κατανάλωση των προγραμμάτων χρησιμοποιώντας reference και test input sets. Τα αποτελέσματα των ενεργειακών μετρήσεων βρίσκονται στις εικόνες 4.20 και 4.21. Στα διαγράμματα συγκρίνουμε την ενεργειακή κατανάλωση των προγραμμάτων όταν εκτελούνται στα δύο είδη πυρήνων. Συγκεκριμένα παρουσιάζουμε στα διαγράμματα ένα συντελεστή ενεργειακής κατανάλωσης benchmark στον little πυρήνα προς την ενεργειακή κατανάλωση του ίδιου benchmark στον big πυρήνα.




Εικόνα 4.20: Ενεργειακή κατανάλωση των Spec2006 προγραμματων με χρήση reference input sets.

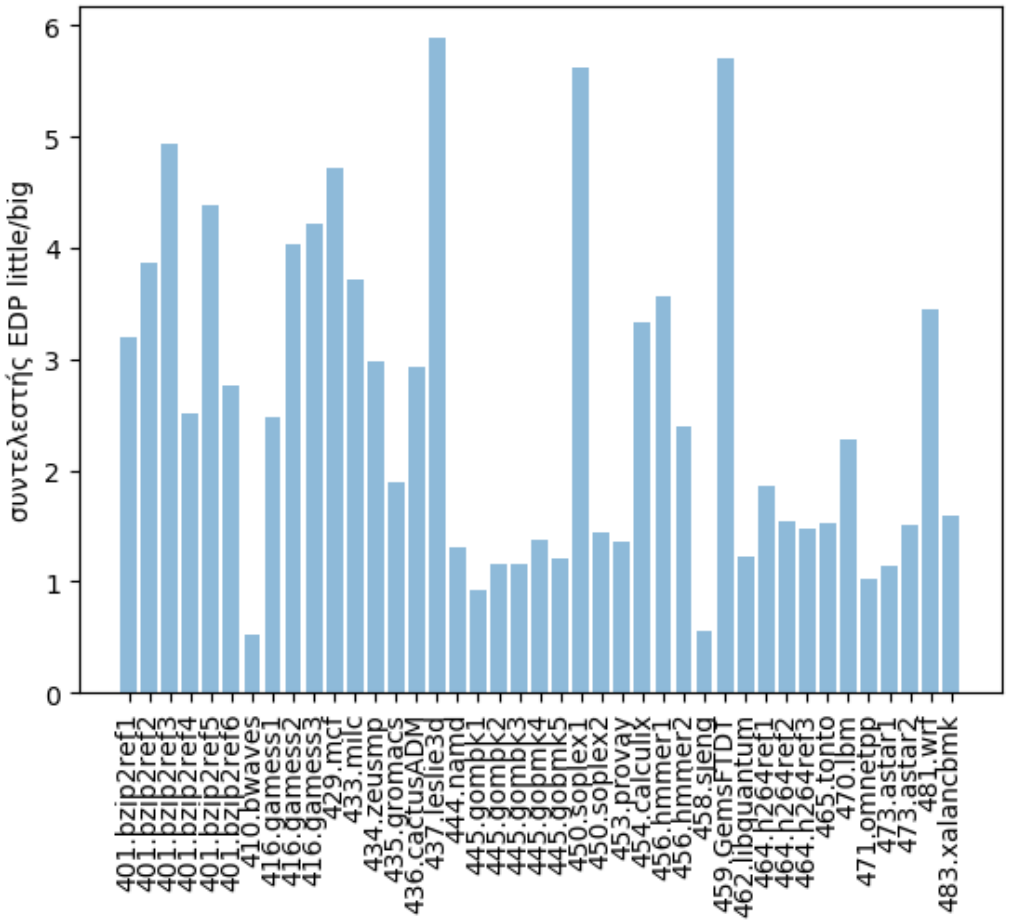



Εικόνα 4.21: Ενεργειακή κατανάλωση των Spec2006 προγραμμάτων με χρήση test input sets.

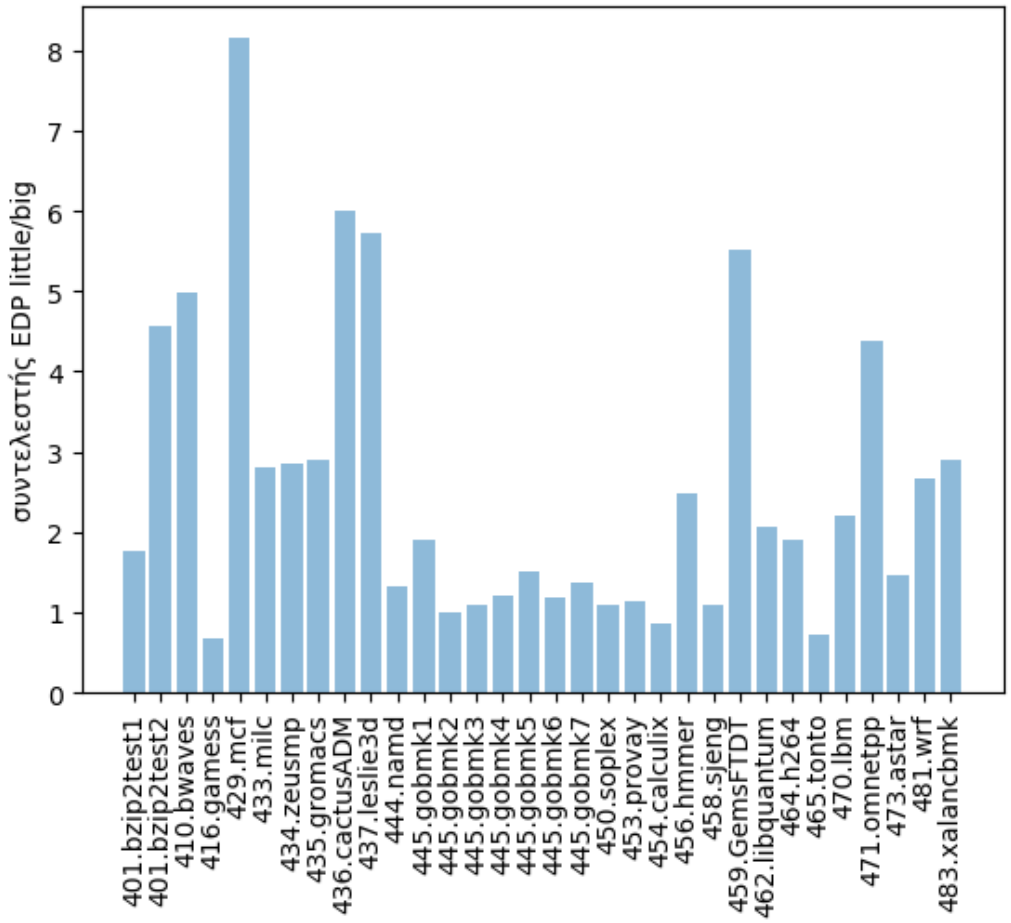
Με βάση τις ενεργειακές μας μετρήσεις παρατηρούμε ότι στα περισσότερα προγράμματα η ενεργειακή τους κατανάλωση είναι χαμηλότερη στους little πυρήνες, παρά το γεγονός ότι εκεί εκτελούνται για περισσότερο χρόνο. Βλέπουμε, όμως, ότι σε εφαρμογές με υψηλό συντελεστή a, δηλαδή σε εφαρμογές που ο χρόνος εκτέλεσής τους σε big πυρήνα είναι πολύ ταχύτερος σε σχέση με το χρόνο εκτέλεσή τους σε little πυρήνα, η ενεργειακή κατανάλωση βελτιστοποιείται, σε πολλές περιπτώσεις, στον big πυρήνα. Ειδικότερα, τα προγράμματα 401.bzip2, 416.gamess, 429.mcf, 437.leslie3d, 450.soplex, 454.calculix, 456.hmmer, 459.GemsFDT που ανήκουν στην κατηγορία προγραμμάτων με υψηλό συντελεστή a εμφανίζουν ακόμα και μικρότερη ενεργειακή κατανάλωση όταν εκτελούνται σε big πυρήνα, τουλάχιστον με τη χρήση κάποιων από τα διαθέσιμα reference input sets. Επιπλέον τα προγράμματα 433.milc, 434.zeusmp και 436.cactusADM που εμφανίζουν χαμηλότερη ενεργειακή κατανάλωση στον

big πυρήνα έχουν και εκείνα υψηλό συντελεστή a, αν και δεν ανήκουν στην κατηγορία με συντελεστή a πάνω από 3. Αντίθετα παρατηρούμε ότι όλες οι εφαρμογές που ανήκουν στην κατηγορία με χαμηλό συντελεστή a εμφανίζουν πολύ χαμηλή ενεργειακή κατανάλωση στον little πυρήνα. Ειδικότερα τα benchmarks της συγκεκριμένης κατηγορίας είναι το 410.bwaves, το 445.gobmk, το 458.sjeng, το 462.libquantum και το 471.omnetpp. Άρα, με βάση τα πειραματικά αποτελέσματα παρατηρούμε ότι ο χρόνος εκτέλεσης ενός προγράμματος στο κάθε είδος πυρήνα καθορίζει την ενεργειακή του κατανάλωση.

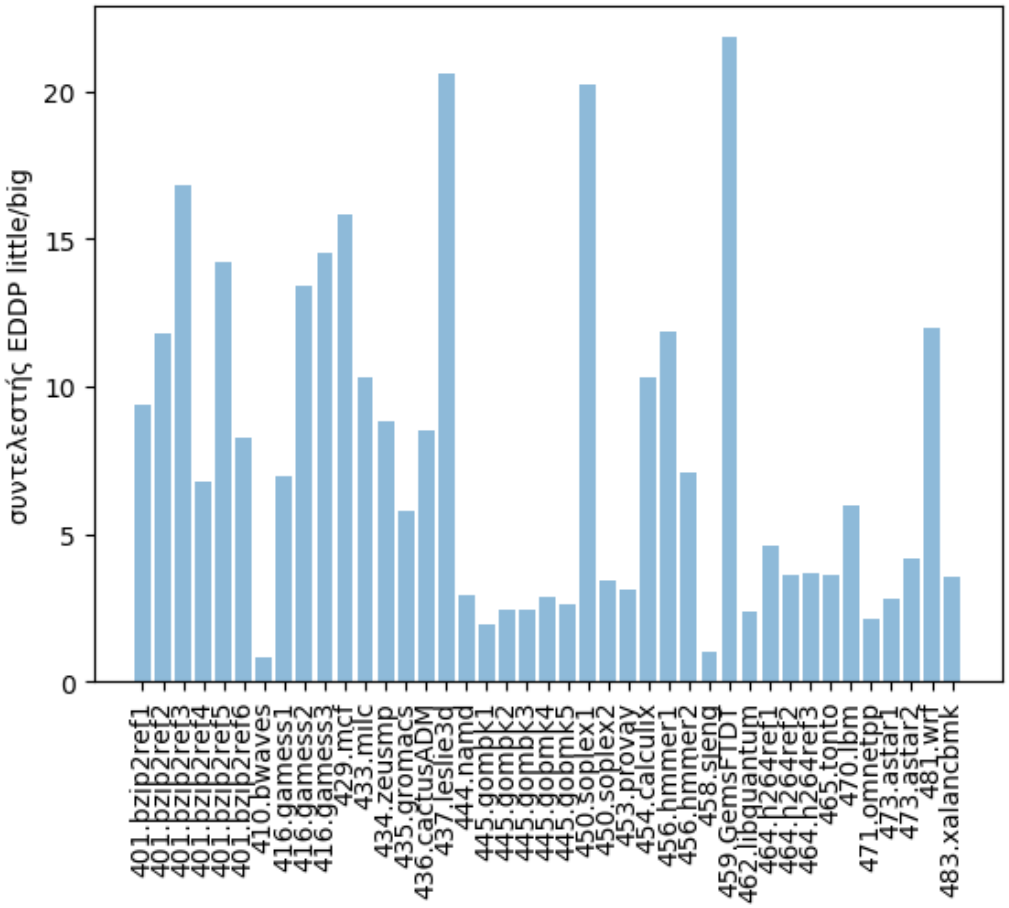
Αφού πραγματοποιήσαμε χρονικές και ενεργειακές μετρήσεις πάνω στα Spec 2006 benchmarks, στη συνέχεια δημιουργήσαμε διαγράμματα με βάση τους συντελεστές EDP, $ED^2P$ για reference και για test εισόδους. Τα συγκεκριμένα διαγράμματα βρίσκονται στις εικόνες 4.22, 4.23, 4.24 και 4.25. Έχουμε χρησιμοποιήσει αυτές τις μετρικές για τη μοντελοποίηση των προγραμμάτων στο αντίστοιχο κεφάλαιο. Στα διαγράμματα συγκρίνουμε τους EDP και $ED^2P$ των προγραμμάτων όταν εκτελούνται στα δύο είδη πυρήνων. Αξίζει να σημειώσουμε ότι η συντριπτική πλειοψηφία των προγραμμάτων παρουσιάζει καλύτερες τιμές EDP και $ED^2P$ στον big πυρήνα.



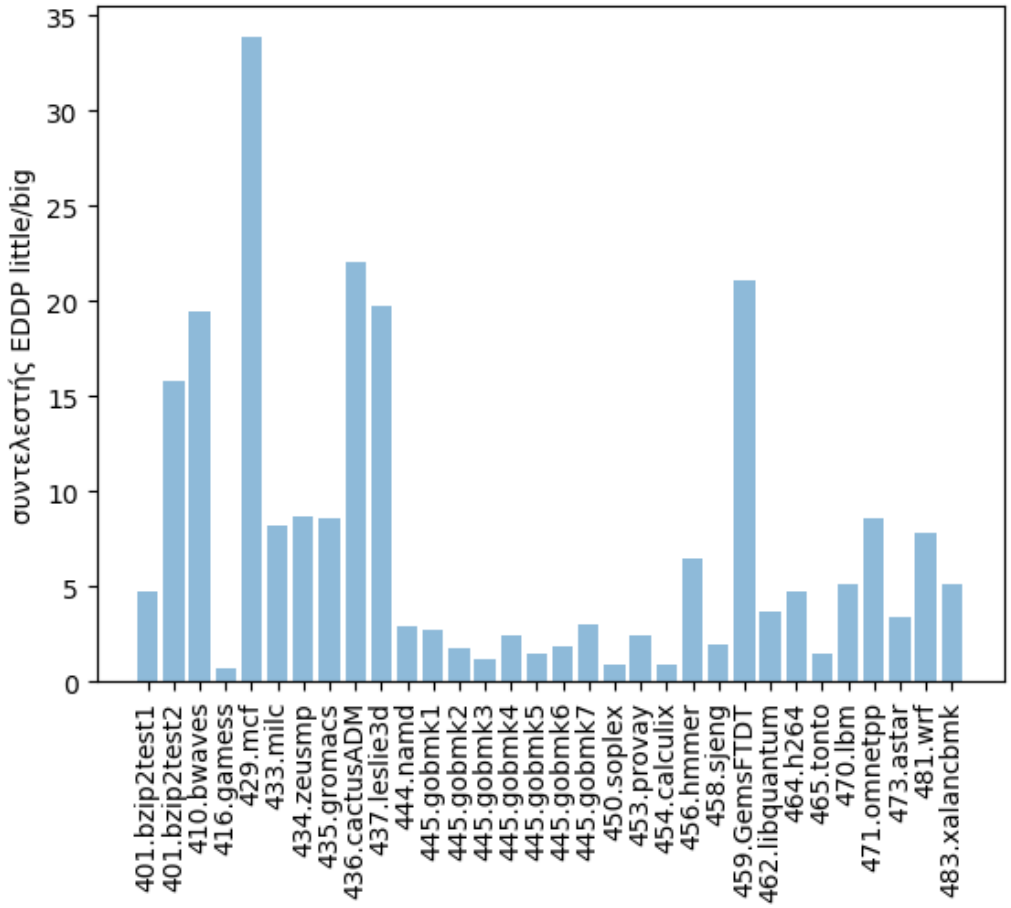

Εικόνα 4.22: Συντελεστές EDP των Spec2006 προγραμματων με χρήση reference input sets.




Εικόνα 4.23: Συντελεστές EDP των Spec2006 προγραμμάτων με χρήση test input sets.




Εικόνα 4.24: Συντελεστές $ED^2P$ των Spec2006 προγραμματων με χρήση reference input sets.




Εικόνα 4.25: Συντελεστές $ED^2P$ των Spec2006 προγραμμάτων με χρήση test input sets.

## 4.8 Συμπεράσματα ανάλυσης Spec2006 προγραμμάτων

Συμπερασματικά με βάση τις μετρήσεις μας και τη βιβλιογραφία παρατηρούμε ότι καθοριστικός παράγοντας για τους χρόνους εκτέλεσης των προγραμμάτων είναι τα cache misses. Η πρόσβαση στην L1 cache εμφανίζει χαμηλό latency, το οποίο πολλαπλασιάζεται για προσβάσεις στην L2 cache. Τα L2 cache misses προκαλούν προσβάσεις στην κύρια μνήμη με κόστος της τάξεως των εκατοντάδων κύκλων ρολογιού. Αξίζει να σημειώσουμε ότι στα περισσότερα benchmarks τα L2 cache misses αυξάνονται σημαντικά στους little πυρήνες, εξαιτίας του πολύ μικρού μεγέθους της L2 cache τους. Λόγω των λιγότερων L2 cache

misses η εκτέλεση των benchmarks στους big πυρήνες επιταχύνεται σημαντικά.

Παρατηρούμε, ακόμα, ότι το υψηλό ποσοστό αριθμητικών πράξεων και ιδιαίτερα floating-point εντολών εκτελείται ταχύτερα στους out-of-order big πυρήνες. Μάλιστα, οι floating-point εντολές είναι πιο χρονοβόρες σε κύκλους σε σχέση με τις integer. Επιπλέον στους A57 και στους A53 πυρήνες μέχρι τρεις πράξεις ακεραίων μπορούν να εκτελεστούν παράλληλα, πάντα σε περίπτωση μη ύπαρξης εξαρτήσεων. Αντίθετα οι πυρήνες A57 διαθέτουν δύο μονάδες floating-point εντολών, ενώ οι πυρήνες A53 μόνο μία. Γι' αυτόν το λόγο οι big πυρήνες ευνοούν την εκτέλεση floating-point εντολών. Χρονικά κοστοβόρες είναι και οι simd αριθμητικές εντολές, αλλά δεν παρατηρούνται εκτεταμένα στα πειραματικά μας αποτελέσματα. Τα πολλά branch misspredictions φαίνεται ότι συμβάλλουν στη σύγκλιση των χρόνων εκτέλεσης μεταξύ των big και των little πυρήνων. Τέλος, ένας ακόμα παράγοντας που ευνοεί την εκτέλεση προγραμμάτων σε little πυρήνες είναι και ο υψηλός αριθμός εξαρτήσεων μεταξύ των εντολών, ο οποίος οδηγεί σε χαμηλούς συντελεστές IPC (Instructions per Cycle) και ILP (Instruction Level Parallelism).

Όσον αφορά την ενεργειακή κατανάλωση των Spec2006 προγραμμάτων, παρατηρούμε ότι προγράμματα με υψηλό συντελεστή a ευνοούν την εκτέλεση των προγραμμάτων σε big πυρήνες. Σε πολλές περιπτώσεις παρουσίαζαν ακόμα χαμηλότερη ενεργειακή κατανάλωση σε big πυρήνες σε σχέση με την αντίστοιχη σε little. Ο χρόνος εκτέλεσης μίας εντολής είναι άρρηκτα συνδεδεμένος με την ενεργειακή της κατανάλωση. RAW (read after write) εξαρτήσεις προκαλούν αύξηση της ενεργειακής κατανάλωσης μίας εντολής. Αξίζει να σημειώσουμε ότι όσο κατεβαίνουμε επίπεδο μνήμης, τόσο αυξάνεται η ενεργειακή κατανάλωση. Οι προσβάσεις στην κύρια μνήμη έχουν ενεργειακό κόστος πολλές φορές παραπάνω σε σχέση με μία σύνθετη εντολή πρόσθεσης.

# Κεφάλαιο 5

# Ανάλυση Parsec Benchmarks

## 5.1 Εισαγωγή

Παραθέτουμε στη συνέχεια τις μετρήσεις μας και την ανάλυσή μας πάνω στα Parsec, δηλαδή για προγράμματα που υποστηρίζουν πολυνηματικές εκτελέσεις. Οι μετρήσεις πραγματοποιήθηκαν στο ίδιο μηχάνημα με αυτές των Spec2006 προγραμμάτων. Υπενθυμίζουμε ότι και στις συγκεκριμένες μετρήσεις οι επεξεργαστικοί πυρήνες ήταν ρυθμισμένοι στις μέγιστες δυνατές συχνότητες. Οι big πυρήνες "έτρεχαν" στα 1,5 GHz και οι little στα 2,1 GHz. Οι συνδυαστικές μετρήσεις της συμπεριφοράς των ετερογενών πυρήνων μας επιτρέπουν να εξάγουμε επιπρόσθετα συμπεράσματα, σε σχέση με τις παρατηρήσεις μας πάνω στα μονονηματικά Spec2006 προγράμματα. Στην Εικόνα 5.1 παρέχουμε πληροφορίες για τα Parsec προγράμματα. Από τα προγράμματα της Εικόνας 5.1 μόνο το freqmine δεν χρησιμοποιήσαμε στις μετρήσεις μας.

| **Program** | **Application Domain** | **Parallelization** | | **Working Set** | **Data Usage** | |
|---|---|---|---|---|---|---|
| | | **Model** | **Granularity** | | **Sharing** | **Exchange** |
| `blackscholes` | Financial Analysis | data-parallel | coarse | small | low | low |
| `bodytrack` | Computer Vision | data-parallel | medium | medium | high | medium |
| `canneal` | Engineering | unstructured | fine | unbounded | high | high |
| `dedup` | Enterprise Storage | pipeline | medium | unbounded | high | high |
| `facesim` | Animation | data-parallel | coarse | large | low | medium |
| `ferret` | Similarity Search | pipeline | medium | unbounded | high | high |
| `fluidanimate` | Animation | data-parallel | fine | large | low | medium |
| `freqmine` | Data Mining | data-parallel | medium | unbounded | high | medium |
| `streamcluster` | Data Mining | data-parallel | medium | medium | low | medium |
| `swaptions` | Financial Analysis | data-parallel | coarse | medium | low | low |
| `vips` | Media Processing | data-parallel | coarse | medium | low | medium |
| `x264` | Media Processing | pipeline | coarse | medium | high | high |

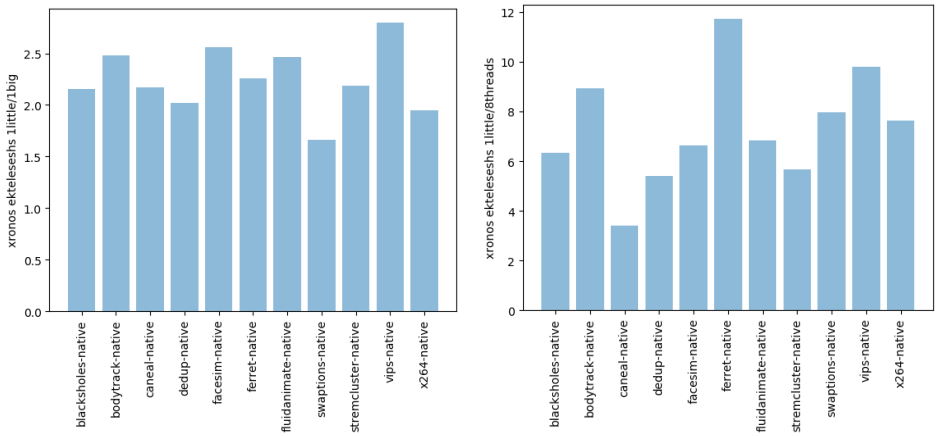
Εικόνα 5.1: Parsec benchmarks.

## 5.2 Αντίκτυπος της ετερογένειας big.LITTLE στην απόδοση

Σε πρώτη φάση μετρήσαμε τους χρόνους εκτέλεσης των Parsec προγραμμάτων με διαφορετικό αριθμό threads. Σε όλες τις εκτελέσεις αναθέταμε την εκτέλεση ενός thread σε ένα μόνο πυρήνα, δηλαδή χωρίς να ενδιαφερόμαστε για την εφαρμογή simultaneous multithreading (SMT). Τους ίδους συνδυασμούς εκτελέσεων των Parsec προγραμμάτων τους πραγματοποιήσαμε με χρήση native και με χρήση simlarge input sets.

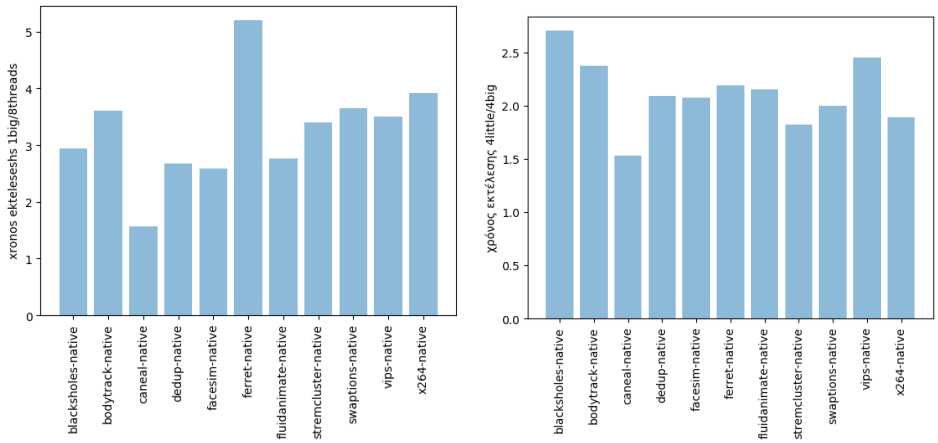
Ορίζουμε τους συντελεστές των χρόνων εκτέλεσης των Parsec προγραμμάτων όταν χρησιμοποιούμε τα native input sets ως εξής:

- σ1 = χρόνος εκτέλεσης Parsec σε 1 little πυρήνα/χρόνος εκτέλεσης Parsec σε 1 big
- σ2 = χρόνος εκτέλεσης Parsec σε 1 little πυρήνα/χρόνος εκτέλεσης Parsec και στους 8 πυρήνες
- σ3 = χρόνος εκτέλεσης Parsec σε 1 big πυρήνα/χρόνος εκτέλεσης Parsec και στους 8 πυρήνες
- σ4 = χρόνος εκτέλεσης Parsec σε 4 little πυρήνες/χρόνος εκτέλεσης Parsec σε 4 big πυρήνες
- σ5 = χρόνος εκτέλεσης Parsec σε 4 little πυρήνες/χρόνος εκτέλεσης Parsec σε 2 big και σε 2 little πυρήνες
- σ6 = χρόνος εκτέλεσης Parsec σε 4 big πυρήνες/χρόνος εκτέλεσης Parsec σε 2 big και σε 2 little πυρήνες



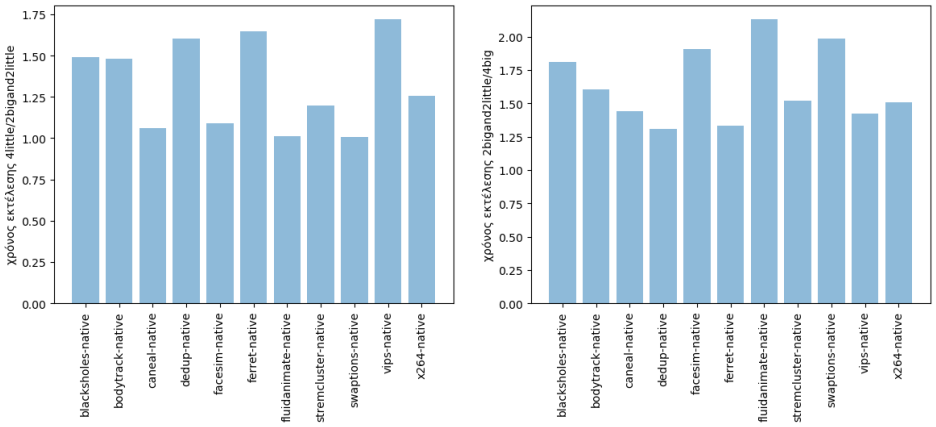

(1) Συντελεστής σ1. (2) Συντελεστής σ2.

Εικόνα 5.2: Αποτελέσματα για τους συντελεστές σ1 και σ2 για τα Parsec benchmarks.




(1) συντελεστής σ3 (2) συντελεστής σ4

Εικόνα 5.3: Αποτελέσματα για τους συντελεστές σ3 και σ4 για τα Parsec benchmarks.




(1) συντελεστής σ5 (2) συντελεστής σ6

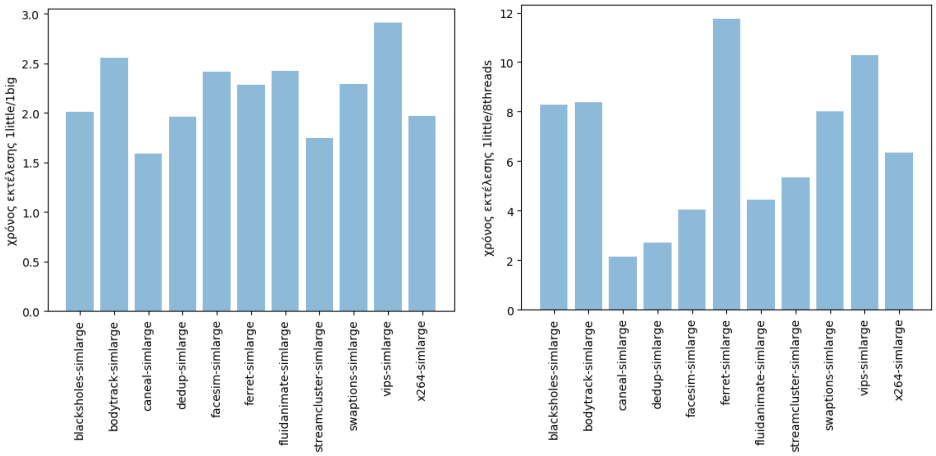
Εικόνα 5.4: Αποτελέσματα για τους συντελεστές σ5 και σ6 για τα Parsec benchmarks.

Ορίζουμε τους συντελεστές των χρόνων εκτέλεσης των Parsec προγραμμάτων όταν χρησιμοποιούμε τα simlarge input sets ως εξής:

- s1 = χρόνος εκτέλεσης Parsec σε 1 little πυρήνα/χρόνος εκτέλεσης Parsec σε 1 big
- s2=χρόνος εκτέλεσης Parsec σε 1 little πυρήνα/χρόνος εκτέλεσης Parsec και στους 8 πυρήνες
- s3=χρόνος εκτέλεσης Parsec σε 1 big πυρήνα/χρόνος εκτέλεσης Parsec και στους 8 πυρήνες
- s4 = χρόνος εκτέλεσης Parsec σε 4 little πυρήνες/χρόνος εκτέλεσης Parsec σε 4 big πυρήνες
- s5=χρόνος εκτέλεσης Parsec σε 4 little πυρήνες/χρόνος εκτέλεσης Parsec σε 2 big και σε 2 little πυρήνες
- s6=χρόνος εκτέλεσης Parsec σε 4 big πυρήνες/χρόνος εκτέλεσης Parsec σε 2 big και σε 2 little πυρήνες

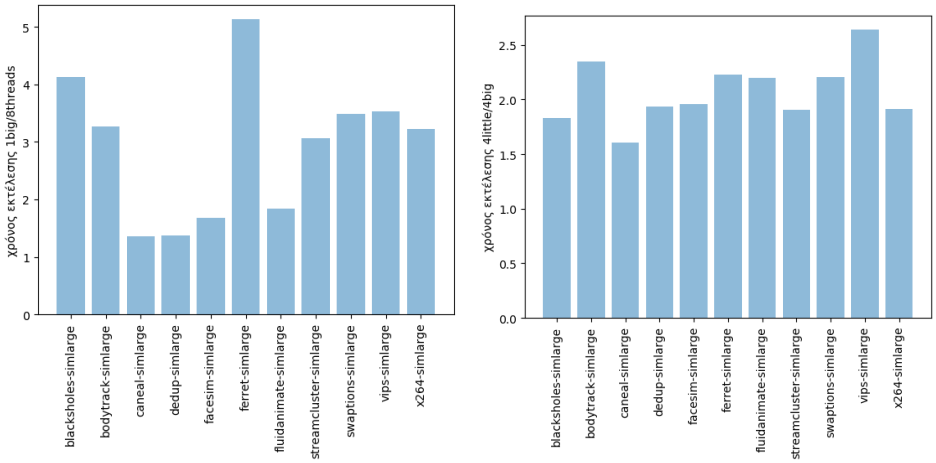
### 5.2.1 Ανάλυση Parsec προγραμμάτων

Στα Parsec προγράμματα εξετάζουμε, λόγω της μορφής τους, νέες παραμέτρους πέρα από αυτές που ασχοληθήκαμε στα Spec2006 προγράμματα. Στα Spec2006



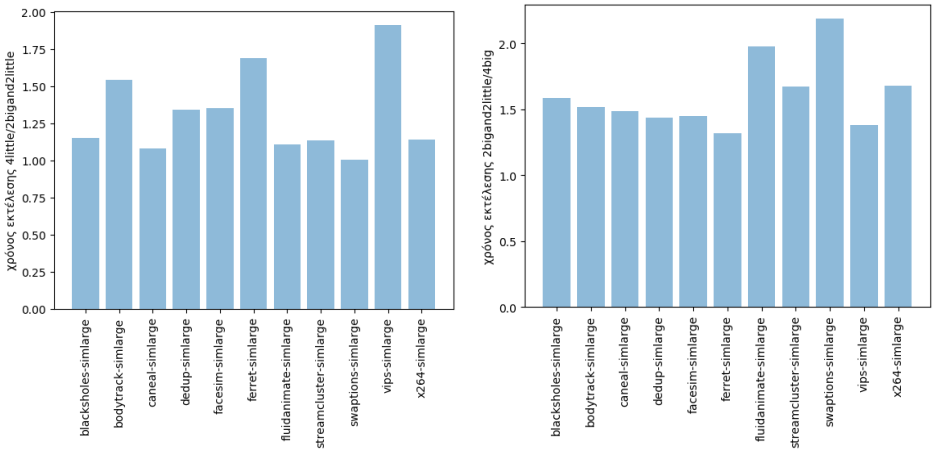

(1) Συντελεστής s1 (2) Συντελεστής s2

Εικόνα 5.5: Αποτελέσματα για τους συντελεστές s1 και s2 για τα Parsec benchmarks.




(1) συντελεστής s3 (2) συντελεστής s4

Εικόνα 5.6: Αποτελέσματα για τους συντελεστές s3 και s4 για τα Parsec benchmarks.




(1) Συντελεστής s5 (2) Συντελεστής s6

Εικόνα 5.7: Αποτελέσματα για τους συντελεστές s5 και s6 για τα Parsec benchmarks.

προγράμματα ενδιαφερόμασταν για τα L1 και L2 cache misses, για το instruction mix των εντολών και για τα branch misspredictions. Στα Parsec προγράμματα μας ενδιαφέρει επιπλέον ο παραλληλισμός των προγραμμάτων, ο αριθμός των locks στα threads, τα barriers, το μέγεθος της κατάτμησης των προγραμμάτων καθώς και η ανάγκη για επικοινωνία μεταξύ των πυρήνων. Επιπλέον στα Parsec προγράμματα εξετάζουμε τα cache misses και τις προσβάσεις στην κύρια μνήμη υπό λίγο διαφορετικό πρίσμα, λόγω του διαμοιρασμού της L2 μνήμης cache και της κύριας μνήμης από τα πολυνηματικά προγράμματα. Οι προσβάσεις στην κύρια μνήμη, πέρα από μεγάλο χρονικό κόστος, έχουν και ενεργειακό κόστος, πολλές φορές παραπάνω σε σχέση με τις σύνθετες αριθμητικές εντολές.

Επιπροσθέτως, με βάση τις παρατηρήσεις από άλλες έρευνες αναφέρουμε ότι σημαντικό ρόλο στις εκτελέσεις των Parsec προγραμμάτων παίζουν οι στοιχειώδεις μονάδες εκτέλεση εντολών. Συγκεκριμένα ο κορεσμός των μονάδων ROB, reservation station, floating-point unit, integer unit μπορεί να οδηγήσει σε σημαντικό αριθμό stalls. Εφαρμογές με χαμηλό ILP φαίνεται ότι μπορούν να υποστηρίξουν την εκτέλεση πολλών thread στον ίδιο πυρήνα, ενώ ταυτόχρονα η εκτέλεσή τους σε in-order πυρήνες δεν παρουσιάζει σημαντική επιβράδυνση σε σχέση με τους ενεργοβόρους out-of-order πυρήνες. Βέβαια για να υπάρχει κέρδος σε multithreaded εφαρμογές πρέπει να μην υπάρχουν εξαρτήσεις μεταξύ των threads.

**Blacksholes.** Το πρόγραμμα blacksholes είναι ένα πρόγραμμα οικονομικών αναλύσεων. Υπολογίζει τιμές για ένα χαρτοφυλάκιο αγορών, μέσω της μερικής διαφορικής εξίσωσης Black-Scholes. Εμφανίζει coarse-grained παραλληλισμό

και static load-balancing. Πραγματοποιεί αρκετές αριθμητικές πράξεις και η επικοινωνία μεταξύ των επεξεργαστών θεωρείται αμελητέα, καθώς δεν υπάρχουν εξαρτήσεις.

Ο συντελεστής σ1 θεωρείται μεσαίος. Παρόλα αυτά παρατηρούμε ότι εμφανίζει το μέγιστο συντελεστή σ4. Τόσο η αμελητέα διαδιεργασιακή επικοινωνία και ο ελάχιστος αριθμός locks, όσο και το υψηλό ποσοστό αριθμητικών floating-point πράξεων μας οδηγούν στο συμπέρασμα ότι η εκτέλεση του black-sholes με 4 threads στους 4 big πυρήνες είναι πολύ ταχύτερη σε σχέση με την εκτέλεση ίδιου αριθμού threads στους 4 little πυρήνες. Σημειώνουμε, ακόμα, ότι το ποσοστό των L2 cache misses στους big πυρήνες είναι 58.17%, ενώ στους little 65.15%. Το ποσοστό των L1 cache misses είναι σχεδόν μηδενικό, κάτω του 1%, και στα δύο είδη πυρήνων.

**Bodytrack.** Το benchmark bodytrack ανήκει στην κατηγορία της όρασης υπολογιστών. Είναι υπολογιστικά απαιτητικό πρόγραμμα, με πολλά στάδια επεξεργασίας. Παρουσιάζει medium-granular παραλληλισμό, dynamic load balancing και χαμηλή διανηματική επικοινωνία.

Η εφαρμογή εμφανίζει πολύ υψηλές τιμές σ1 και σ4, παρά τα barriers και το μεγάλο αριθμό locks. Φαίνεται ότι η εκτέλεση των σειριακών τμημάτων επιταχύνεται από τους big πυρήνες. Επιπλέον επιδεικνύει σημαντική κλιμακωσιμότητα κατά τον παραλληλισμό, όπως φαίνεται από τους συντελεστές σ2 και σ3. Με βάση τα αποτελέσματα των μετρήσεων το συγκεκριμένο benchmark είναι καλύτερο να εκτελείται σε big πυρήνες με ένα ή περισσότερα threads. Τη συγκεκριμένη συμπεριφορά ευνοεί και ο χαμηλός αριθμός branches της εφαρμογής, καθώς και το υψηλό ποσοστό floating-point εντολών. Παρατηρούμε, ακόμα, ο αριθμός των L1 cache misses είναι υψηλός, λίγο πάνω από 5%, και στα δύο είδη πυρήνων. Αντίθετα στη συγκεκριμένη εφαρμογή τα L2 cache misses είναι σχεδόν μηδενικά, κάτω του 1%, πάλι και στα δύο είδη πυρήνων.

**Canneal.** Το πρόγραμμα canneal επιχειρεί να ελαχιστοποιήσει το κόστος δρομολόγησης κατά τη σχεδίαση ενός chip, μέσω cache aware πιθανοτικών τεχνικών. Σημειώνουμε ότι οι cache aware αλγόριθμοι σχεδιάζονται με στόχο να ελαχιστοποιήσουν την μετακίνηση των σελίδων μνήμης στην cache. Πραγματοποιεί εναλλαγές στοιχείων μέχρι να επιτύχει τη βέλτιστη δρομολόγηση. Χρησιμοποιεί fine-grained παραλληλισμό και επιτυγχάνει συγχρονισμό χωρίς locks. Λόγω της έλλειψης κλειδωμάτων μπορεί να πραγματοποιηθούν λάθη δρομολόγησης, από τα οποία το πρόγραμμα επανέρχεται αυτόματα.

Με βάση τις μετρήσεις παρατηρούμε ότι εμφανίζει τους χαμηλότερους σ2 και σ3 συντελεστές, δηλαδή τη μικρότερη δυνατή παραλληλοποίηση. Εμφανίζει πάνω από 80% L2 cache misses και στα δύο είδη πυρήνων, ενώ σχετικά υψηλά είναι και τα L1 cache misses. Γενικότερα, η off-chip κίνηση θεωρείται πολύ μεγάλη. Παρατηρούμε, ακόμα, ότι είναι υπολογιστικά απαιτητική εφαρμογή με

σημαντικό ποσοστό integer εντολών.

**Dedup.** Το πρόγραμμα dedup έχει ως στόχο τη συμπίεση δεδομένων και επιτυγχάνεται σε 5 pipeline στάδια. Υποστηρίζει την παράλληλη επεξεργασία δεδομένων σε ξεχωριστά pipeline, ενώ ταυτόχρονα υπάρχει σημαντική διανηματική επικοινωνία.

Βλέπουμε ότι οι συντελεστές σ2 και σ3 παρουσιάζουν χαμηλές τιμές. Αναμέναμε τη συγκεκριμένη συμπεριφορά, λόγω της σημαντικής διανηματικής επικοινωνίας και του μεγάλου αριθμού κλειδωμάτων. Κάθε thread μπορεί να εκτελεί διαφορετικές λειτουργίες και να εμφανίζει διαφορετικό αριθμό cache misses. Με βάση τις μετρήσεις μας παρατηρούμε ότι η συγκεκριμένη εφαρμογή εμφανίζει χαμηλό αριθμό αριθμητικών πράξεων και το ποσοστό των cache misses είναι πολύ χαμηλό και στα δύο είδη πυρήνων.

**Facesim.** Το facesim είναι ένα πρόγραμμα κινουμένων σχεδίων με στόχο τη ρεαλιστική απεικόνιση του ανθρώπινου προσώπου. Παρουσιάζει coarse-grained παραλληλισμό δεδομένων και χαμηλή διανηματική επικοινωνία.

Παρατηρούμε ότι εμφανίζει υψηλό συντελεστή σ1 και υψηλό συντελεστή σ4, δηλαδή είτε σε πολυνηματικές είτε σε μονονηματικές εκτελέσεις οι big πυρήνες επιταχύνουν αισθητά την ολοκλήρωσή του προγράμματος. Γενικά αποτελεί υπολογιστικά απαιτητική εφαρμογή με πολύ υψηλό ποσοστό floating-point εντολών. Με βάση τις μετρήσεις μας με χρήση native input sets τα L2 cache misses είναι υψηλά, άνω του 40%, και στα δύο είδη πυρήνων. Τα L1 cache misses διατηρούνται χαμηλά και στα δύο είδη πυρήνων.

**Ferret.** Το πρόγραμμα ferret αποτελεί μία υπολογιστικά απαιτητική εφαρμογή αναζήτησης ομοιοτήτων. Δέχεται ως είσοδο ένα σύνολο από εικόνες και τις συγκρίνει με μία βασική εικόνα. Χωρίζει κάθε εικόνα σε τμήματα και εξάγει συγκεκριμένους δείκτες χαρακτηριστικών. Στο τέλος επιλέγει τις εικόνες που μοιάζουν περισσότερο με τη βασική εικόνα, με βάση ένα συντελεστή. Υποστηρίζει pipeline παραλληλισμό και απαιτεί έντονη διανηματική επικοινωνία.

Με βάση τις μετρήσεις μας παρατηρούμε ότι εμφανίζει τους υψηλότερους συντελεστές σ2 ,σ3 και γενικά βλέπουμε κακούς δείκτες παραλληλοποίησης, σ4 και σ5 , που σημαίνει ότι πρέπει να αποφεύγουμε την 1 thread και τη multithreaded εκτέλεση της εφαρμογής σε little πυρήνες. Με βάση τις μετρήσεις μας την κατατάσουμε στις υπολογιστικά απαιτητικές εφαρμογές, με σημαντικό ποσοστό εκτέλεσης integer εντολών. Τα L2 cache misses του προγράμματος είναι πολύ χαμηλά και στα δύο είδη πυρήνων.

**Fluidanimate.** Το πρόγραμμα fluidanimate προσομοιώνει την κίνηση υγρών με στόχο την ικανοποίηση αναγκών σε animation εφαρμογές. Υποστηρίζει coarse-grained παραλληλισμό, static load balancing και απαιτεί χαμηλή διανηματική επικοινωνία.

Με βάση το instruction mix του προγράμματος παρατηρούμε ότι υπάρχει

πολύ μεγάλος αριθμός locks, τουλάχιστον όταν χρησιμοποιούνται οι simlarge είσοδοι. Είναι compute intensive πρόγραμμα με σημαντικό ποσοστό τόσο integer, όσο και floating-point εντολών. Εμφανίζει σχετικά υψηλό αριθμό L2 cache misses, άνω του 40%, και στα δύο είδη πυρήνων. Βλέπουμε ότι ο συντελεστής σ1 είναι υψηλός και δευτερευόντως ο σ4. Παρατηρούμε ακόμα ότι ο συντελεστής σ5 είναι σχεδόν 1, δηλαδή δεν υπάρχει διαφορά αν εκτελεστεί η εφαρμογή με 4 threads σε 4 little πυρήνες ή σε 2 big και σε 2 little πυρήνες.

**Streamcluster.** Το πρόγραμμα streamcluster πραγματοποιεί προσεγγίσεις για την εύρεση της βέλτιστης ομαδοποίησης από μία εισερχόμενη ροή δεδομένων. Ανήκει στην κατηγορία της εξόρυξης δεδομένων και έχει εφαρμογές στην αναγνώριση προτύπων και στα δίκτυα υπολογιστών.

Με βάση τις μετρήσεις μας παρατηρούμε ότι το πρόγραμμα εμφανίζει πολύ χαμηλούς συντελεστές σ1 και σ4. Με βάση τις μετρήσεις μας, η εφαρμογή θεωρείται υπολογιστικά απαιτητική και εμφανίζει υψηλό αριθμό floating-point εντολών. Όταν το τρέχουμε με είσοδο τα simlarge input sets οι big πυρήνες παρουσιάζουν L2 cache misses πάνω από 75%, ενώ οι little πυρήνες πάνω από 80%. Τέλος, ο πολύ μεγάλος αριθμός barriers, φαίνεται ότι μετριάζει τις δυνατότητες των out-of-order επεξεργαστών κατά τον παραλληλισμό.

**Swaptions.** Μία ακόμα υπολογιστική εφαρμογή στον τομέα των οικονομικών είναι το swaptions. Υποστηρίζει coarse-granular παραλληλισμό, load static balancing των εργασιών ενώ απαιτεί μικρή διανηματική επικοινωνία.

Εμφανίζει μεσαίους συντελεστές σ1 και σ4, δηλαδή η επιτάχυνση της εκτέλεσης του προγράμματος στους big πυρήνες περίπου υποδιπλασιάζει τους χρόνους εκτέλεσης. Παρόλα αυτά παρατηρούμε ότι ο συντελεστής σ5 είναι περίπου 1, που σημαίνει ότι αν η εφαρμογή δεν εκτελεστεί ταυτόχρονα και στους 4 big πυρήνες, αλλά εκτελεστεί σε 2 big και σε 2 little πυρήνες, δεν θα υπάρχει ουσιαστική επιτάχυνση. Το ποσοστό των L1 cache misses του προγράμματος θεωρείται χαμηλό, ενώ το ποσοστό των L2 cache misses του είναι σχεδόν μηδενικό. Επιπρόσθετα, το πρόγραμμα swaptions είναι υπολογιστικά απαιτητική εφαρμογή με μεγάλο ποσοστό integer εντολών.

**Vips.** Η εφαρμογή vips εφαρμόζει σειρά μετασχηματισμών σε εικόνες, επιτελώντας βασικές λειτουργίες επεξεργασίας εικόνας. Υποστηρίζει medium-granular παραλληλισμό και dynamic load balancing.

Παρουσιάζει τον υψηλότερο συντελεστή σ1 και πολύ υψηλό συντελεστή σ4, αλλά γενικότερα και οι 6 συντελεστές του είναι υψηλοί. Σίγουρα ο πολύ μικρός αριθμός shared δεδομένων φαίνεται ότι ευνοεί τον παραλληλισμό. Με βάση τις μετρήσεις μας, με χρήση native input sets, το πρόγραμμα εμφανίζει σημαντικό ποσοστό τόσο integer, όσο και floating-point εντολών. Με βάση τις μετρήσεις μας με χρήση simlarge input sets, παρατηρούμε ότι τα L2 cache misses στους big πυρήνες είναι 40,66%, ενώ στους little πυρήνες φτάνουν το

ποσοστό 82,67%.

**x264.** Το πρόγραμμα x264 κωδικοποιεί video δεχόμενο ως είσοδο μία ακολουθία διαδοχικών εικόνων. Τα συμπιέζει με τρεις διαφορετικούς τρόπους εκμεταλλευόμενο τη συνοχή που παρουσιάζουν τα τμήματα κάθε μεμονωμένης εικόνας μεταξύ τους, αλλά και με τα προηγούμενα στιγμιότυπα στην ίδια θέση. Εμφανίζει coarse-granular παραλληλισμό με pipeline, δέχεται μεσαίου μεγέθους working sets και απαιτεί υψηλή διανηματική επικοινωνία.

Ο συντελεστής σ1 της εφαρμογής είναι σχετικά χαμηλός. Οι συντελεστές παραλληλοποίησης, σ2 και σ3, βλέπουμε ότι είναι υψηλοί. Με την αύξηση του αριθμού των threads το σύστημα κλιμακώνει πάρα τις εξαρτήσεις μεταξύ των threads. Βλέπουμε, ακόμα, ότι στις multithreaded εκτελέσεις του προγράμματος οι big πυρήνες δεν επιταχύνουν σημαντικά την εκτέλεσή του σε σύγκριση με τους little. Λόγω σφαλμάτων δε λάβαμε μετρήσεις, με το binary tool mambo, από το συγκεκριμένο πρόγραμμα. Γι΄ αυτόν το λόγο δεν έχουμε λάβει πληροφορίες για το instruction mix και τα cache misses του x264.

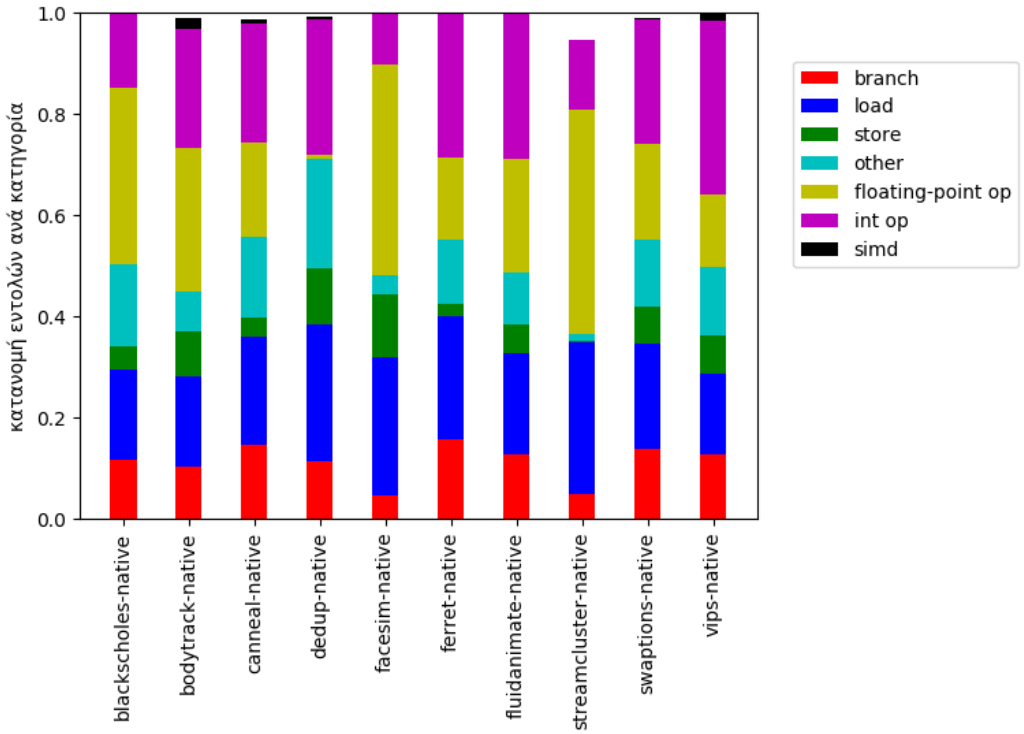
## 5.3 Instruction mix

Αφού περιγράψαμε σε αυτό το κεφάλαιο τα βασικά χαρακτηριστικά των Parsec προγραμμάτων, παραθέτουμε στη συνέχεια τα αποτελέσματα από μετρήσεις μας πάνω στο instruction mix των Parsec benchmarks. Σε όσα benchmarks δεν υπάρχουν επιμέρους αποτελέσματα πάνω στο instruction mix τους είναι λόγω αδυναμίας σωστής εκτέλεσής τους με το mambo. Στην Εικόνα 5.5 παρουσιάζουμε την κατανομή των εντολών για τα native input sets των Parsec προγραμμάτων και στην Εικόνα 5.6 για simlarge input sets.

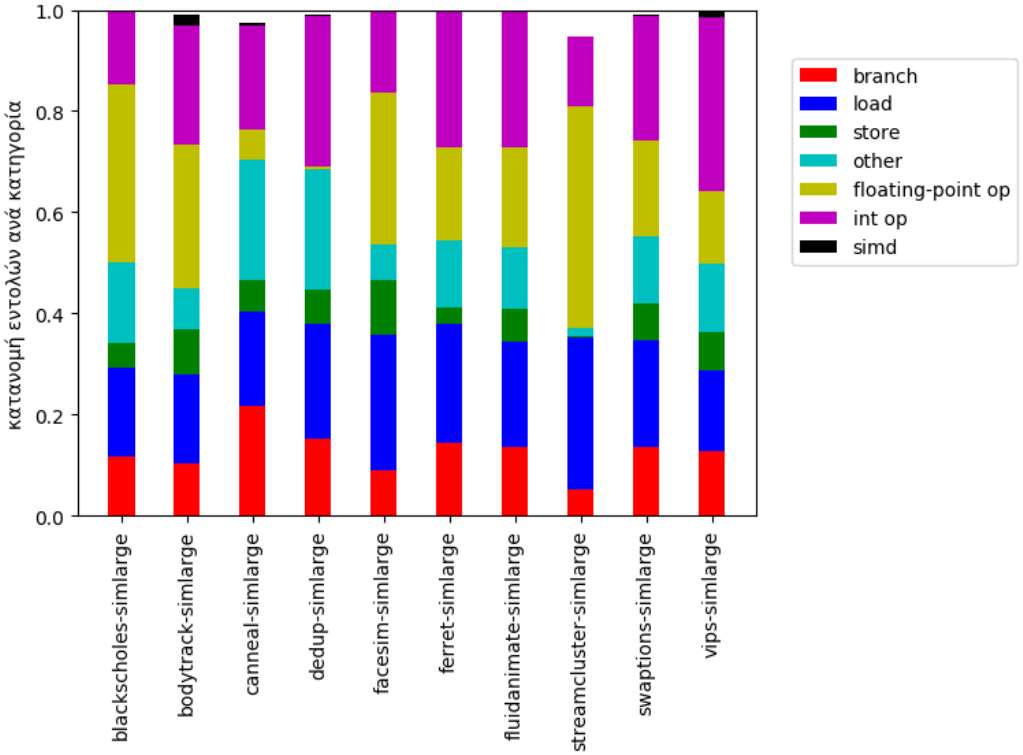
## 5.4 Συμπεριφορά των μνημών caches

Κατ'αρχάς αναφέρουμε ότι οι μετρήσεις στις cache μνήμες μας έχουν πραγματοποιηθεί σε εκτελέσεις των τεσσάρων threads πάνω σε τέσσερις πυρήνες κάθε είδους πυρήνα ξεχωριστά. Σε όσα benchmarks δεν υπάρχουν επιμέρους αποτελέσματα είναι λόγω σφαλμάτων εκτέλεσης τους με το mambo. Σημειώνουμε ότι και στη συγκεκριμένη μετρική χρησιμοποιήσαμε τόσο τα native όσο και τα simlarge input sets.

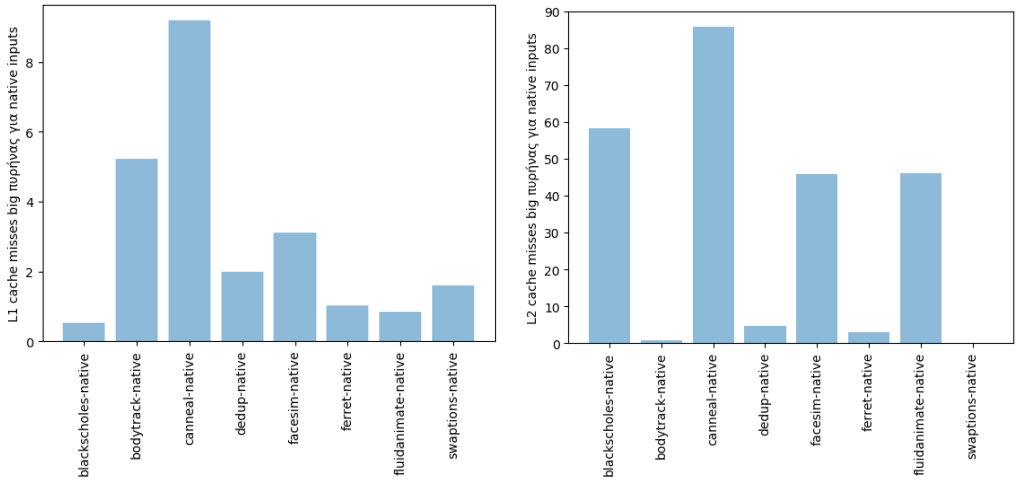
Με βάση τις μετρήσεις μας παρατηρούμε ότι σε κάποια benchmarks τα L2 cache misses των little πήυρνων είναι υψηλότερα σε σχέση με τα αντίστοιχα των big πυρήνων. Ακριβώς, αυτή τη συμπεριφορά αναμέναμε, καθώς το μέγεθος της L2 cache των big πυρήνων είναι οκτώ φορές μεγαλύτερη σε σχέση με την αντίστοιχη των little πυρήνων. Τα cache misses στους big πυρήνες με LRU




Εικόνα 5.8: Αποτελέσματα instruction mix των Parsec προγραμμάτων για native input sets.



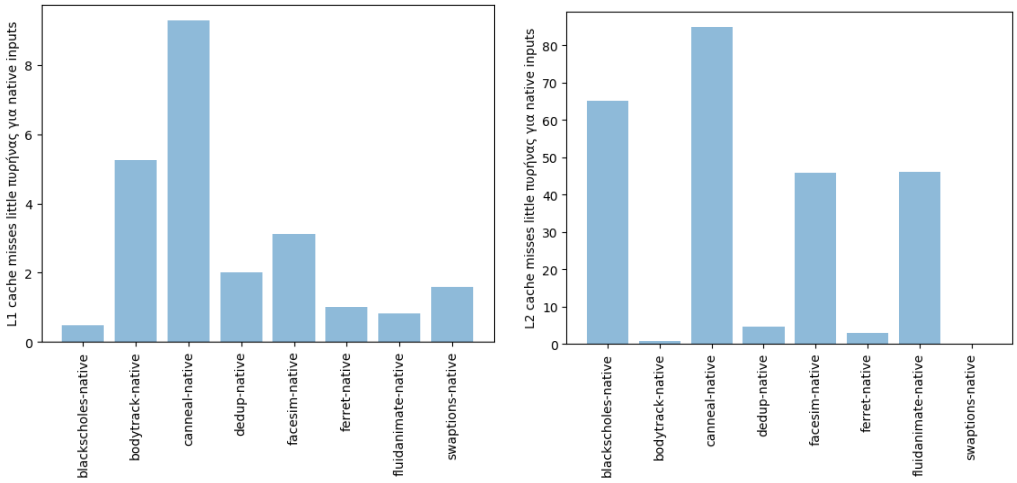

Εικόνα 5.9: Αποτελέσματα Instruction mix των Parsec προγραμμάτων για simlarge input sets.




(1) L1 data cache misses.

(2) L2 cache misses.

Εικόνα 5.10: Ποσοστό των misses στις L1 και L2 caches του big πυρήνα για τα Parsec benchmarks με native input sets.




(1) L1 data cache misses.

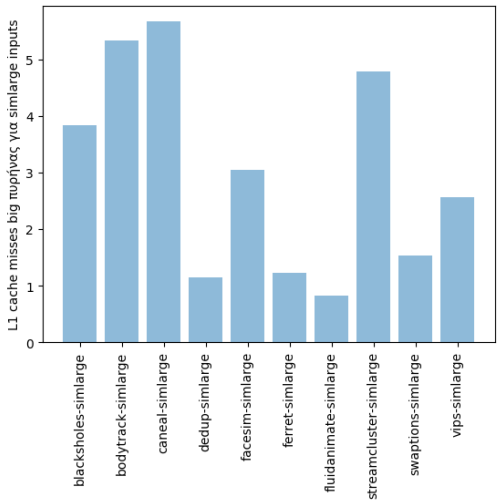
(2) L2 cache misses.

Εικόνα 5.11: Ποσοστό των misses στις L1 και L2 caches του little πυρήνα για τα Parsec benchmarks με native input sets.

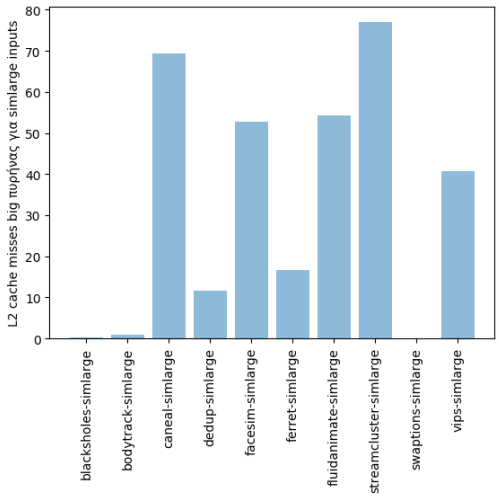
(Last Recently Used) δρομολόγηση πρέπει να είναι μικρότερα σε σχέση με τα αντίστοιχα των little πυρήνων.

## 5.5 Συσχετιστικά διαγράμματα

Όπως και στα Spec2006 προγράμματα, έτσι και στα Parsec κατασκευάσαμε συσχετιστικά διαγράμματα. Στόχος μας ήταν να παραστήσουμε τη συσχέτιση instruction mix και cache misses των προγραμμάτων με τους συντελεστές χρόνων εκτέλεσής τους. Τις μετρήσεις πάνω στο instruction mix και στα cache misses τις πραγματοποιήσαμε όταν εκτελούσαμε τα Parsec προγράμματα με 4 threads. Γι' αυτόν το λόγο χρησιμοποιήσαμε το συντελεστή σ4 στα συσχετιστικά διαγράμματα.

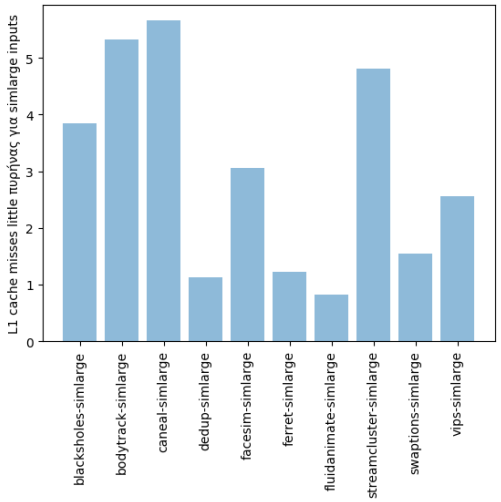



(1) L1 data cache misses.

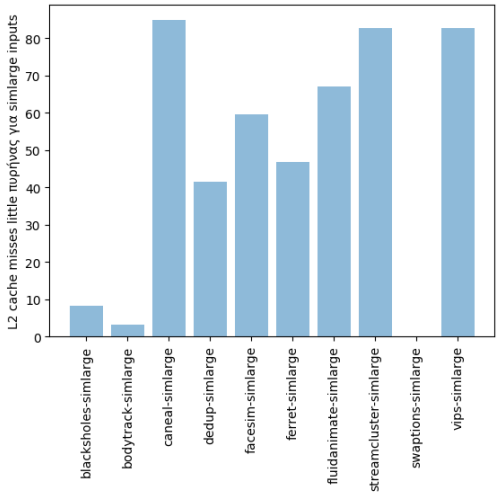



(2) L2 cache misses.

Εικόνα 5.12: Ποσοστό των misses στις L1 και L2 caches του big πυρήνα για τα Parsec benchmarks με simlarge input sets.

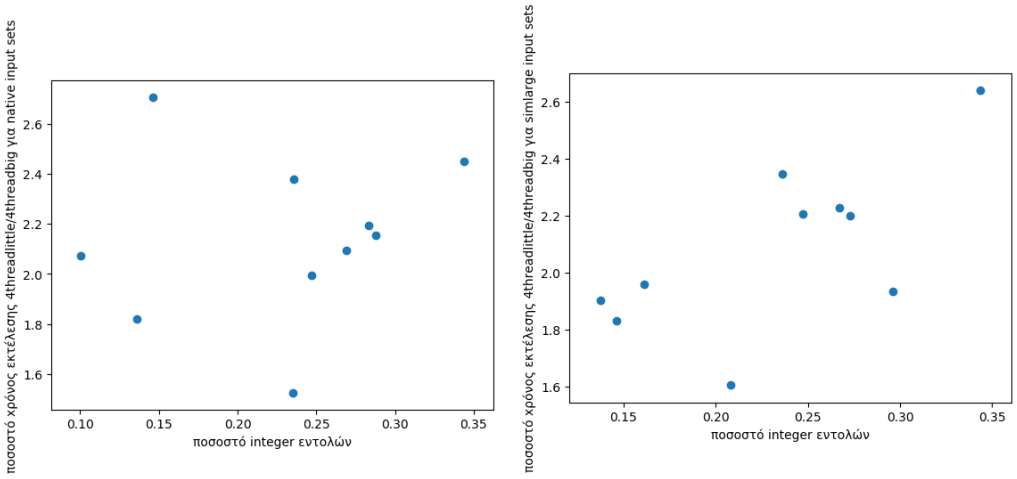



(1) L1 cache misses.



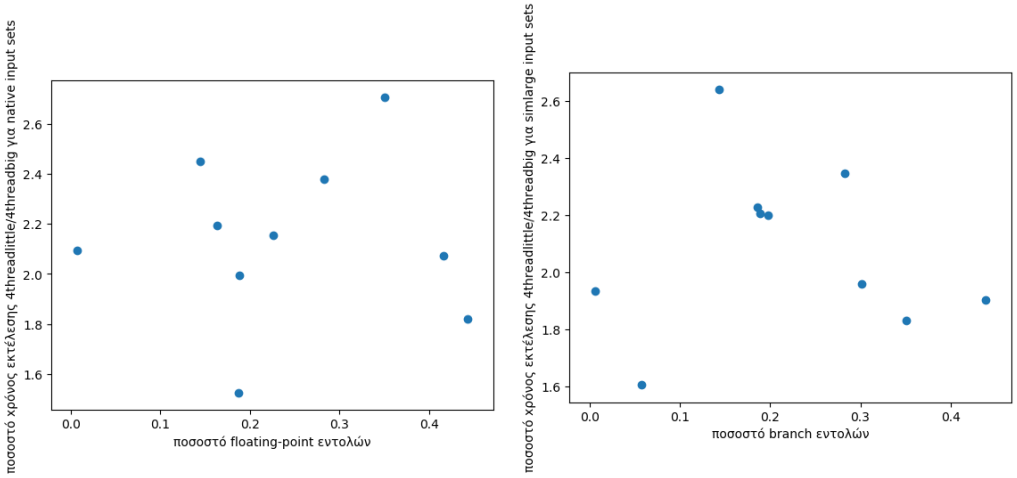

(2) L2 cache misses.

Εικόνα 5.13: Ποσοστό των misses στις L1 και L2 caches του little πυρήνα για τα Parsec benchmarks με simlarge input sets.



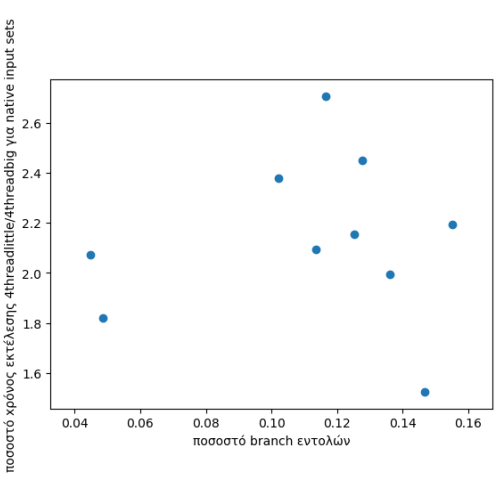

(1) Native input sets. (2) Simlarge input sets.

Εικόνα 5.14: Συσχετιστικό διάγραμμα integer εντολών και συντελεστή σ4.

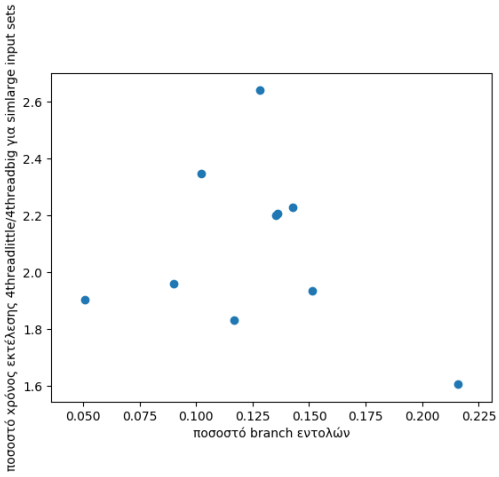



(1) Native input sets. (2) Simlarge input sets.

Εικόνα 5.15: Συσχετιστικό διάγραμμα floating-point εντολών και συντελεστή σ4.

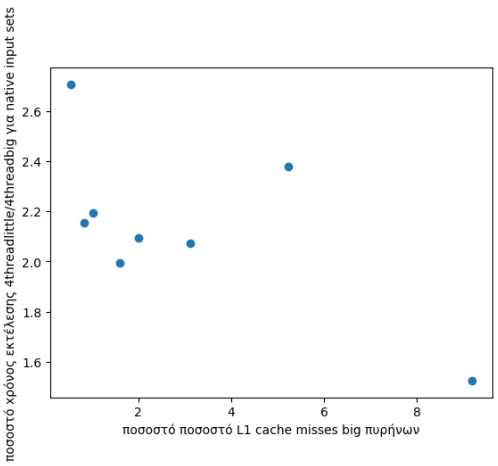



(1) Native input sets.

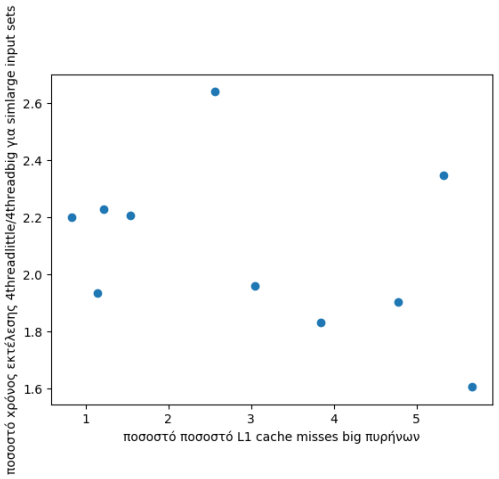



(2) Simlarge input sets.

Εικόνα 5.16: Συσχετιστικό διάγραμμα branch εντολών και συντελεστή σ4.

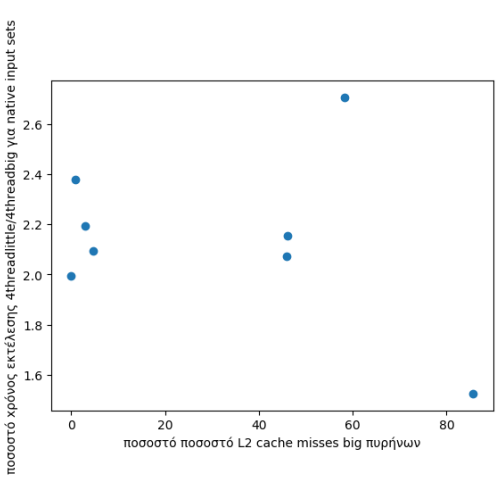



(1) Native input sets.

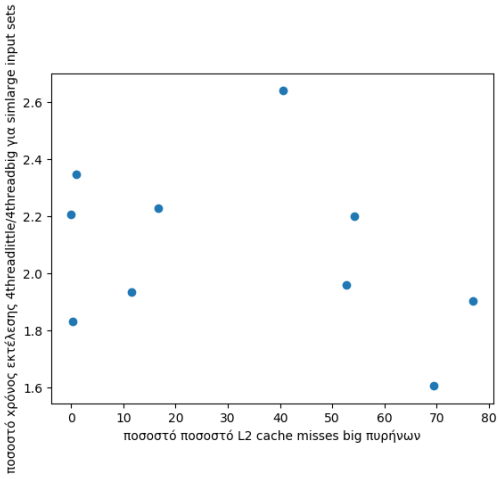



(2) Simlarge input sets.

Εικόνα 5.17: Συσχετιστικό διάγραμμα L1 cache misses και συντελεστή σ4 στους big πυρήνες.

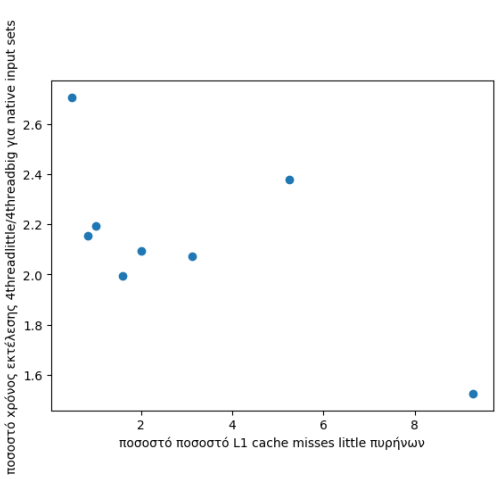



(1) Native input sets.

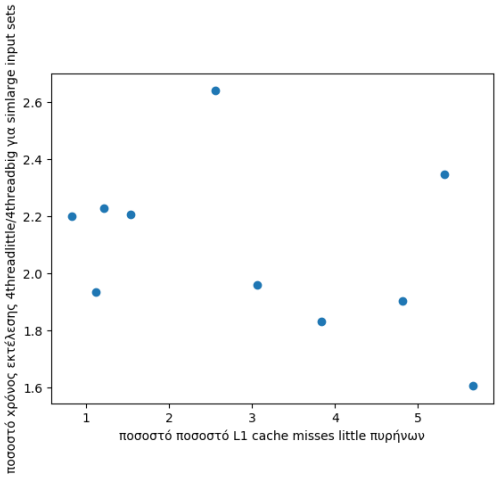



(2) Simlarge input sets.

Εικόνα 5.18: Συσχετιστικό διάγραμμα L2 cache misses και συντελεστή σ4 στους big πυρήνες.

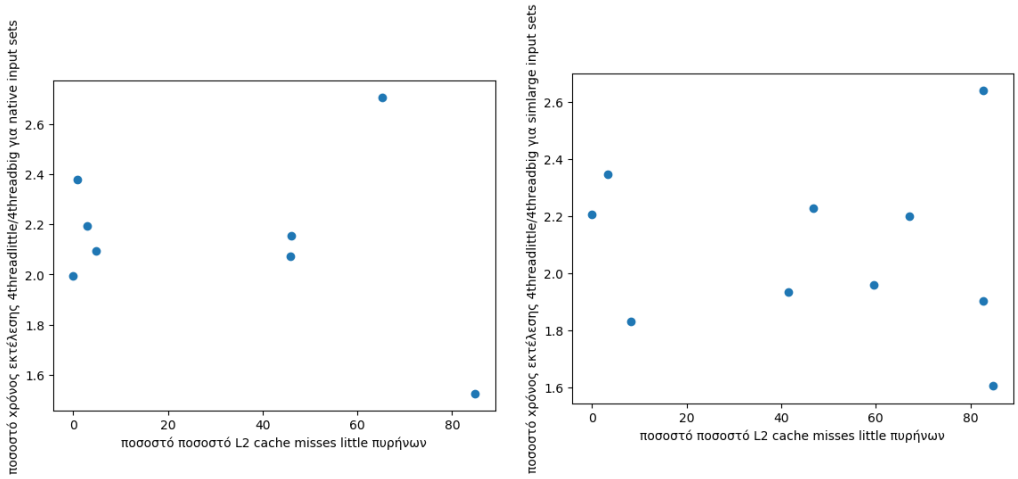



(1) Native input sets.




(2) Simlarge input sets.

Εικόνα 5.19: Συσχετιστικό διάγραμμα L1 cache misses και συντελεστή σ4 στους little πυρήνες.




(1) Native input sets.

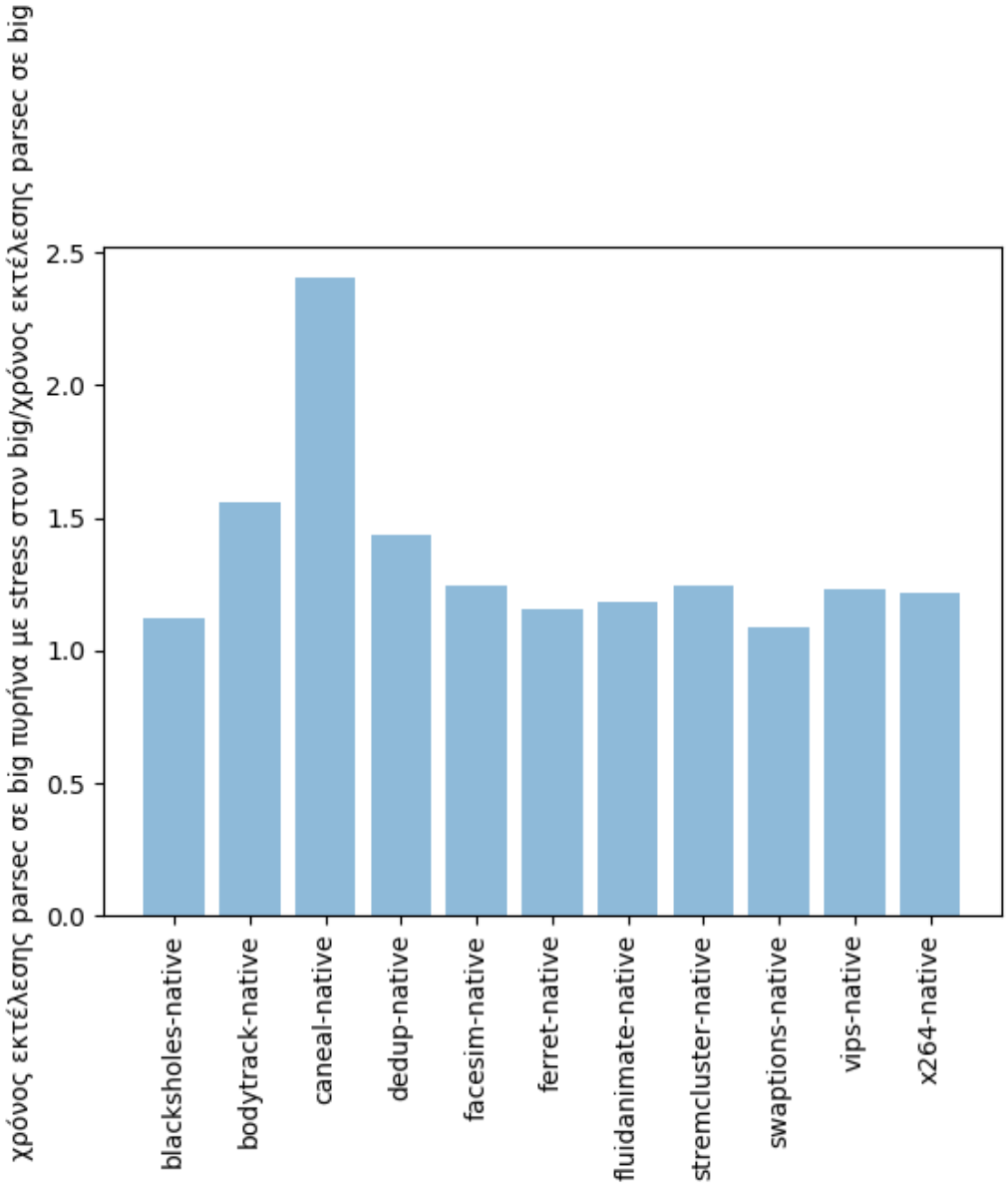
(2) Simlarge input sets.

Εικόνα 5.20: Συσχετιστικό διάγραμμα L2 cache misses και συντελεστή σ4 στους little πυρήνες.

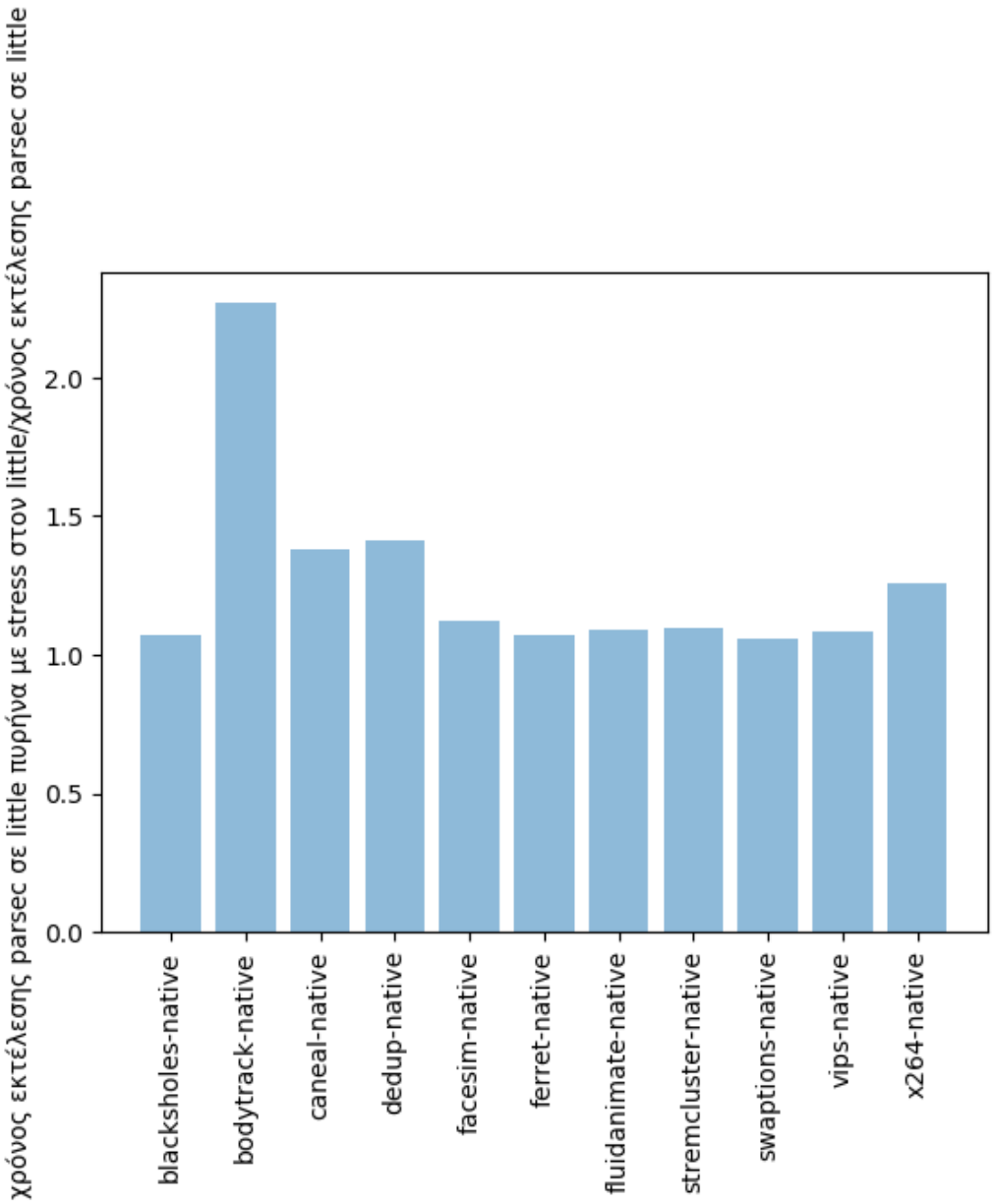
## 5.6 Συνεκτέλεση Parsec και stress προγραμμάτων

Σε αυτήν τη φάση της ερευνάς μας αναθέσαμε την εκτέλεση ενός stress προγράμματος σε έναν big ή little πυρήνα και σε έναν άλλον πυρήνα, ίδιου είδους, "τρέχαμε" διαδοχικά τα Parsec προγράμματά μας. Σε αυτό το υποκεφάλαιο παραθέτουμε τα αποτελέσματα όταν χρησιμοποιούμε τα native input sets. Σχετικά με τις μετρήσεις με χρήση simlarge input sets, θεωρήσαμε ότι δεν ήταν αντιπροσωπευτικές της πραγματικότητας και γι' αυτόν το λόγο δεν παρουσιάζουμε τις μετρήσεις με χρήση simlarge εισόδων. Στις εικόνες 5.21 και 5.22 βλέπουμε τα διαγράμματα αυτής της κατηγορίας.

Στα συγκεκριμένα πειραματικά αποτελέσματα εξετάζουμε τη συμπεριφορά των Parsec προγραμμάτων όταν μοιράζονται την L2 cache με το stress πρόγραμμα. Με βάση τις μετρήσεις μας παρατηρούμε ότι στους big πυρήνες το πρόγραμμα canneal εμφανίζει τη χειρότερη συμπεριφορά. Όπως έχουμε αναφέρει, το canneal είναι ένα memory intensive πρόγραμμα με χαμηλούς χρόνους παραλληλισμού και όπως αναμενόταν εμφανίζει αρνητική συμπεριφορά κατά τον παραλληλισμό του με το stress πρόγραμμα. Αντίστοιχα στους little πυρήνες τη χειρότερη συμπεριφορά εμφανίζει το πρόγραμμα bodytrack, το οποίο γενικά εμφανίζει χαμηλούς χρόνους παραλληλισμού στους little πυρήνες. Αξίζει να σημειώσουμε ότι τα συγκεκριμένα αποτελέσματα δεν είναι αντιπροσωπευτικά για τα Parsec προγράμματα, καθώς εδώ τα τρέχουμε με μόνο ένα thread.




Εικόνα 5.21: Χρόνος εκτέλεσης μονονηματικού Parsec σε big πυρήνα, όταν τρέχει παράλληλα με stress πρόγραμμα στο ίδιο σύμπλεγμα πυρήνων, ως προς τον χρόνο εκτέλεσης Parsec, με 1 thread, σε big πυρήνα όταν τρέχει μόνο του.




Εικόνα 5.22: Χρόνος εκτέλεσης μονονηματικού Parsec σε little πυρήνα, όταν τρέχει παράλληλα με stress πρόγραμμα στο ίδιο σύμπλεγμα πυρήνων, ως προς τον χρόνο εκτέλεσης Parsec, με 1 thread, σε little πυρήνα όταν τρέχει μόνο του.

Στη συνέχεια αναθέσαμε την εκτέλεση ενός stress προγράμματος σε έναν big ή little πυρήνα και σε έναν άλλον πυρήνα, διαφορετικού είδους, "τρέχαμε" διαδοχικά τα Parsec προγράμματά μας. Ο στόχος μας ήταν να παρατηρήσουμε τη συμπεριφορά των Parsec προγραμμάτων όταν χρησιμοποιούν μαζί με το stress πρόγραμμα τα buses, τους row buffers και τα banks της κύριας μνήμης. Και στις συγκεκριμένες μετρικές παραθέτουμε μόνο τα αποτελέσματα των μετρήσεων με native input sets. Στις εικόνες 5.23 και 5.24 βλέπουμε τα διαγράμματα της συγκεκριμένης κατηγορίας.

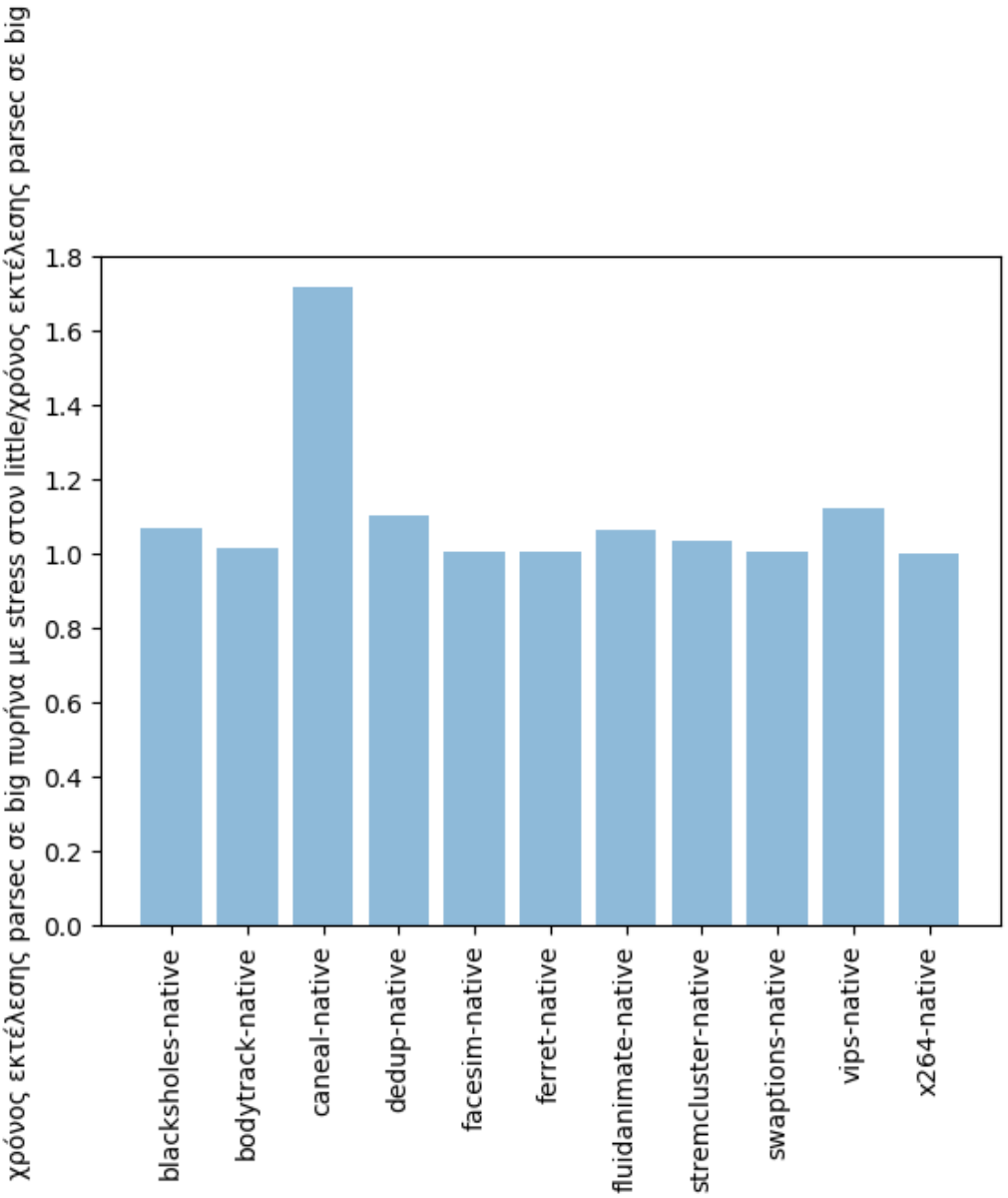
Στο συγκεκριμένο είδος μετρήσεων τη χειρότερη συμπεριφορά εμφανίζει το πρόγραμμα bodytrack και δευτερευόντως τα προγράμματα canneal και dedup, τόσο όταν εκτελούνται στους big, όσο και όταν εκτελούνται στους little πυρήνες. Δεν έχουμε μετρήσει τα cache misses όταν τα Parsec προγράμματα εκτελούνται με ένα thread. Όταν, όμως, εκτελείται με τέσσερα thread το πρόγραμμα bodytrack εμφανίζει σχεδόν μηδενικά L2 cache misses, με αποτέλεσμα να μην αναμένεται η συμπεριφορά αυτή. Οι προσβάσεις του στην κύρια μνήμη θεωρούνται οι ελάχιστες δυνατές.

## 5.7 Ενεργειακές μετρήσεις στα Parsec προγράμματα

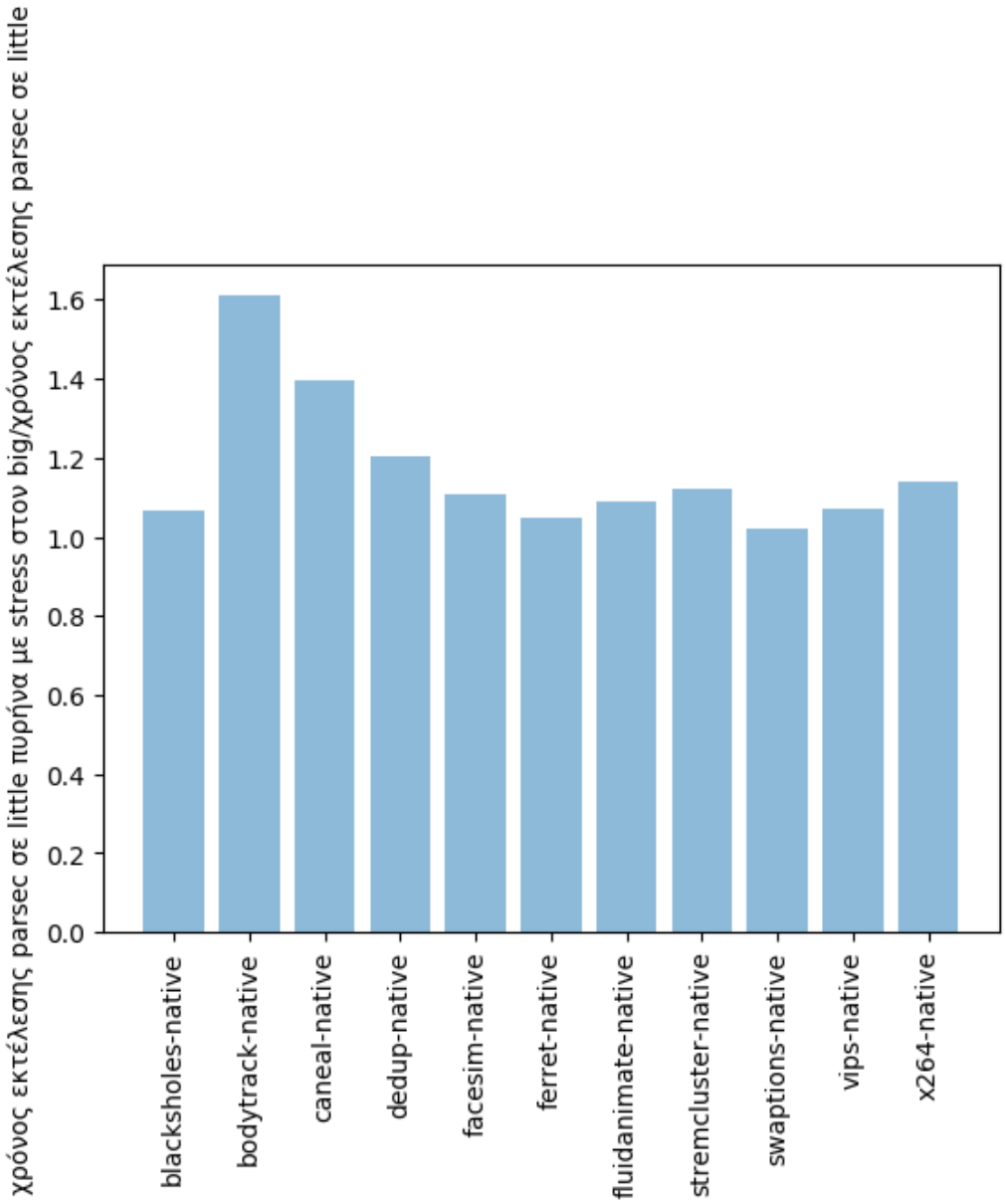
Πραγματοποιήσαμε μετρήσεις πάνω στην ενεργειακή κατανάλωση των Parsec προγραμμάτων με βάση τη μεθοδολογία που έχουμε αναλύσει. Πραγματοποιήσαμε μετρήσεις για την εκτέλεση μονονηματικών και τεσσάρων threads πάνω στα Parsec προγράμματα. Τα αποτελέσματα των μονονηματικών εκτελέσεων των Parsec προγραμμάτων βρίσκονται στις Εικόνες 5.25 και 5.26. Τα αποτελέσματα των εκτελέσεων των Parsec προγραμμάτων με 4 threads βρίσκονται στις Εικόνες 5.27 και 5.28.

Με βάση τις μετρήσεις μας παρατηρούμε ότι στις περισσότερες μονονηματικές εκτελέσεις Parsec προγραμμάτων, οι little πυρήνες εμφανίζουν χαμηλότερη ενεργειακή κατανάλωση. Σίγουρα ο χρόνος εκτέλεσης στο κάθε είδος πυρήνα καθορίζει την ενεργειακή κατανάλωση. Όσο περισσότερο επιβραδύνεται η εκτέλεση ενός προγράμματος στους little, τόσο αυξάνεται η ενεργειακή κατανάλωση. Επιπλέον το instruction mix, οι RAW εξαρτήσεις και τα cache misses του κάθε επιμέρους Parsec προγράμματος επηρεάζουν την ενεργειακή κατανάλωση του επεξεργαστή.

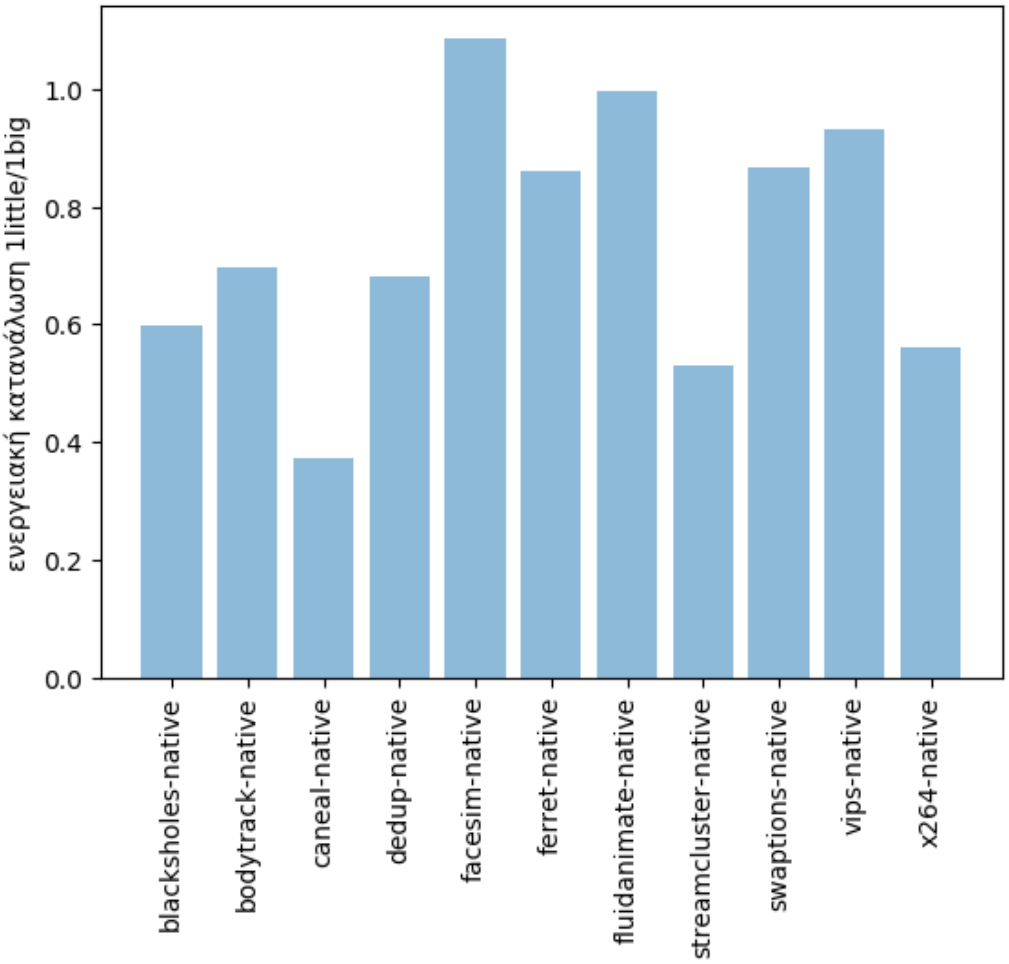
Στις περισσότερες εκτελέσεις Parsec προγραμμάτων με 4 threads παρατηρούμε διαφορετική συμπεριφορά. Τα περισσότερα προγράμματα εμφανίζουν καλύτερη ενεργειακή συμπεριφορά στους big πυρήνες. Παρατηρούμε ότι προγράμματα με υψηλό συντελεστή σ4 ευνοούν την εκτέλεση των προγραμμάτων σε big πυρήνες



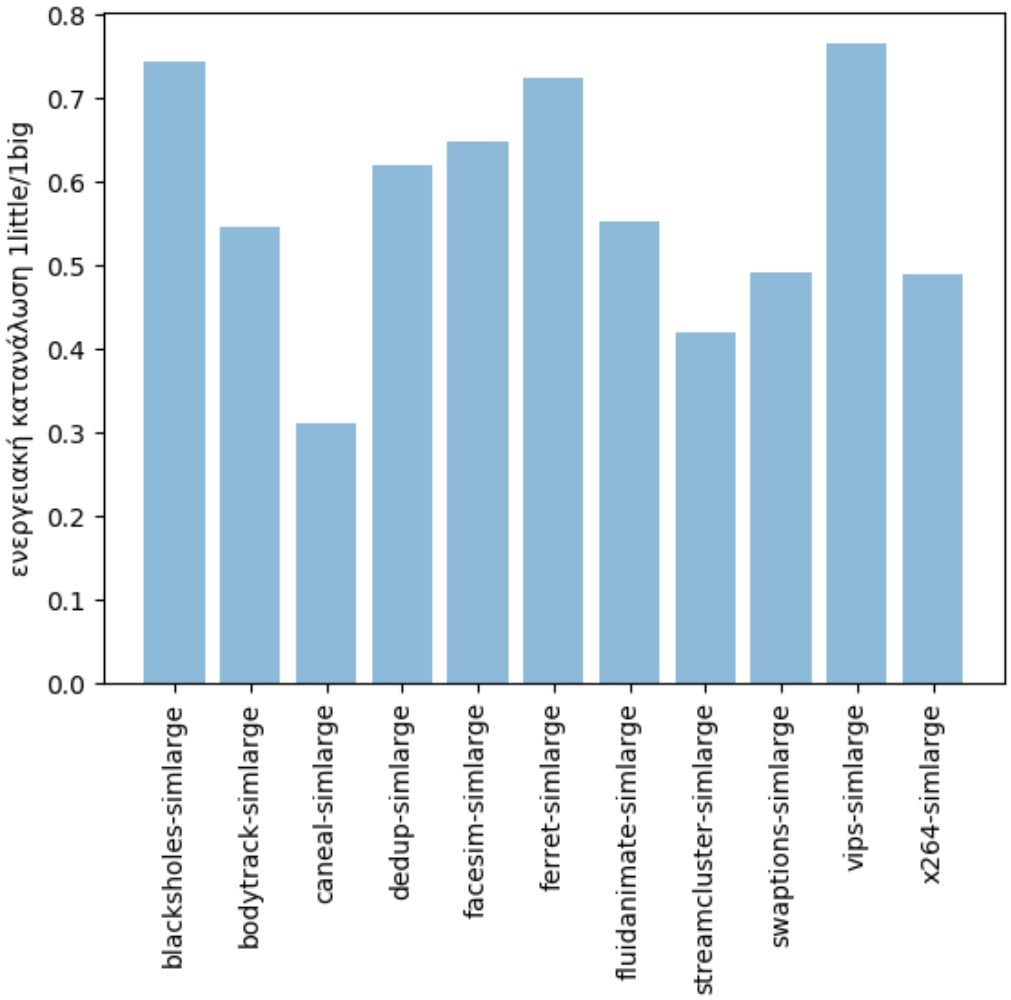

Εικόνα 5.23: Χρόνος εκτέλεσης μονονηματικού Parsec σε big πυρήνα, όταν τρέχει παράλληλα με stress πρόγραμμα σε διαφορετικό σύμπλεγμα πυρήνων, ως προς τον χρόνο εκτέλεσης Parsec, με 1 thread, σε big πυρήνα όταν τρέχει μόνο του.



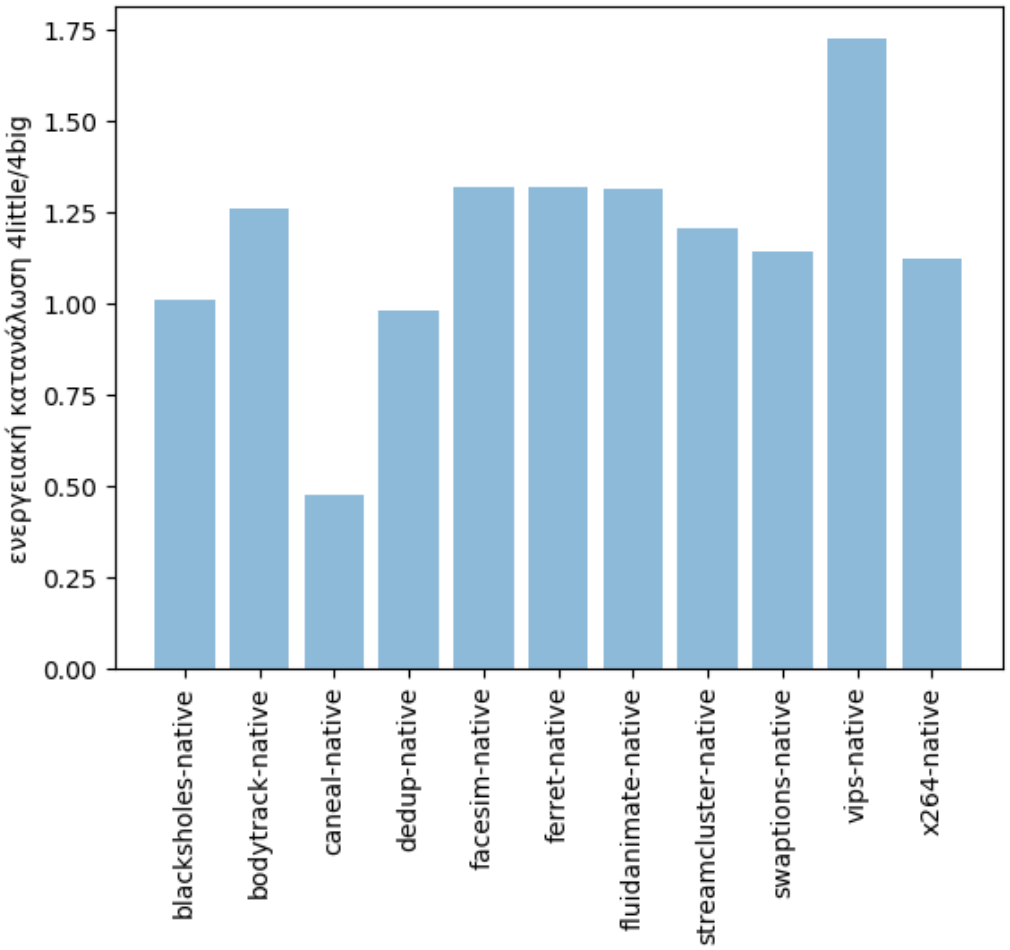

Εικόνα 5.24: Χρόνος εκτέλεσης μονονηματικού Parsec σε little πυρήνα, όταν τρέχει παράλληλα με stress πρόγραμμα σε διαφορετικό σύμπλεγμα πυρήνων, ως προς τον χρόνο εκτέλεσης Parsec, με 1 thread, σε little πυρήνα όταν τρέχει μόνο του.



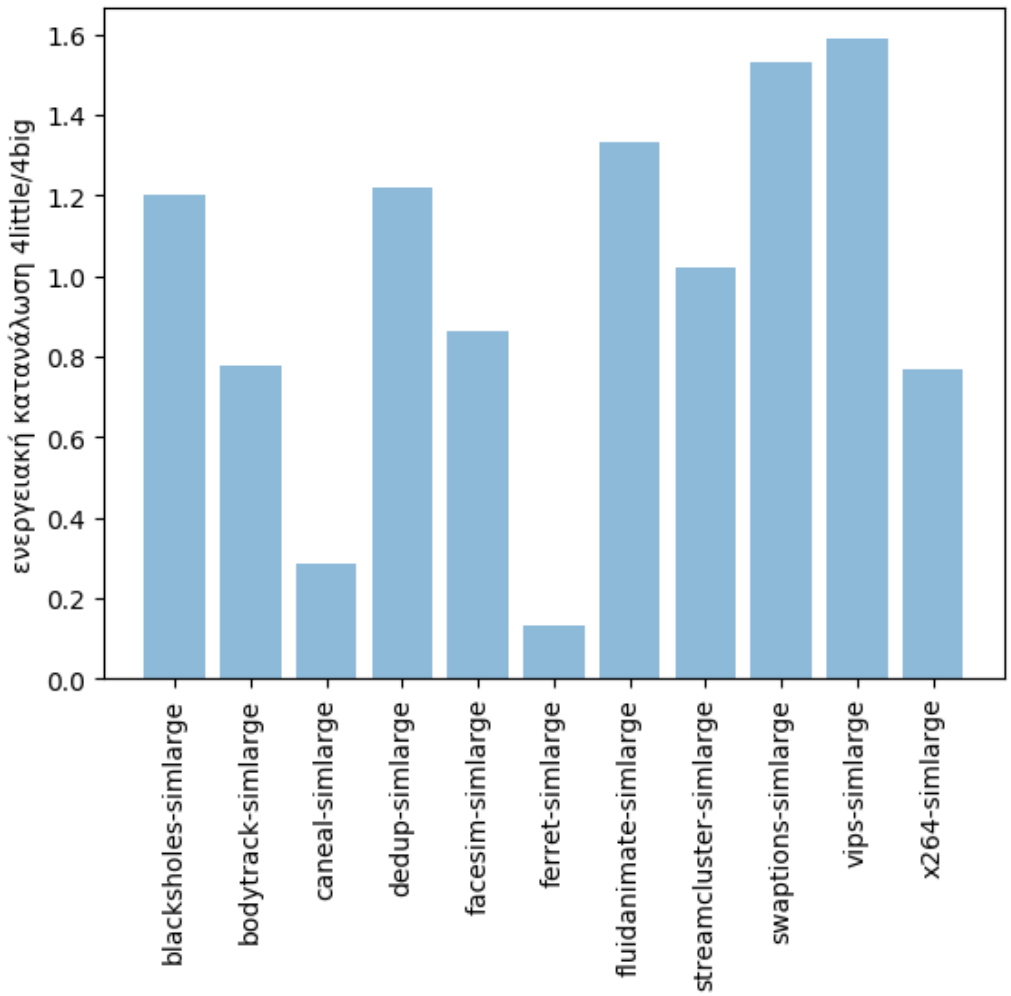

Εικόνα 5.25: Ενεργειακή κατανάλωση των μονονηματικών Parsec προγραμμάτων με native input sets.




Εικόνα 5.26: Ενεργειακή κατανάλωση των μονονηματικών Parsec προγραμμάτων με simlarge input sets.



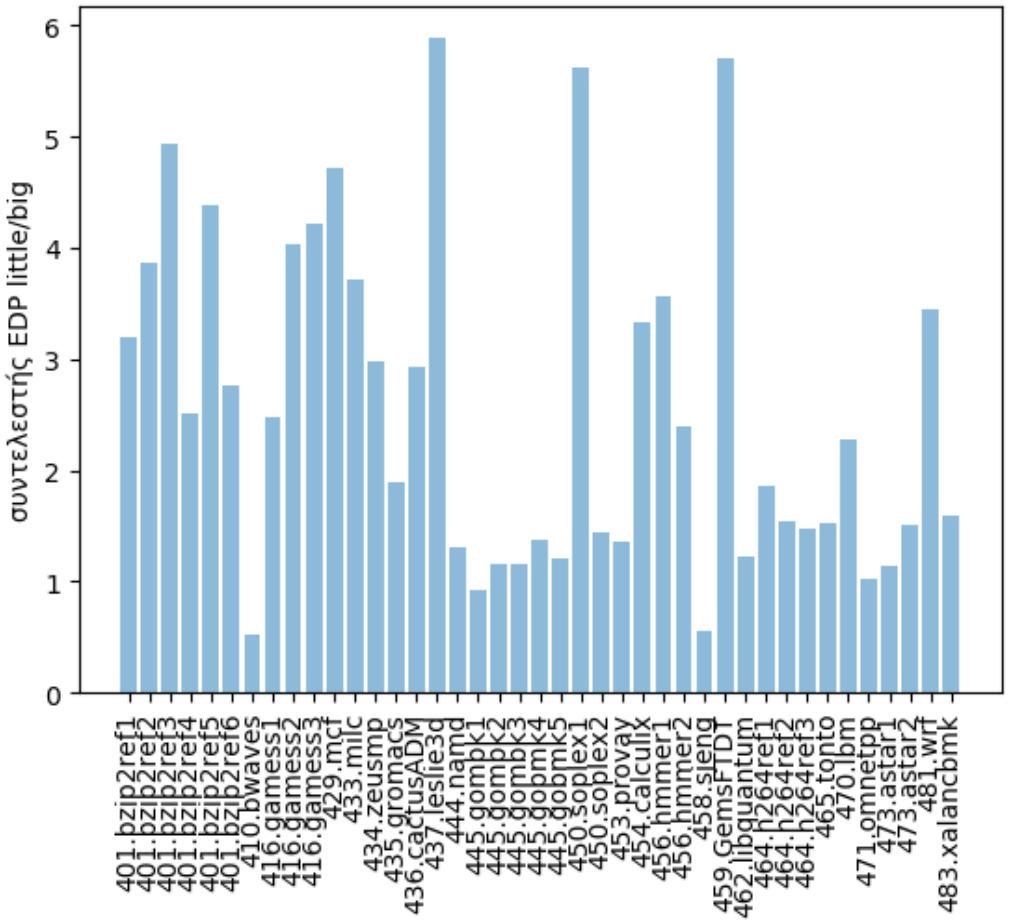

Εικόνα 5.27: Ενεργειακή κατανάλωση Parsec προγραμμάτων όταν εκτελούνται με 4 threads και native input sets.



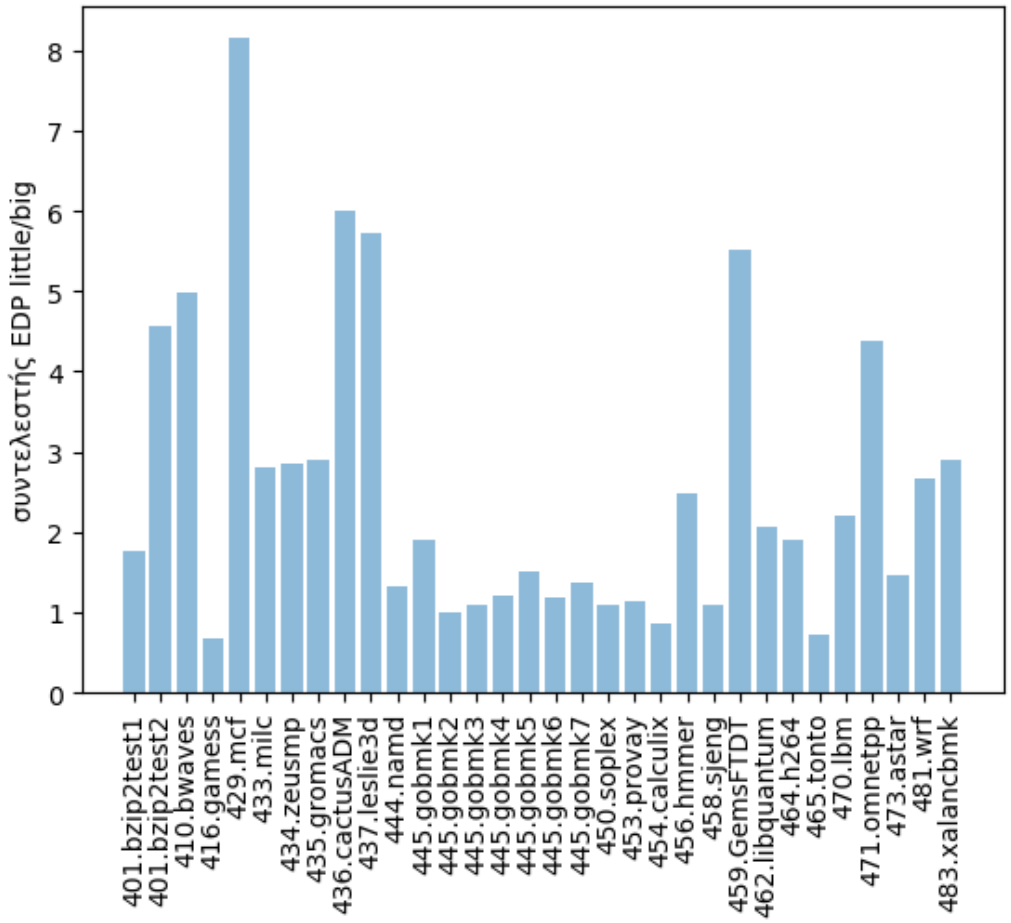

Εικόνα 5.28: Ενεργειακή κατανάλωση Parsec προγραμμάτων όταν εκτελούνται με 4 threads και simlarge input sets.

από πλευράς ενεργειακής κατανάλωσης.

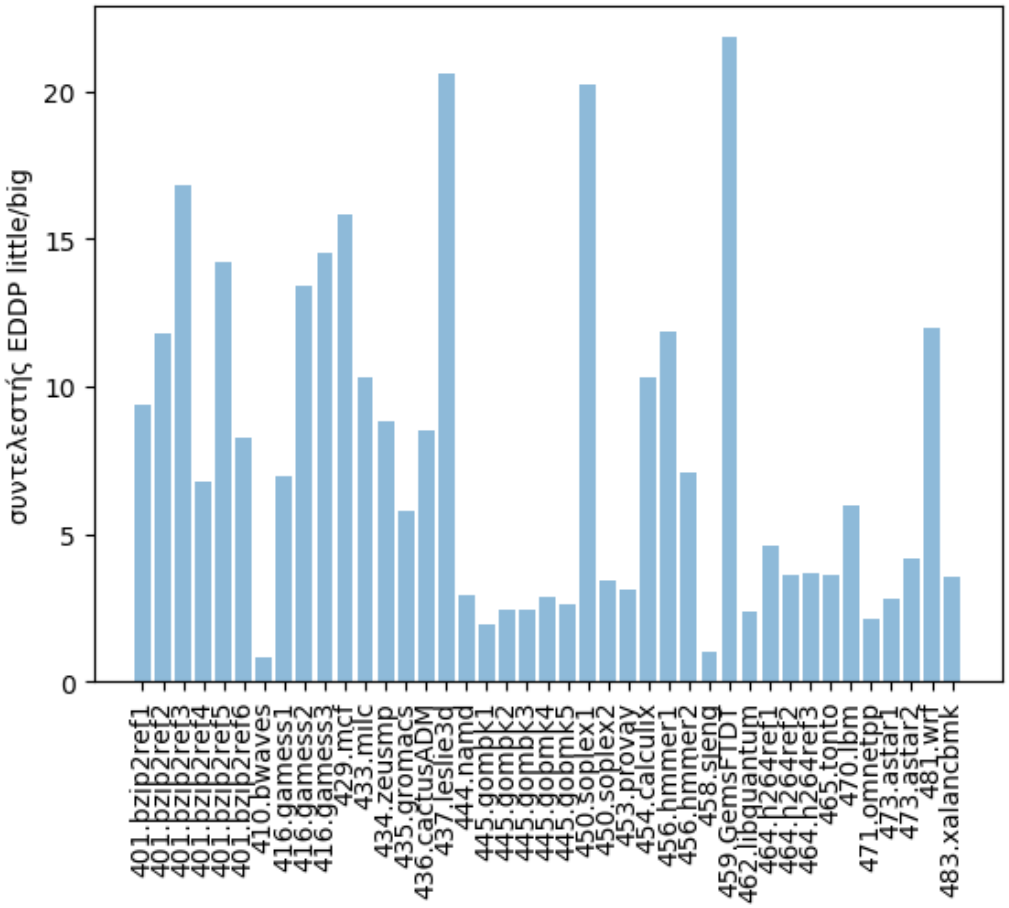
Αφού πραγματοποιήσαμε χρονικές και ενεργειακές μετρήσεις πάνω στα μονονηματικά Parsec benchmarks, στη συνέχεια δημιουργήσαμε διαγράμματα με βάση τους συντελεστές EDP, $ED^2P$ για native και για simlarge εισόδους. Τα συγκεκριμένα διαγράμματα βρίσκονται στις εικόνες 5.29, 5.30, 5.31 και 5.32. Έχουμε χρησιμοποιήσει αυτές τις μετρικές για τη μοντελοποίηση των μονονηματικών Parsec προγραμμάτων στο αντίστοιχο κεφάλαιο. Στα διαγράμματα συγκρίνουμε τους EDP και $ED^2P$ των προγραμμάτων όταν εκτελούνται στα δύο είδη πυρήνων. Παρατηρούμε ότι η μεγάλη πλειοψηφία των προγραμμάτων παρουσιάζει καλύτερες τιμές EDP και $ED^2P$ στον big πυρήνα.



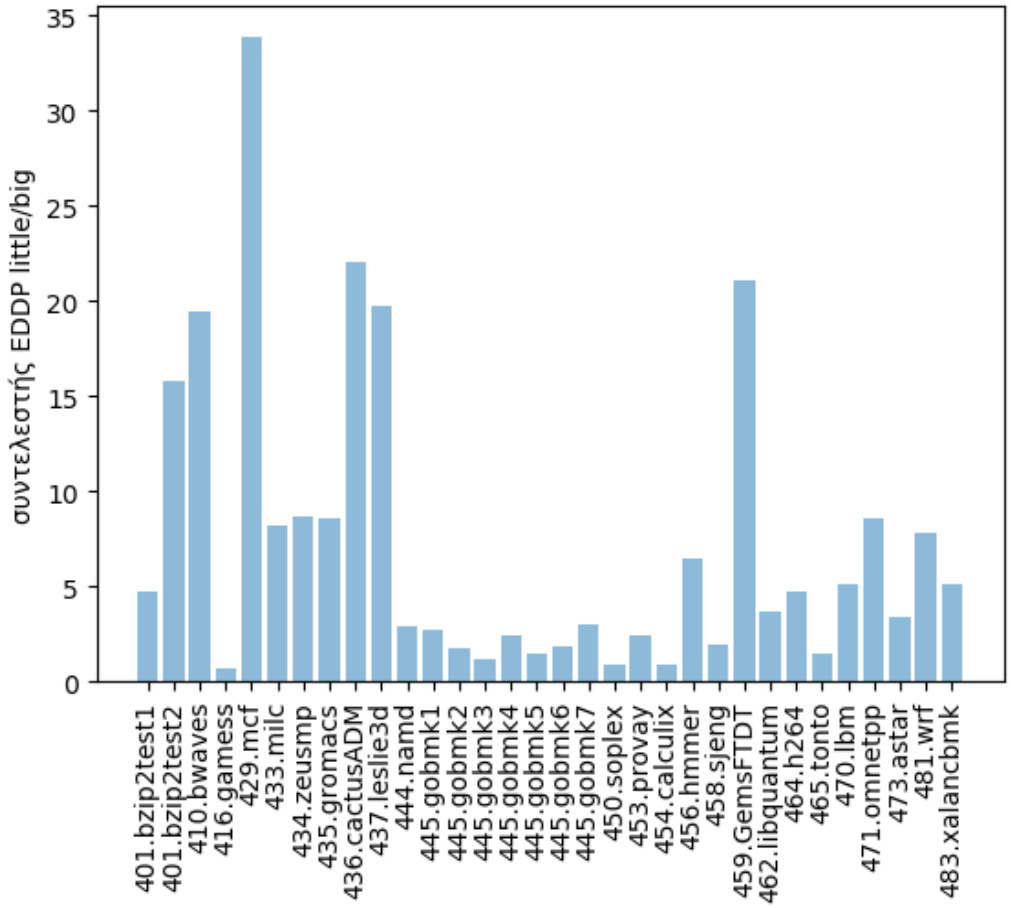

Εικόνα 5.29: Συντελεστές EDP των μονονηματικών Parsec προγραμματων με χρήση native input sets.




Εικόνα 5.30: Συντελεστές EDP των μονονηματικών Parsec προγραμμάτων με χρήση simlarge input sets.




Εικόνα 5.31: Συντελεστές $ED^2P$ των μονονηματικών Parsec προγραμματων με χρήση native input sets.




Εικόνα 5.32: Συντελεστές $ED^2P$ των μονονηματικών Parsec προγραμμάτων με χρήση simlarge input sets.

## 5.8 Συμπεράσματα ανάλυσης Parsec προγραμμάτων

Όπως αναφέραμε στην ανάλυση των Parsec προγραμμάτων, στα συγκεκριμένου τύπου προγράμματα εξετάζουμε νέες παραμέτρους πέρα από αυτές που ασχοληθήκαμε στα Spec2006 προγράμματα. Στα Spec2006 προγράμματα ενδιαφερόμασταν για τα L1 και L2 cache misses, για το instruction mix των εντολών, και για τα branch misspredictions. Στα Parsec προγράμματα μας ενδιαφέρει επιπλέον ο παραλληλισμός των προγραμμάτων, ο αριθμός των locks στα threads, τα barriers, το μέγεθος της κατάτμησης των προγραμμάτων καθώς και η ανάγκη για επικοινωνία μεταξύ των πυρήνων. Επιπλέον στα Parsec

προγράμματα εξετάζουμε τα cache misses και τις προσβάσεις στην κύρια μνήμη υπό λίγο διαφορετικό πρίσμα, λόγω του διαμοιρασμού της L2 μνήμης cache και της κύριας μνήμης από τα πολυνηματικά προγράμματα. Τέλος, αξίζει να σημειώσουμε ότι σημαντικό ρόλο στις εκτελέσεις των Parsec προγραμμάτων παίζει η διαθεσιμότητα των στοιχειωδών μονάδων εκτέλεσης εντολών.

Όσον αφορά την ενεργειακή κατανάλωση των Parsec προγραμμάτων, παρατηρούμε ότι προγράμματα με υψηλό συντελεστή σ1 ευνοούν την εκτέλεση των προγραμμάτων σε big πυρήνες. Μόνο στην περίπτωση του προγράμματος facesim, η ενεργειακή κατανάλωση στον big πυρήνα ήταν χαμηλότερη σε σχέση με την αντίστοιχη στον little. Ο χρόνος εκτέλεσης μίας εντολής είναι άρρηκτα συνδεδεμένος με την ενεργειακή της κατανάλωση. RAW (read after write) εξαρτήσεις προκαλούν αύξηση της ενεργειακής κατανάλωσης μίας εντολής. Αντίθετα στις περισσότερες εκτελέσεις Parsec προγραμμάτων με 4 threads τα περισσότερα προγράμματα εμφάνιζαν καλύτερη ενεργειακή συμπεριφορά στους big πυρήνες. Βλέπουμε ότι προγράμματα με υψηλό συντελεστή σ4 ευνοούν κατά βάση την εκτέλεση των προγραμμάτων στους ενεργειακά κοστοβόρους big πυρήνες.

# Κεφάλαιο 6

# Μοντελοποίηση επεξεργαστών ARM big.LITTLE με χρήση τεχνικών μηχανικής μάθησης

## 6.1 Εισαγωγή

Στόχος της μοντελοποίησή μας είναι να επιτύχουμε την ταχύτερη δυνατή εκτέλεση των προγραμμάτων σε συνδυασμό με την ελάχιστη δυνατή ενεργειακή κατανάλωση. Χρησιμοποιήσαμε τους συντελεστές EDP (Energy Delay Product) και $ED^2P$ (Energy Delay Delay Product), τους οποίους θα περιγράψουμε στη συνέχεια του κεφαλαίου. Ακόμα, επιχειρήσαμε να μοντελοποιήσουμε τα προγράμματα και με βάση τη συμπεριφορά τους όταν εκτελούνται παράλληλα με άλλα προγράμματα. Στόχος μας ήταν για τις εφαρμογές Spec2006 και Parsec του ενός thread, να μπορούμε να προβλέψουμε ποιο είδος πυρήνα επιτυγχάνει χαμηλότερο συντελεστή EDP ή $ED^2P$, ανάλογα με την περίπτωση. Αφού μορφοποιήσαμε τα πειραματικά αποτελέσματά μας, τα χρησιμοποιήσαμε σαν τιμές στόχους πάνω σε μοντέλα μηχανικής μάθησης.

## 6.2 Θεωρητικό Υπόβαθρο

Σε αυτήν την εργασία έχουμε υλοποιήσει διαφορετικά μοντέλα πρόβλεψης των συντελεστών EDP, $ED^2P$ και παράλληλης επίδοσης εφαρμογών μέσω της open-source διανομής της python anaconda. Στα μοντέλα αυτά έχουμε χρησιμοποιήσει ξεχωριστά τις μεθόδους logistic regression using stochastic gradient descent, decision tree, random forest, k nearest neighbours και το Multilayer Perceptron. Επιλέξαμε τα παραπάνω μοντέλα για δύο λόγους. Πρώτον, αποτελούν διαφορετικές μεθόδους machine learning και καλύπτουν διαφορετικά

εύρη learning αλγορίθμων είτε για γραμμικά είτε για μη γραμμικά προβλήματα. Δεύτερον οι συγκεκριμένες μέθοδοι παράγουν ντετερμινιστικές προβλέψεις, οι οποίες είναι κατάλληλες στο να προβλέψουν τους ζητούμενους συντελεστές.

Σε αυτό το σημείο αναλύουμε καθένα από τα παραπάνω μοντέλα πρόβλεψης:

### 6.2.1 Logistic Regression using Stochastic Gradient Descent

Ο αλγόριθμος αυτής της κατηγορίας προσπαθεί να δημιουργήσει μία γραμμική συνάρτηση πρόβλεψης τη μορφής $f(x) = w_0 + w_1 * x_1 + ... + w_n * x_n$ για κάθε σύνολο χαρακτηριστικών εισόδου $x_1, ..., x_n$. Στη συνέχεια η συνάρτηση f(x) χρησιμοποιείται ως είσοδος σε μία σιγμοειδή συνάρτηση για να προβλεφθεί η τιμή στόχος. Η σιγμοειδής συνάρτηση δίνει πάντα εξόδους στο διάστημα (0,1), οι οποίες αντιπροσωπεύουν τη πιθανότητα να κατηγοριοποιηθεί σε κάποια κλάση η τιμή στόχος. Συμβολίζουμε h(f(x)) τη σιγμοειδή συνάρτηση. Η συνάρτηση κόστους του μοντέλου έχει την εξής μορφή:

$$J = -y * \log h - (1 - y) * \log(1 - h) \tag{6.1}$$

όπου J η συνάρτηση κόστους για κάθε έξοδο, και y η μία τιμή στόχος.

Στην περίπτωση πολλαπλών τιμών στόχων η συνάρτηση κόστους έχει την εξής μορφή:

$$J = -\sum(y_i * logp_i) \tag{6.2}$$

όπου y διάνυσμα με μονάδα για την κλάση που αντιπροσωπεύει και μηδενικά στα υπόλοιπα στοιχεία του, $y_i$ η τιμή στόχος του διανύσματος y, $p_i$ η πιθανότητα να επιλεγεί η τιμή $y_i$.

Η πιθανότητα $p_i$ δίνεται σε αυτήν την κατηγορία από τη συνάρτηση :

$$p_i = softmax(f(x_i)) \tag{6.3}$$

$$softmax(z_j) = \frac{e^{z_j}}{\sum_1^K e^{z_k}}, \forall j \in 1, ..., k$$

Στη συνέχεια περνάμε στο κομμάτι του stochastic gradient descent, όπου με βάση τη σχέση

$$w_i new = w_i + \alpha * \frac{\partial J(w_i)}{\partial w_i}$$

ενημερώνονται τα βάρη κάθε m εισόδους, όπου m το μέγεθος του batch. Οι τιμές των βαρών μπορεί να ενημερωθούν είτε μετά από τον υπολογισμό των συνολικών απωλειών ενός υποσυνόλου του train data set (mini batch) ή ακόμα και ολόκληρου του δείγματος εισόδου (full batch). Ο συντελεστής α θεωρείται υπερπαράμετρος του δικτύου και ονομάζεται ρυθμός εκμάθησης του δείγματος (learning rate). Για να αποφευχθεί η υπερεκπαίδευση του δείγματος, ιδιαίτερα σε μεγάλα δείγματα εκπαίδευσης, προστίθενται στη συνάρτηση κόστους οι όροι $L1(\beta * \sum |w_j|)$, $L2(\beta * \sum {w_j}^2)$ είτε ένας από αυτούς είτε και οι δύο μαζί. Αφού εκπαιδευτεί το δείγμα με την παραπάνω διαδικασία, στη συνέχεια μπορεί να πραγματοποιήσει πρόβλεψη για κάθε δείγμα εισόδου.

### 6.2.2 Decision Trees

Τα δέντρα αποφάσεων είναι σε θέση να προβλέψουν μία τιμή στόχο (target value). Συμπεραίνουν κανόνες από τα data features (τιμές των χαρακτηριστικών των δεδομένων εισόδου) και με τα συγκεκριμένα αυτά αποτελέσματα δημιουργούν κατάλληλο δυαδικό δέντρο. Το γεγονός ότι δε χρειάζεται να κανονικοποιήσουμε τα δεδομένα εισόδου, θεωρείται ένα από τα πλεονεκτήματα των δέντρων αποφάσεων. Ο αλγόριθμος κατασκευάζει ένα δυαδικό δέντρο κόμβο-κόμβο εστιάζοντας σε ένα χαρακτηριστικό κάθε φορά και μία οριακή τιμή για το χαρακτηριστικό αυτό. Υπάρχουν διάφοροι αλγόριθμοι κατασκευής decision tree. Εμείς χρησιμοποιήσαμε τον αλγόριθμο ID3, ο οποίος χρησιμοποιεί την εντροπία για την εξαγωγή της ομοιογένειας του δείγματος. Με βάση το συντελεστή κέρδους επιλέγει σε κάθε κόμβο το χαρακτηριστικό που προκαλεί το μέγιστο κέρδος στο δέντρο. Σημειώνουμε ότι επιλέγεται για κάθε κόμβο το χαρακτηριστικό με το μεγαλύτερο κέρδος από πάνω προς κάτω. Περιγράφουμε αναλυτικά τις σχέσεις του αλγορίθμου:

**Εντροπία.** Η εντροπία $H(S_1)$ μετράει το μέγεθος της αβεβαιότητας σε ένα σύνολο $S_1$.

$$H(S_1) = -\sum_{c \in C} (p(c) * logp(c)) \tag{6.4}$$

όπου $S_1$ είναι το παρόν σύνολο δειγμάτων, C το σύνολο των κλάσεων τιμών στόχου, c η τιμή στόχος που επιλέγεται, και p(c) ο αριθμός των δειγμάτων με τιμή στόχο στην κατηγορία c, προς το συνολικό αριθμό δειγμάτων στο $S_1$.

**Κέρδος Πληροφορίας.** Κέρδος πληροφορίας IG(A) θεωρείται η διαφορά της εντροπίας ενός συνόλου S, όταν εκείνο διαχωριστεί με βάση τιμές της κατηγορίας A. Με άλλα λόγια κέρδος πληροφορίας είναι το πόσο μειώθηκε η αβεβαιότητα του S, όταν το διαχωρίσουμε με βάση τις τιμές της κατηγορίας A.

$$IG(A,S) = H(S) - \sum_{t \in T}(H(t) * p(t)) \tag{6.5}$$

όπου S είναι το παρόν σύνολο δειγμάτων, T είναι τα υποσύνολα t του S αφού διαχωριστεί με βάση την κατηγορία A, H(t) είναι η εντροπία του υποσυνόλου t, και p(t) ο αριθμός των δειγμάτων που ανήκουν στο υποσύνολο t προς το συνολικό αριθμό δειγμάτων στο S.

Τα δέντρα αποφάσεων μας παρέχουν τη δυνατότητα να κρίνουμε ποια ιδιότητα εισόδου είναι πιο σημαντική για τις αποφάσεις, με βάση την απόστασή της από τη ρίζα του δέντρου. Όσο πιο ψηλά στο δέντρο βρίσκεται μία κατηγορία εισόδου τόσο πιο σημαντική θεωρείται η τιμή της για την επιλογή του στόχου. Τα δέντρα είναι μη παραμετρικά και τείνουν να υπερεκπαιδεύονται πάνω στο training δείγμα, αν δεν επιβληθούν περιορισμοί κατά την κατασκευή τους. Μειώνοντας τους βαθμούς ελευθερίας του δέντρου αντιμετωπίζεται το πρόβλημα της υπερεκπαίδευσης, με τίμημα την αύξηση των σφαλμάτων. Κάποιες βασικές υπερπαράμετροι που ομαλοποιούν τις αποφάσεις των δέντρων αποφάσεων είναι το βάθος του δέντρου, ο ελάχιστος αριθμός φύλλων, καθώς και ο ελάχιστος αριθμός στοιχείων που απαιτούνται για να διασπαστεί ένας κόμβος.

### 6.2.3 Random Forest

Στη μέθοδο Random Forest πολλά δέντρα αποφάσεων εκπαιδεύονται στο παρασκήνιο από υποσύνολα του αρχικού συνόλου δειγμάτων εισόδου. Η εκτίμηση, μέσω της συγκεκριμένης μεθόδου, είναι ο μέσος όρος των εκτιμήσεων όλων των δέντρων παρασκηνίου. Η μεγάλη κατάτμηση του αρχικού δείγματος σε υποσύνολα προκαλεί μεγάλη μεροληψία στις αποφάσεις του κάθε δέντρου ξεχωριστά, αλλά τελικά οι προβλέψεις της μεθόδου παρουσιάζουν μικρότερη διακύμανση σε σχέση με τις αντίστοιχες προβλέψεις ενός μόνο δέντρου. Η συγκεκριμένη μέθοδος θεωρείται καλύτερο μοντέλο πρόβλεψης από τα μεμονωμένα δέντρα αποφάσεων, εκτός από την περίπτωση υψηλής συσχέτισης των χαρακτηριστικών εισόδου. Η μέθοδος Random Forest περιέχει τις υπερπαραμέτρους των δέντρων αποφάσεων, ενώ ακόμα δέχεται ως παράμετρο και τον αριθμό των δέντρων αποφάσεων που θα χρησιμοποιήσει.

### 6.2.4 K Nearest Neighbours (kNN)

Δοθέντος ενός ακεραίου k και ενός σημείου εισόδου, ο αλγόριθμος συγκρίνει το σημείο εισόδου με κάθε άλλο σημείο του δείγματος και υπολογίζει την κάθε απόσταση. Συνήθως υπολογίζεται η Ευκλείδεια απόσταση. Ο εκτιμητής του σημείου εισόδου παίρνει τη μέση τιμή από τις target values των k κοντινότερων γειτόνων, αν ο αλγόριθμος είναι τύπου regression. Αν ο αλγόρι-

θμος έχει σκοπό την ταξινόμηση των τιμών πρόβλεψης σε κατηγορίες, τότε επιλέγεται η κατηγορία που στοχεύει η πλειοψηφία των k γειτόνων (classification algorithm). Επιλέγεται συνήθως περιττός αριθμός για την υπερπαράμετρο k, ώστε να μην υπάρχει πρόβλημα σε περίπτωση ισοβαθμίας προβλέψεων. Όσο υψηλότερη είναι η τιμή της υπερπαραμέτρου k, τόσο μειώνεται ο θόρυβος. Το συγκεκριμένο δίκτυο μηχανικής μάθησης δέχεται μη παραμετροποιήσιμη είσοδο και προσαρμόζεται εύκολα στην εφαρμογή νέων εισόδων. Τα βασικά μειονεκτήματά του είναι η ευαισθησία του σε τοπικές ανωμαλίες και biases. Επιπλέον για κάθε νέα είσοδο ο αλγόριθμος καλεί το δίκτυο να υπολογίσει τις αποστάσεις του νέου σημείου εισόδου, με κάθε άλλο σημείο.

### 6.2.5 Multi-Layer Perceptron (Artificial Neural Network)

Η χρήση αλγορίθμων της κατηγορίας artificial neural networks είναι ιδιαίτερα δημοφιλής σήμερα. Χρησιμοποιούνται κυρίως για την πρόβλεψη τιμής στόχου μη γραμμικών προβλημάτων. Οι συσχετίσεις που συμπεραίνονται από τα συγκεκριμένα δίκτυα θεωρούνται ιδιαίτερα περίπλοκες. Θα περιγράψουμε σε αυτό το σημείο τη δομή ενός multilayer perceptron (fully feedforward ANN). Το νευρωνικό δίκτυο δέχεται ως είσοδο ένα διάνυσμα N διαστάσεων $x_1, ..., x_n$. Από το επίπεδο εισόδου κάθε $x_i$ συνδέεται με κάθε μονάδα του πρώτου κρυφού επιπέδου. Κάθε είσοδος $x_i$ πολλαπλασιάζεται με το αντίστοιχο βάρος $w_{i,j}$ πριν την είσοδό του στην αντίστοιχη μονάδα j του πρώτου κρυφού επιπέδου. Σε κάθε μονάδα του πρώτου κρυφού επιπέδου παράγεται η τιμή:

$$z_j = (x_1 * w_{1,j} + ... + x_n * w_{n,j})$$

.

Η παραγόμενη τιμή εισέρχεται ως είσοδος σε μία συνάρτηση ενεργοποίησης. Παραθέτουμε κάποιες από τις δημοφιλέστερες συναρτήσεις ενεργοποίησης:

$h(z_j) = max(0, z_j)$ (rectifier)

$h(z\_j) = log(1 + e^{z_j})$ (softplus)

$h(z_j) = tanh(z_j)$ (hyperbolic tangent)

$h(z_j) = (1 + e^{-z_j})^{-1}$ (logistic function)

Σε κάθε κρυφό επίπεδο L το output της συνάρτησης ενεργοποίησης εισέρχεται ως είσοδος σε κάθε μονάδα του επιπέδου (L+1), πολλαπλασιαζόμενο πάντα από το αντίστοιχο βάρος. Το αποτέλεσμα βασίζεται στις εξόδους των μονάδων του τελευταίου επιπέδου, χωρίς να περνάει απαραίτητα από την ίδια συνάρτηση ενεργοποίησης. Κριτήριο επιτυχίας θεωρείται το αποτέλεσμα της συνάρτησης απωλειών κατά την έξοδο.

Για την εκπαίδευση η ίδια διαδικασία επαναλαμβάνεται για κάθε είσοδο, αφού ανανεωθούν οι τιμές των βαρών από την έξοδο προς την είσοδο (Back-Propagation Algorithm). Τα επιμέρους βάρη ανανεώνονται με βάση συγκεκριμένη συνάρτηση. Η μερική παράγωγος της συνάρτησης απωλειών, ως προς το κάθε βάρος ξεχωριστά, πολλαπλασιάζεται με την υπερπαράμετρο α και προστίθεται στην παλαιά τιμή του βάρους, ώστε να προκύψει το νέο ανανεωμένο βάρος. Παραθέτουμε τη σχέση αναλυτικά:

$$w_{i,jnew} = w_{i,j} + \alpha * \frac{\partial J(w_{i,j})}{\partial wi, j}$$

όπου J η συνάρτηση απωλειών.

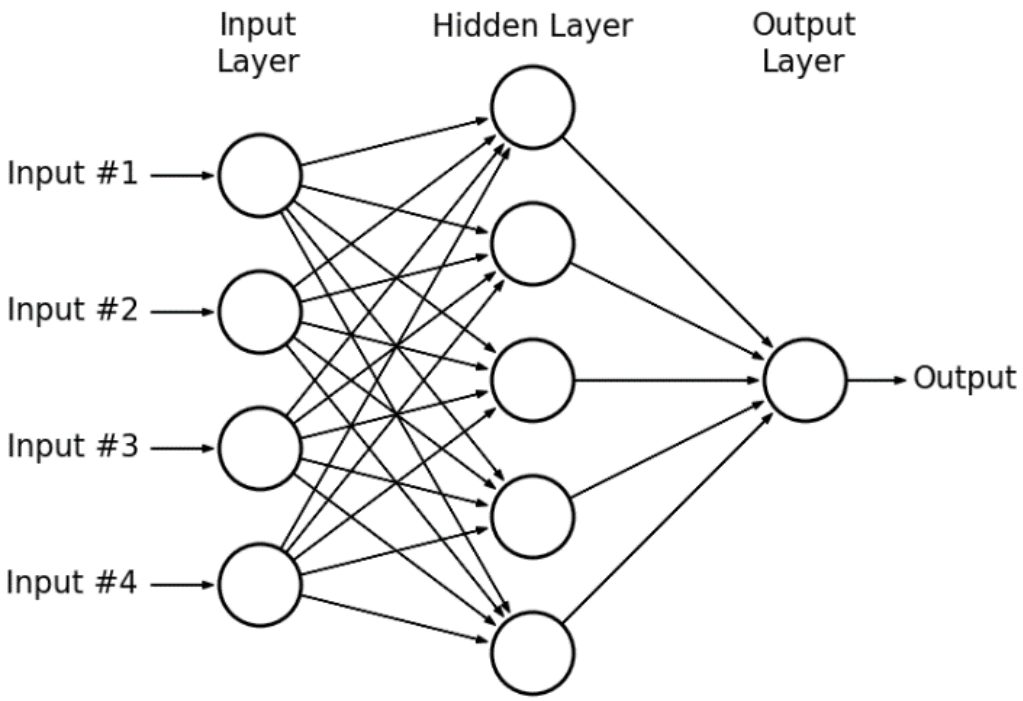
Η διαδικασία σταματάει όταν υπάρξει σύγκλιση, δηλαδή όταν το αποτέλεσμα της συνάρτησης απωλειών, κατά την έξοδο είναι μηδέν, ή όταν η έξοδος της συνάρτησης απωλειών αρχίζει σταθερά να αυξάνεται. Οι υπερπαράμετροι του δικτύου είναι το πλήθος των μονάδων κάθε επιπέδου, το πλήθος των επιπέδων, η συνάρτηση ενεργοποίησης, η συνάρτηση απωλειών, ο αριθμός των δειγμάτων εισόδου σε κάθε επανάληψη (batch size), το learning rate και οι όροι ομαλοποίησης (regularazation terms).

Το δίκτυο Multi-Layer Perceptron θεωρείται ένα είδος ANN. Περιέχει τουλάχιστον τρία επίπεδα κόμβων, το επίπεδο εισόδου, το επίπεδο εξόδου και τουλάχιστον ένα ενδιάμεσο επίπεδο. Χρησιμοποιεί μη γραμμική συνάρτηση ενεργοποίησης σε κάθε κόμβο. Για την εκπαίδευση του δείγματος χρησιμοποιείται η τεχνική backpropagation, την οποία περιγράψαμε στις προηγούμενες παραγράφους.

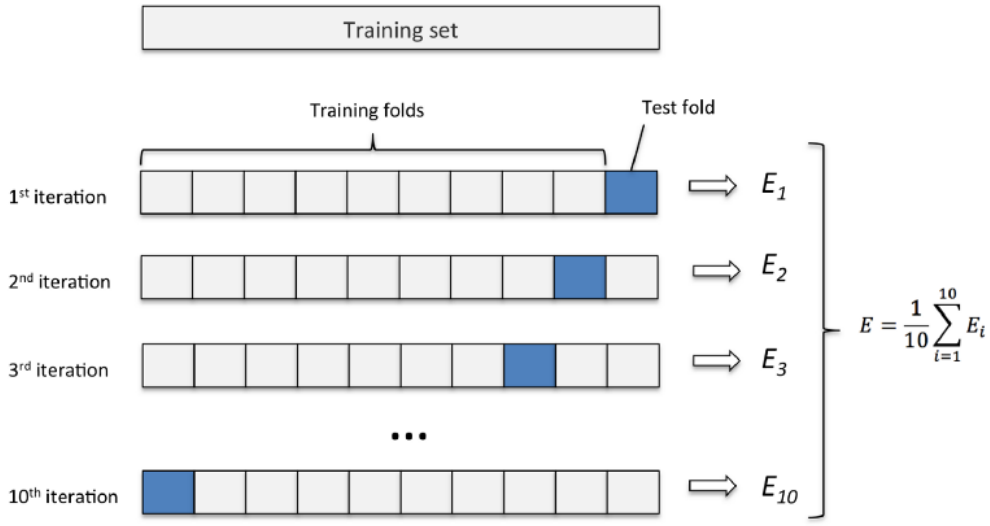
### 6.2.6 Το μοντέλο cross validation

Το μοντέλο cross validation αποτελεί μία στατιστική μέθοδο, η οποία χρησιμοποιείται για την αξιολόγηση των μοντέλων μηχανικής μάθησης. Εφαρμόζεται στην έρευνα, κυρίως, όταν οι μετρήσεις είναι περιορισμένες, με αποτέλεσμα να δυσκολευόμαστε να εξάγουμε ακριβή αποτελέσματα. Στην έρευνά μας σε κάθε μοντέλο μηχανικής μάθησης χρησιμοποιήσαμε τη στατιστική μέθοδο cross validation.

Η μέθοδος του cross validation βασίζεται σε μια πολύ απλή και αποδοτική ιδέα. Έστω ότι έχουμε t αντικείμενα. Χωρίζουμε τα t αντικείμενα σε k ομάδες, όπου κάθε ομάδα αποτελείται από n στοιχεία. Ο διαχωρισμός των στοιχειών σε ομάδες γίνεται με τυχαίο τρόπο. Αυτή η μέθοδος του cross validation είναι γνωστή ως k-fold cross-validation. Το κάθε στοιχείο μπορεί να ανήκει μόνο σε μια ομάδα. Γενικά προσπαθούμε να διατηρούμε την σχέση p = n * k, όσο περισσότερο είναι δυνατόν έτσι ώστε να υπάρχει ομοιομορφία στην μέθοδο.




Εικόνα 6.1: Perceptron με ένα κρυφό επίπεδο.




Εικόνα 6.2: Η μέθοδος cross validation.

Αφού κατασκευαστούν οι n ομάδες, η διαδικασία που πραγματοποιείται είναι η εξής:
1) Επιλέγουμε μία ομάδα, έστω i, για να πραγματοποιήσουμε τον έλεγχο.
2) Συλλέγουμε τις n-1 ομάδες για να πραγματοποιήσουμε την εκπαίδευση του μοντέλου.
3) Εκπαιδεύουμε το μοντέλο μηχανικής μάθησης.
4) Ελέγχουμε την εκπαίδευση χρησιμοποιώντας την ομάδα i.
5) Συλλέγουμε το ποσοστό επιτυχίας του συγκεκριμένου γύρου.
6) Αναιρούμε τις αλλαγές στο μοντέλο, λόγω της εκπαίδευσης.

Η διαδικασία επαναλαμβάνεται για κάθε ομάδα i στο [0,n-1]. Αφού πραγματοποιηθούν n επαναλήψεις, λαμβάνουμε τον μέσο όρο των ποσοστών επιτυχίας των n επαναλήψεων. Το συγκεκριμένο αποτέλεσμα αποτελεί το ποσοστό επιτυχίας της μεθόδου. Στην Εικόνα 6.2 βλέπουμε σχηματικά τη μέθοδο.

Στην κλασική μέθοδο Train/Test Split δε γνωρίζουμε κατά πόσο η κατανομή των δειγμάτων στο test σύνολο ανταποκρίνεται στην πραγματικότητα, με αποτέλεσμα, το ποσοστό επιτυχίας του μοντέλου μηχανικής μάθησης πολλές φορές να μην είναι πραγματικό. Με την τεχνική k folder cross validation το πρόβλημα αυτό εξαλείφεται ή και μετριάζεται. Επιπλέον, με τη συγκεκριμένη τεχνική περιορίζεται το πρόβλημα του overfitting. Overfitting σημαίνει ότι το μοντέλο έχει εξειδικευτεί πάρα πολύ πάνω σε ένα σύνολο δειγμάτων εισόδου και ότι αν το ελέγξουμε με ένα άλλο διαφορετικό σύνολο δειγμάτων εισόδου το σφάλμα θα είναι μεγάλο.

## 6.3 Μοντελοποίηση προγραμμάτων

Για να αποφασίσουμε σε ποιο είδος πυρήνα θα εκτελεστούν τα προγράμματα, χρησιμοποιήσαμε και τα πέντε παραπάνω μοντέλα μηχανικής μάθησης με στόχο την πρόβλεψη δυαδικών τιμών στόχων. Ως εισόδους, για την εκπαίδευση των μοντέλων, χρησιμοποιήσαμε ομαλοποιημένο τον αριθμό branches, load, store, integer, floating-point, simd και other εντολών, ως ποσοστό επί του συνόλου των εντολών, του κάθε προγράμματος. Επιπλέον, ως παραμέτρους εισάγαμε και τα L1 και L2 cache misses των δύο ειδών πυρήνων. Συνολικά, δηλαδή, είχαμε έντεκα παραμέτρους εισόδου. Θεωρούμε ότι οι συγκεκριμένες παράμετροι μπορούν να οδηγήσουν ένα μοντέλο μηχανικής μάθησης σε υψηλά ποσοστά επιτυχημένων προβλέψεων, καθώς καθορίζουν σε πολύ μεγάλο βαθμό οι τιμές τους την απόδοση και την ενεργειακή κατανάλωση των πυρήνων. Υπενθυμίζουμε ότι οι μετρήσεις των δειγμάτων εισόδου πραγματοποιήθηκαν στις μέγιστες συχνότητες λειτουργίας κάθε πυρήνα. Ο big πυρήνας "έτρεχε" στα 2,1 GHz και ο little στα 1,5 GHz.

Πρέπει να υπογραμμίσουμε ότι λόγω του μεγάλου αριθμού μετρήσεων με test και simlarge input sets, τα οποία είναι μικρότερα σε σχέση με τα reference και native input sets, στα Spec2006 και στα Parsec προγράμματα αντίστοιχα, θα υπάρχει σημαντικό ποσοστό θορύβου. Ήταν ευκολότερο να πραγματοποιηθούν μετρήσεις στις caches, καθώς και να λάβουμε το instruction mix των προγραμμάτων για μικρά input sets. Επίσης λόγω των διαφόρων σφαλμάτων στις μετρήσεις των προγραμμάτων, Spec2006 και Parsec, σίγουρα θα υπάρχει μεροληψία στα δεδομένα εισόδου των μεθόδων μηχανικής μάθησης. Ακόμα αξίζει να αναφέρουμε πως για την καλύτερη δυνατή αξιολόγηση των μοντέλων μηχανικής μάθησης χρησιμοποιήσαμε σε όλες τις περιπτώσεις τη μέθοδο cross validation.

Όσον αφορά τα Spec2006 προγράμματα συνολικά χρησιμοποιήσαμε 36 δείγματα, τα οποία περιείχαν μετρήσεις προγραμμάτων τόσο με χρήση test input sets, όσο και με χρήση reference input sets. Ο στόχος μας ήταν να επιλεγεί το κατάλληλο είδος πυρήνα για κάθε κατηγορία εκτέλεσης. Δηλαδή ο στόχος είναι να επιλεγεί ο πυρήνας που δίνει το χαμηλότερο συντελεστή EDP είτε $ED^2P$, ανάλογα με την περίπτωση. Επιπλέον προσπαθήσαμε να προβλέψουμε με τις ίδιες μεθόδους μηχανικής μάθηση ποιες Spec2006 εφαρμογές υποστηρίζουν την παράλληλη εκτέλεσή τους με άλλα προγράμματα.

Ακριβώς την ίδια μεθοδολογία ακολουθήσαμε και για τα Parsec προγράμματα. Ο στόχος μας ήταν να επιλεγεί πάλι το είδος πυρήνα που δίνει το χαμηλότερο συντελεστή EDP είτε $ED^2P$, για εκτελέσεις του ενός thread. Σημαντική επέκταση, θα ήταν πέρα από το βέλτιστο είδος πυρήνα, να μπορούμε να επιλέξουμε και το βέλτιστο αριθμό thread με τον οποίο θα εκτελείται το κάθε πρόγραμμα αυτής της κατηγορίας, αλλά και τη βέλτιστη συχνότητα λειτουργίας

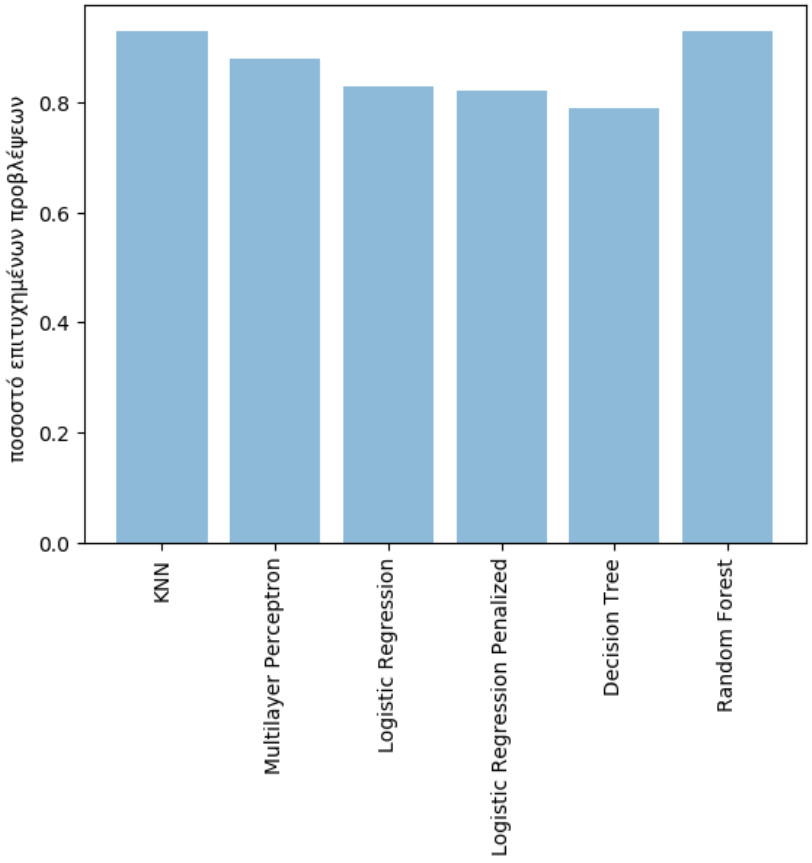
των πυρήνων. Επιπρόσθετα και στο συγκεκριμένο είδος προγραμμάτων προσπαθήσαμε να προβλέψουμε με τις ίδιες μεθόδους μηχανικής μάθηση ποιες Parsec μονονηματικές εφαρμογές υποστηρίζουν την παράλληλη εκτέλεσή τους με άλλα προγράμματα. Συνολικά χρησιμοποιήσαμε 20 δείγματα στη συγκεκριμένη κατηγορία. Οι είσοδοι προέρχονταν από μετρήσεις σε Parsec προγράμματα είτε με native input sets είτε με simlarge input sets.

Τέλος, αναμείξαμε τα δεδομένα εισόδων-τιμών στόχων των προγραμμάτων Spec2006 και Parsec, ώστε να αποκτήσουμε ένα μεγαλύτερο σύνολο δειγμάτων εκπαίδευσης των μοντέλων μηχανικής μάθησης, χρησιμοποιώντας συνολικά 56 δείγματα στη συγκεκριμένη κατηγορία.

### 6.3.1 Πρόβλεψη με βάση την μετρική EDP

Ο συντελεστής EDP (energy delay product) είναι το γινόμενο χρόνου εκτέλεσης μίας εφαρμογής επί την ενεργειακή της κατανάλωση σε αυτό το χρονικό διάστημα. Δίνει ίδιο βάρος τόσο στην απόδοση του μετρούμενου επεξεργαστή όσο και στην ενεργειακή του κατανάλωση. Το είδος πυρήνα που εμφανίζει το χαμηλότερο συντελεστή EDP για ένα πρόγραμμα, είναι το κατάλληλο για την εκτέλεση του προγράμματος.

Προσπαθήσαμε να προβλέψουμε σε ποιο είδος πυρήνα θα τρέξει μία εφαρμογή, σε μεγάλο ή σε μικρό, με βάση το συντελεστή EDP. Εισάγαμε στα μοντέλα μηχανικής μάθησης τις εισόδους που έχουμε περιγράψει. Εισάγαμε το κανονικοποιημένο instruction mix των προγραμμάτων και τα L1, L2 cache misses και των δύο ειδών πυρήνων. Ως τιμές στόχους θέσαμε με μονάδα την επιλογή μεγάλου πυρήνα και με μηδέν την επιλογή μικρού πυρήνα. Για κάθε πρόγραμμα επιδιώκουμε να επιλέγεται ο πυρήνας με τον μικρότερο συντελεστή EDP. Τα αποτελέσματα που πήραμε είχαν σχετικά υψηλά ποσοστά επιτυχίας. Πιστεύουμε ότι θα είχαμε ακόμα υψηλότερα ποσοστά επιτυχίας αν διαθέταμε περισσότερες μετρήσεις από benchmarks με μεγάλα input sets. Τα ποσοστά επιτυχημένων προβλέψεων των μοντέλων μηχανικής μάθησης με βάση το συντελεστή EDP για τα Spec2006 προγράμματα βρίσκονται στην εικόνα 6.3.

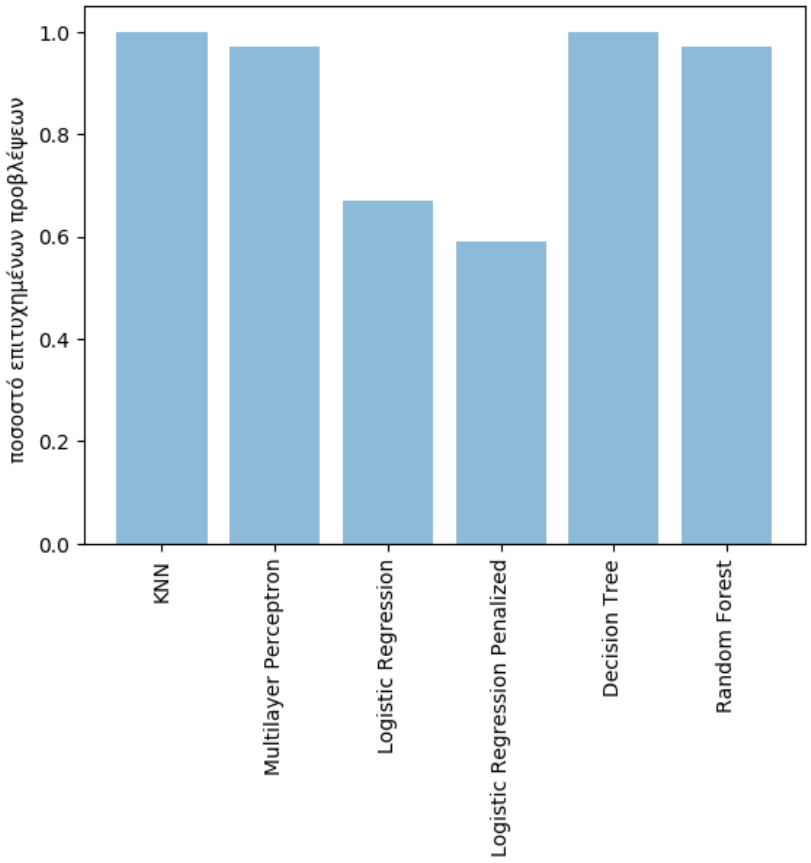



Εικόνα 6.3: Ποσοστά επιτυχημένων προβλέψεων των μοντέλων μηχανικής μάθησης με βάση το συντελεστή EDP για τα Spec2006 προγράμματα.

Με βάση τα αποτελέσματά της Εικόνας 6.3 υψηλότερα ποσοστά επιτυχίας εμφάνισαν τα μοντέλα kNN και Random Forest, με ποσοστό 93% το καθένα. Φαίνεται το μοντέλο kNN να μπορεί να επιλύσει το συγκεκριμένο πρόβλημα. Αναμέναμε το μοντέλο Random Forest να εμφανίζει καλύτερη συμπεριφορά από το μοντέλο Decision Tree, όπως και συνέβη. Συγκεκριμένα το μοντέλο Decision Tree εμφάνισε ποσοστό επιτυχίας 79%. Ποσοστό 88% επιτυχημένων προβλέψεων εμφάνισε το Multilayer Perceptron. Το μοντέλο Logistic Regression εμφάνισε ποσοστό επιτυχίας 83% χωρίς τη χρήση της νόρμας L2, ενώ με τη χρήση της συγκεκριμένης νόρμας εμφάνισε ποσοστό επιτυχίας 82%. Χρησιμοποιούμε τη νόρμα L2 για να αποφευχθεί υπερεκπαίδευση του μοντέλου πάνω σε εξειδικευμένα δείγματα. Σημειώνουμε πως η μορφή των δειγμάτων, καθώς και ο αριθμός τους δεν μας επιτρέπει να καταλήξουμε σε ασφαλή συμπεράσματα παρά τη χρήση της μεθόδου cross validation.

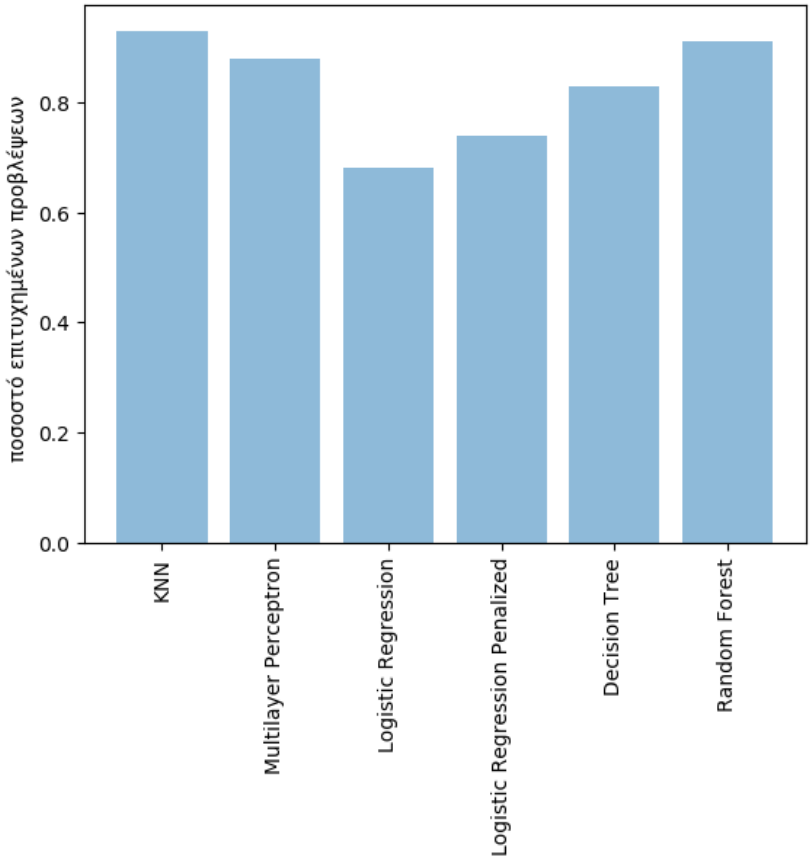
Την ίδια διαδικασία που περιγράψαμε παραπάνω ακολουθήσαμε και για τα

Parsec προγράμματα, όταν τα τρεχουμε με ένα thread. Εκπαιδεύσαμε με τον ίδιο τρόπο τα πέντε μοντέλα μηχανικής μάθησης και τα χρησιμοποιήσαμε για να προβλέψουμε σε ποιο είδος πυρήνα πρέπει να εκτελεστεί το κάθε Parsec μονονηματικό πρόγραμμα με βάση το συντελεστή EDP. Και σε αυτήν την περίπτωση χρησιμοποιήσαμε το γνωστό μοντέλο cross validation. Τα ποσοστά επιτυχημένων προβλέψεων των μοντέλων μηχανικής μάθησης με βάση το συντελεστή EDP για τα Parsec προγράμματα βρίσκονται στην Εικόνα 6.4.




Εικόνα 6.4: Ποσοστά επιτυχημένων προβλέψεων των μοντέλων μηχανικής μάθησης με βάση το συντελεστή EDP για τα Parsec προγράμματα.

Με βάση τα αποτελέσματά μας υψηλότερα ποσοστά επιτυχίας εμφάνισαν τα μοντέλα kNN, Multilayer Perceptron, Decision Tree και Random Forest, με ποσοστά 100%, 97% ,100% και 97% αντίστοιχα. Παρατηρούμε ότι ιδιαίτερα χαμηλά ήταν τα ποσοστά επιτυχημένων προβλέψεων του μοντέλου μηχανικής μάθησης Logistic Regression using Stochastic Gradient Descent τόσο με τη χρήση νόρμας L2, όσο και χωρίς. Συγκεκριμένα το μοντέλο Logistic Regres-




Εικόνα 6.5: Ποσοστά επιτυχημένων προβλέψεων των μοντέλων μηχανικής μάθησης με βάση το συντελεστή EDP για τα Spec2006 και Parsec προγράμματα.

sion using Stochastic Gradient Descent εμφάνισε ποσοστό επιτυχημένων προβλέψεων 67%, ενώ με τη χρήση της νόρμας L2 παρουσίασε ποσοστό επιτυχημένων προβλέψεων 59%.

Αναμείξαμε τα δεδομένα εισόδων και τιμών στόχων των προγραμμάτων Spec2006 και Parsec, ώστε να αποκτήσουμε ένα μεγαλύτερο σύνολο δειγμάτων εκπαίδευσης των μοντέλων μηχανικής μάθησης. Εκπαιδεύσαμε με τη γνωστή μεθοδολογία τα μοντέλα μηχανικής μάθησης. Τα ποσοστά επιτυχημένων προβλέψεων των μοντέλων μηχανικής μάθησης με βάση το συντελεστή EDP για τα Spec2006 και τα Parsec προγράμματα βρίσκονται στην Εικόνα 6.5.

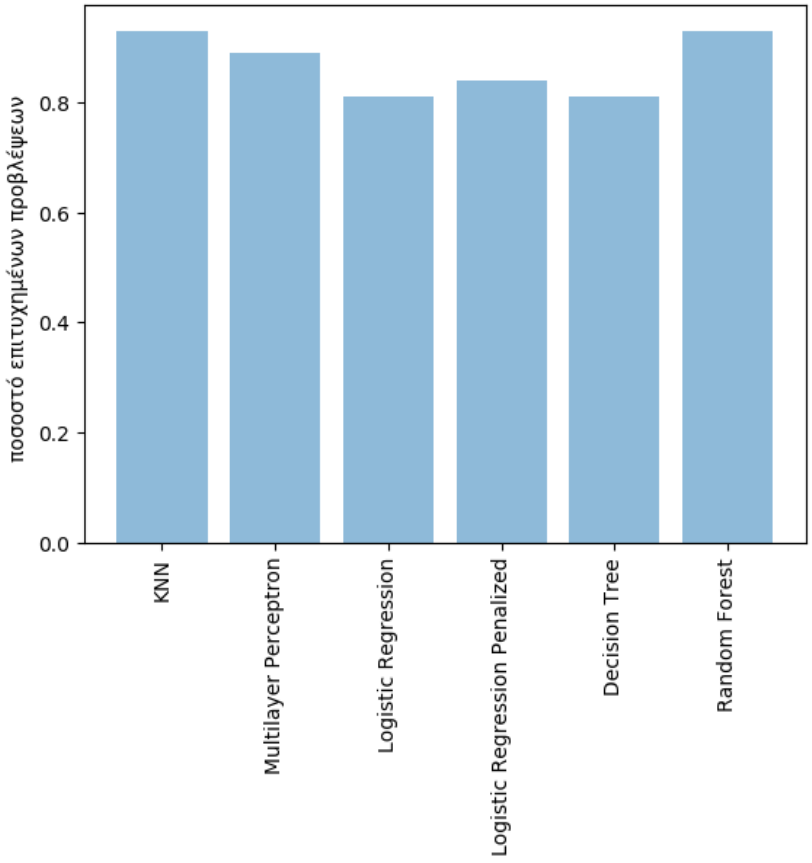
Με βάση τα αποτελέσματά μας υψηλότερα ποσοστά επιτυχίας εμφάνισαν τα μοντέλα kNN, Multilayer Perceptron και Random Forest, με ποσοστό 93%, 88% και 91% αντίστοιχα. Χαμηλά ήταν τα ποσοστά επιτυχημένων προβλέψεων του μοντέλου Logistic Regression using Stochastic Gradient Descent τόσο

με τη χρήση νόρμας L2, όσο και χωρίς. Το συγκεκριμένο μοντέλο εμφάνισε ποσοστό επιτυχημένων προβλέψεων 68% χωρίς τη χρήση της νόρμας L2, ενώ με τη χρήση της εμφάνισε ποσοστό 74%. Το μοντέλο Decision Tree είχε επιτυχημένες προβλέψεις σε ποσοστό 83%.

### 6.3.2 Πρόβλεψη με βάση την μετρική $ED^2P$

Ο συντελεστής $ED^2P$ (energy delay delay product) είναι το γινόμενο του τετραγώνου του χρόνου εκτέλεσης μίας εφαρμογής επί την ενεργειακή της κατανάλωση σε αυτό το χρονικό διάστημα εκτέλεσης. Ο συγκεκριμένος συντελεστής δίνει έμφαση στην επίδοση του συστήματος και δευτερευόντως ασχολείται με την ενεργειακή κατανάλωσή του. Ανταποκρίνεται σε πολλές περιπτώσεις καλύτερα στις απαιτήσεις του χρήστη. Και εδώ το είδος πυρήνα που εμφανίζει το χαμηλότερο συντελεστή $ED^2P$ για ένα πρόγραμμα, είναι το κατάλληλο για την εκτέλεσης του προγράμματος.

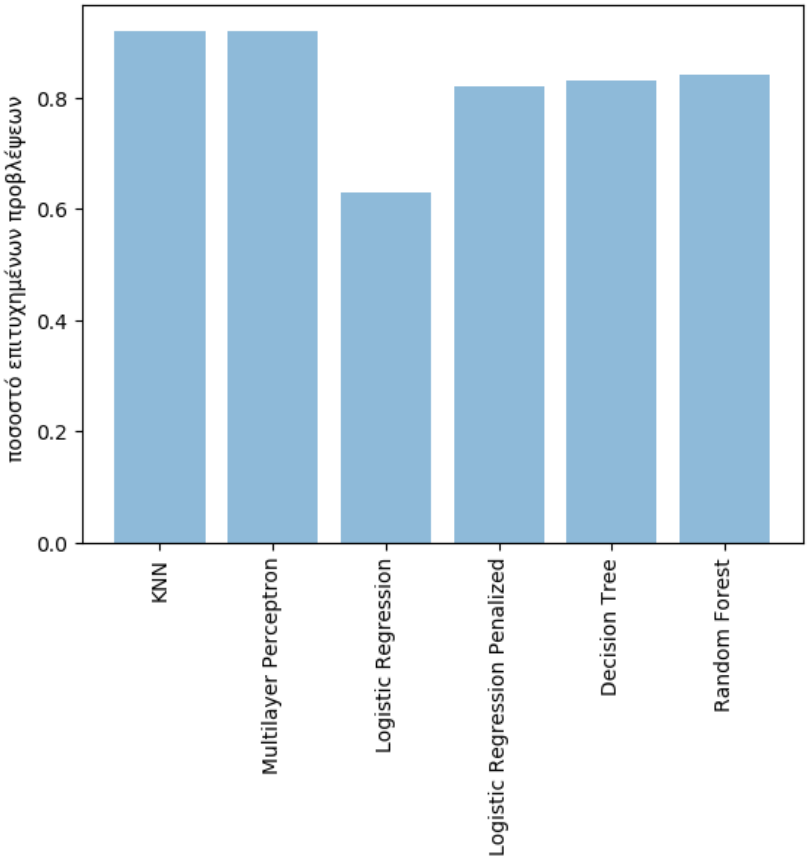
Και σε αυτό το υποκεφάλαιο προσπαθήσαμε να προβλέψουμε σε ποιο είδος πυρήνα θα "τρέξει" μία εφαρμογή, σε μεγάλο ή σε μικρό, με βάση τις γνωστές εισόδους, το κανονικοποιημένο instruction mix των προγραμμάτων και τα L1, L2 cache misses τους. Ως τιμές στόχους θέσαμε με μονάδα την επιλογή μεγάλου πυρήνα και με μηδέν την επιλογή μικρού πυρήνα. Κάθε φορά επιλέγεται ο πυρήνας με τον μικρότερο συντελεστή $ED^2P$. Τα αποτελέσματα που πήραμε είχαν σχετικά υψηλά ποσοστά επιτυχίας. Τα ποσοστά επιτυχημένων προβλέψεων των μοντέλων μηχανικής μάθησης με βάση το συντελεστή $ED^2P$ για τα Spec2006 προγράμματα βρίσκονται στην Εικόνα 6.6.




Εικόνα 6.6: Ποσοστά επιτυχημένων προβλέψεων των μοντέλων μηχανικής μάθησης με βάση το συντελεστή $ED^2P$ για τα Spec2006 προγράμματα.

Τα υψηλότερα ποσοστά επιτυχημένων προβλέψεων είδους πυρήνα με βάση το συντελεστή εμφάνισαν τα μοντέλα kNN, Random Forest και Multilayer Perceptron, με ποσοστό 93%, 93% και 89% αντίστοιχα. Το μοντέλο Decsion Tree εμφάνισε ποσοστό επιτυχημένων προβλέψεων 81% και σε αυτήν την περίπτωση παρουσιάζει χαμηλότερο ποσοστό επιτυχημένων προβλέψεων από εκείνο της μεθόδου Random Forest. Ποσοστό επιτυχίας 81% εμφάνισε το Logistic Regression χωρίς τη χρήση της νόρμας L2, ενώ με τη χρήση της συγκεκριμένης νόρμας εμφάνισε ποσοστό επιτυχίας 84%.

Την ίδια διαδικασία ακολουθήσαμε και για τα Parsec προγράμματα. Τα ποσοστά επιτυχημένων προβλέψεων των μοντέλων μηχανικής μάθησης με βάση το συντελεστή $ED^2P$ για τα Parsec προγράμματα βρίσκονται στην Εικόνα 6.7.

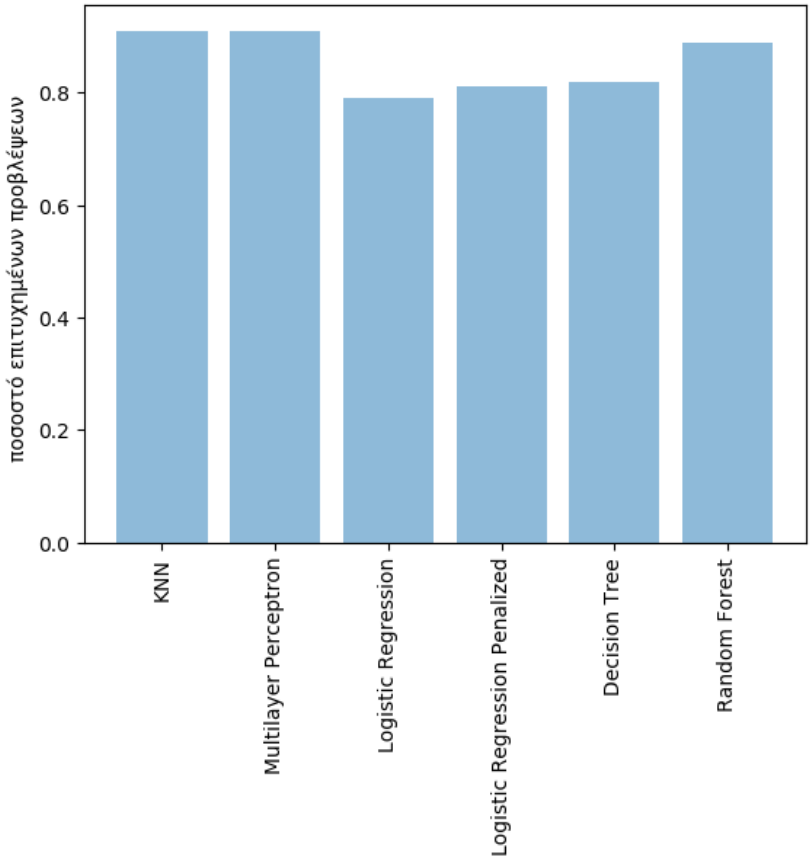



Εικόνα 6.7: Ποσοστά επιτυχημένων προβλέψεων των μοντέλων μηχανικής μάθησης με βάση το συντελεστή $ED^2P$ για τα Parsec προγράμματα.

Παρατηρούμε ότι στα συγκεκριμένα αποτελέσματα, τα ψηλότερα ποσοστά επιτυχίας εμφάνισαν τα μοντέλα kNN και Multilayer Perceptron, με ποσοστό 92% το καθένα. Τα μοντέλα Decsion Tree και Random Forest εμφάνισαν ποσοστό επιτυχημένων προβλέψεων 83% κια 84% αντίστοιχα. Τέλος, το μοντέλο Logistic Regression χωρίς τη χρήση της νόρμας L2 εμφάνισε ποσοστό επιτυχημένων προβλέψεων 63%, ενώ με τη χρήση της συγκεκριμένης νόρμας εμφάνισε ποσοστό επιτυχίας 82%.

Αναμείξαμε τα δεδομένα εισόδων-τιμών στόχων των προγραμμάτων Spec2006 και Parsec, ώστε να αποκτήσουμε ένα μεγαλύτερο σύνολο δειγμάτων εκπαίδευσης των μοντέλων μηχανικής μάθησης. Εκπαιδεύσαμε με τη γνωστή μεθοδολογία τα μοντέλα μηχανικής μάθησης. Τα ποσοστά επιτυχημένων προβλέψεων των μοντέλων μηχανικής μάθησης με βάση το συντελεστή $ED^2P$ για τα Spec2006 και τα Parsec προγράμματα βρίσκονται στην εικόνα 6.8.

Με βάση τα αποτελέσματά μας υψηλότερα ποσοστά επιτυχίας εμφάνισαν τα




Εικόνα 6.8: Ποσοστά επιτυχημένων προβλέψεων των μοντέλων μηχανικής μάθησης με βάση το συντελεστή $ED^2P$ για τα Spec2006 και Parsec προγράμματα.

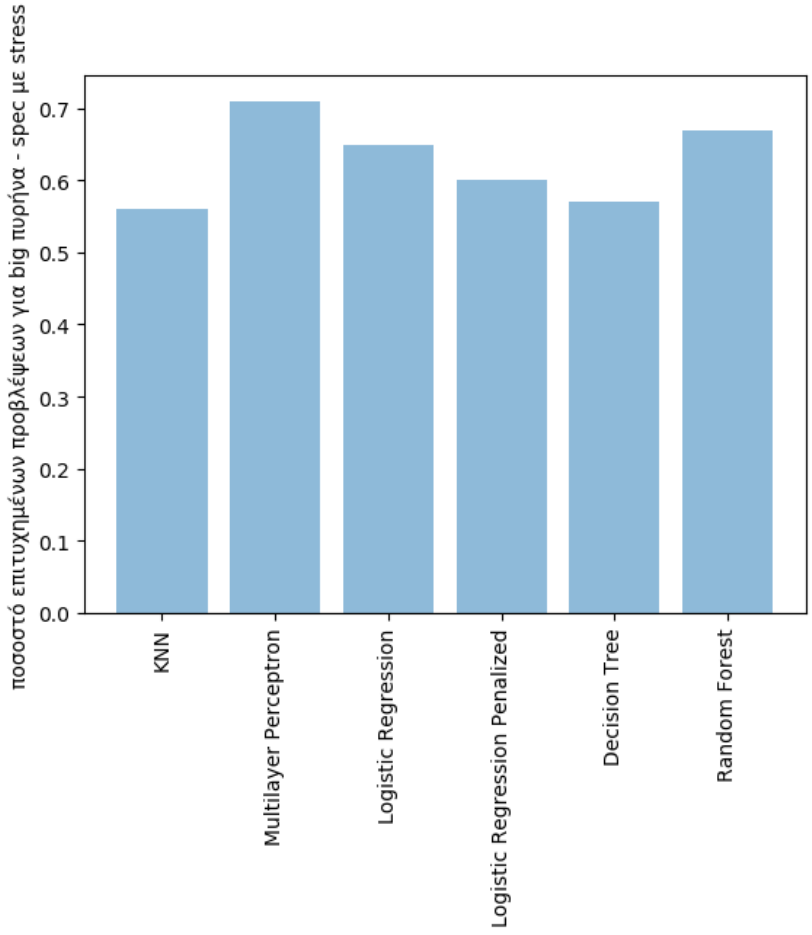
μοντέλα kNN, Multilayer Perceptron και Random Forest, με ποσοστά 91%, 91% και 89% αντίστοιχα. Σχετικά υψηλά ήταν τα ποσοστά επιτυχημένων προβλέψεων του μοντέλου Logistic Regression using Stochastic Gradient Descent, τουλάχιστον σε σχέση με τα αντίστοιχα αποτελέσματα όταν χρησιμοποιούμε ως τιμή στόχο το συντελεστή EDP. Το συγκεκριμένο μοντέλο εμφάνισε ποσοστό επιτυχημένων προβλέψεων 79% χωρίς τη χρήση της νόρμας L2, ενώ με τη χρήση της εμφάνισε ποσοστό 81%. Το μοντέλο Decision Tree είχε επιτυχημένες προβλέψεις σε ποσοστό 82%.

### 6.3.3 Πρόβλεψη με βάση την συνεκτέλεση stress και τον ανταγωνισμό στην L2 cache

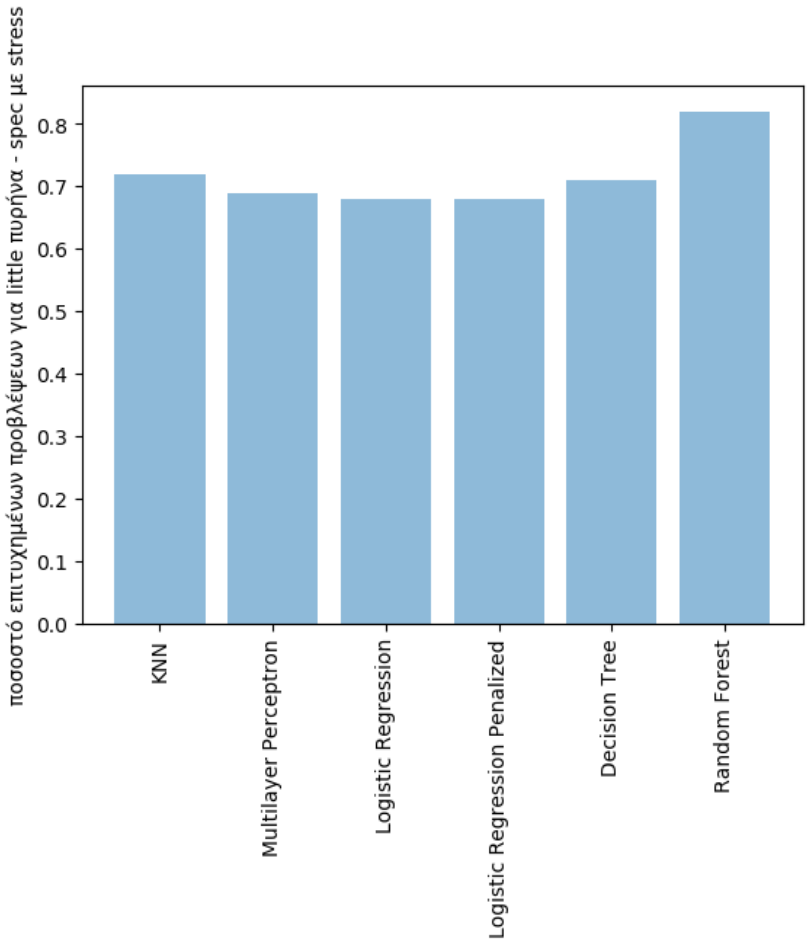
Ελέγξαμε τη συμπεριφορά των προγραμμάτων σε πραγματικές συνθήκες. Χωρίσαμε τα προγράμματα με βάση τη συμπεριφορά τους όταν τρέχουν με άλλο πρόγραμμα (stress) παράλληλα στο ίδιο σύμπλεγμα πυρήνων. Εκείνα που παρουσίαζαν καθυστέρηση χρόνου πάνω από 25% σε σχέση με το αν πραγματοποιούσαν εκτέλεση μόνα τους στο ίδιο σύμπλεγμα πυρήνων τα θεωρήσαμε ακατάλληλα για παράλληλη εκτέλεση. Τα υπόλοιπα τα τοποθετήσαμε στην κατηγορία κατάλληλα για τέτοιου είδους παράλληλη εκτέλεση. Χρησιμοποιήσαμε τα δεδομένα από τις μετρήσεις μας με stress προγράμματα. Τη συγκεκριμένη κατηγοριοποίηση την πραγματοποιήσαμε τόσο για τα Spec2006 όσο και για τα Parsec προγράμματα του ενός thread.

Αφού πραγματοποιήσαμε την κατηγοριοποίηση των προγραμμάτων, πάντα με βάση τις μονονηματικές εκτελέσεις τους, τα χρησιμοποιήσαμε σαν τιμές στόχους στα μοντέλα μηχανικής μάθησης. Το σύνολο των κατηγοριών των δειγμάτων εισόδου είναι ίδιο με αυτό που χρησιμοποιήσαμε για την πρόβλεψη των συντελεστών EDP, $ED^2P$. Χρησιμοποιήσαμε, πάλι, το instruction mix των προγραμμάτων και τα L1 και L2 cache misses τους. Τα μοντέλα μηχανικής μάθησης σε αυτήν την περίπτωση προβλέπουν αν πρέπει να τρέξει μία εφαρμογή του ενός thread παράλληλα με άλλο πρόγραμμα του ενός thread σε ένα σύμπλεγμα πυρήνων με βάση το input της. Και σε αυτού του είδους τη μοντελοποίηση χρησιμοποιήσαμε τη μέθοδο cross validation, λόγω των περιορισμών που έπρεπε να αντιμετωπίσουμε. Τα ποσοστά επιτυχημένων προβλέψεων των μοντέλων μηχανικής μάθησης με βάση την καταπόνηση της L2 cache μνήμης βρίσκονται στις εικόνες 6.9, 6.10, 6.11 και 6.12. Επιχειρήσαμε να μοντελοποιήσουμε τα προγραμματα Spec2006 και Parsec με βάση την καταπόνηση της L2 cache μνήμης και στα δύο είδη πυρήνων big και little.

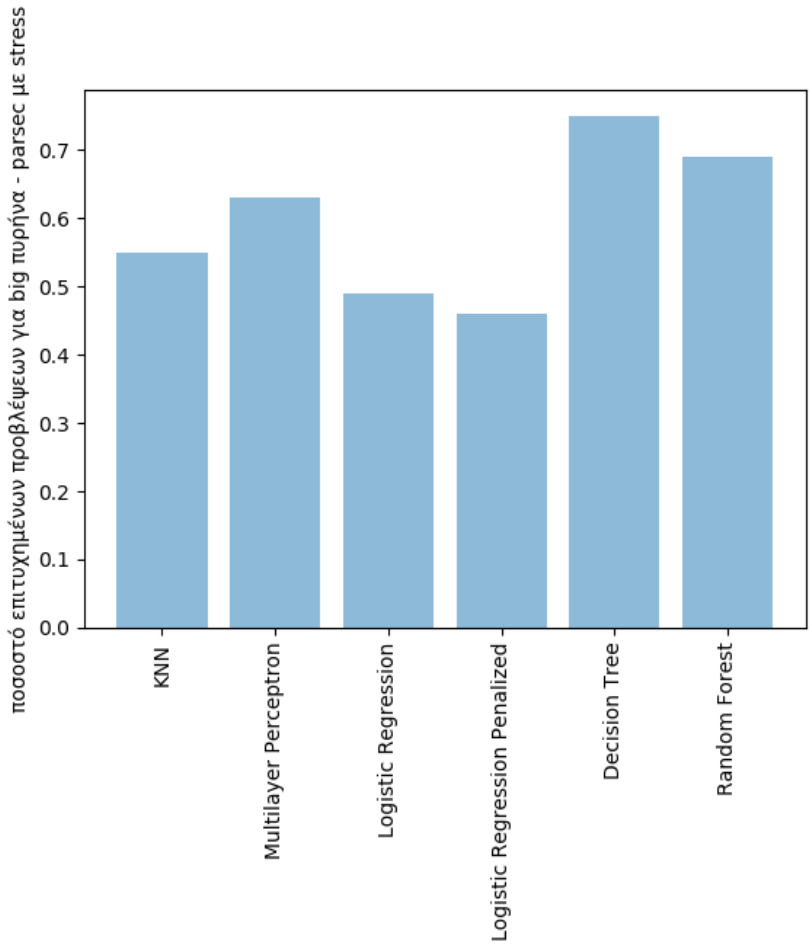
Τα ποσοστά επιτυχημένων προβλέψεων των μοντέλων μηχανικής μάθησης με βάση την καταπόνηση της L2 cache μνήμης ήταν ιδιαίτερα χαμηλά και στα δύο είδη πυρήνων και για τα δύο είδη συνόλων, Spec2006 και Parsec. Θεωρούμε



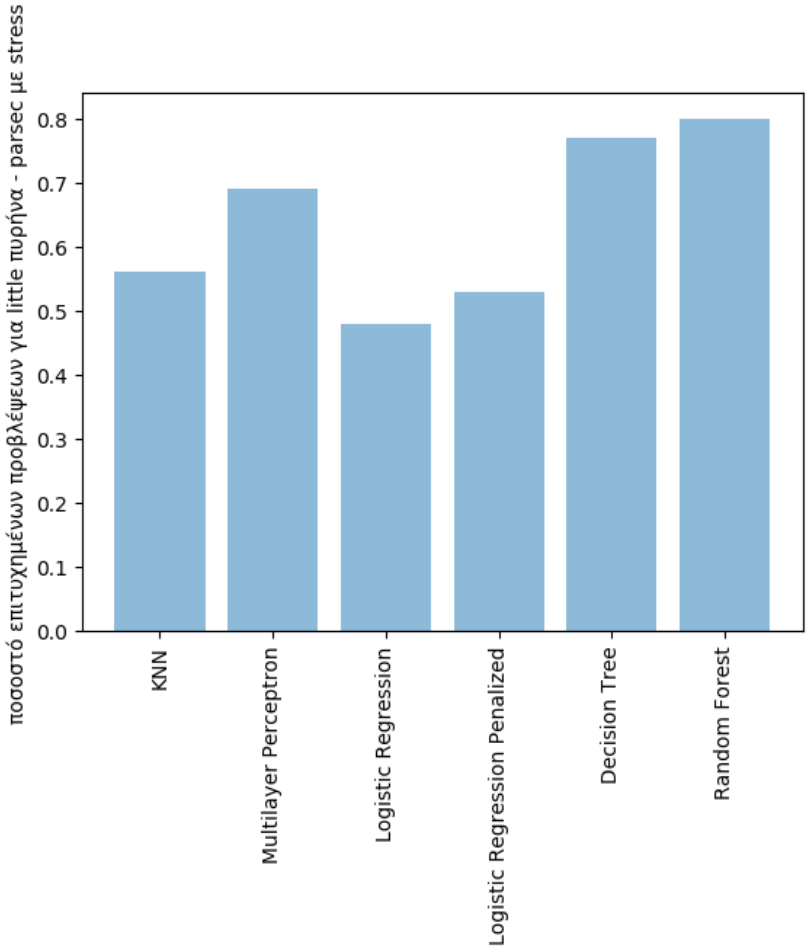

Εικόνα 6.9: Ποσοστά επιτυχημένων προβλέψεων των μοντέλων μηχανικής μάθησης με βάση την καταπόνηση της L2 cache στους big πυρήνες για τα Spec2006 προγράμματα.




Εικόνα 6.10: Ποσοστά επιτυχημένων προβλέψεων των μοντέλων μηχανικής μάθησης με βάση την καταπόνηση της L2 cache στους little πυρήνες για τα Spec2006 προγράμματα.




Εικόνα 6.11: Ποσοστά επιτυχημένων προβλέψεων των μοντέλων μηχανικής μάθησης με βάση την καταπόνηση της L2 cache στους big πυρήνες για τα Parsec προγράμματα.




Εικόνα 6.12: Ποσοστά επιτυχημένων προβλέψεων των μοντέλων μηχανικής μάθησης με βάση την καταπόνηση της L2 cache στους little πυρήνες για τα Parsec προγράμματα.

ότι τα είδη των εισόδων που χρησιμοποιούμε στα μοντέλα μηχανικής μάθησης δεν μπορούν να καθορίσουν απόλυτα την παράλληλη συμπεριφορά των προγραμμάτων. Περισσότερο θα μας ενδιέφεραν τα είδη των cache misses ανά κατηγορία. Δηλαδή θα έπρεπε να μετρήσουμε τα ποσοστά κάθε κατηγορίας cache misses, όταν το πρόγραμμα τρέχει μόνο του και όταν εκτελείται παράλληλα με το stress πρόγραμμα στο ίδιο σύμπλεγμα πυρήνων. Τις συγκεκριμένες μετρήσεις θα έπρεπε να τις θέσουμε ως εισόδους στα μοντέλα μηχανικής μάθησης. Όμως θα ήταν πολύ δύσκολο να πραγματοποιηθούν τέτοιου είδους μετρήσεις σε πραγματικό χρόνο. Παρόλα αυτά θεωρούμε ότι τα cache misses και τα ποσοστά load και store εντολών κάθε προγράμματος επηρεάζουν τη συμπεριφορά τους.

Αξίζει να αναφέρουμε ότι αναμείξαμε τα δεδομένα εισόδων-τιμών στόχων της συγκεκριμένης κατηγορίας των προγραμμάτων Spec2006 και Parsec. Εκπαιδεύσαμε με τη γνωστή μεθοδολογία τα μοντέλα μηχανικής μάθησης. Τα ποσοστά επιτυχημένων προβλέψεων των μοντέλων μηχανικής μάθησης με βάση την καταπόνηση της L2 cache μνήμης ήταν εξαιρετικά χαμηλά. Γι' αυτόν το λόγο θεωρήσαμε ότι δεν ήταν απαραίτητο να παραθέσουμε τα αποτελέσματα. Όπως αναφέραμε και στην προηγούμενη παράγραφο, τα είδη των εισόδων που χρησιμοποιούμε στα μοντέλα μηχανικής μάθησης θεωρούμε ότι δεν μπορούν να καθορίσουν απόλυτα την παράλληλη συμπεριφορά των προγραμμάτων.

Τέλος, πρέπει να αναφέρουμε ότι δεν πραγματοποιήσαμε μοντελοποίηση με βάση την καταπόνηση της κύριας μνήμης. Τα πειραματικά αποτελέσματα δεν μας το επέτρεψαν. Πρωτίστως τα προγράμματα Spec2006 και δευτερευόντως τα προγράμματα Parsec είχαν μικρές αποκλίσεις στους χρόνους εκτέλεσής τους όταν έτρεχαν μόνα τους σε σχέση με την παράλληλη εκτέλεσή τους με stress προγράμματα σε διαφορετικά συμπλέγματα πυρήνων.

## 6.4 Συμπεράσματα μοντελοποίησης προγραμμάτων

Συμπερασματικά καταφέραμε να μοντελοποιήσουμε σε ποιον πυρήνα θα εκτελεστούν τα προγράμματα Spec2006 και τα μονονηματικά Parsec προγράμματα με βάση τους συντελεστές EDP και $ED^2P$. Χρησιμοποιήσαμε τα μοντέλα μηχανικής μάθησης για προβλέψουμε σε ποιον πυρήνα θα πρέπει να εκτελεστεί το κάθε πρόγραμμα. Τα ποσοστά επιτυχημένων προβλέψεων ήταν σχετικά υψηλά όταν θέταμε ως τιμές στόχους τους συντελεστές EDP και $ED^2P$. Σημειώνουμε ότι τα ποσοστά επιτυχημένων προβλέψεων των μοντέλων μηχανικής μάθησης με βάση το συντελεστή EDP για Parsec προγράμματα ήταν σε τέσσερις περιπτώσεις άνω του 95%.

Πρέπει να υπογραμμίσουμε ότι λόγω του μεγάλου αριθμού μετρήσεων με

test και simlarge input sets, στα Spec2006 και στα Parsec προγράμματα αντίστοιχα, θα υπάρχει σημαντικό ποσοστό θορύβου. Ήταν ευκολότερο να πραγματοποιηθούν μετρήσεις στις caches, καθώς και να λάβουμε το intrsuction mix των προγραμμάτων για μικρά input sets. Επίσης λόγω των διαφόρων προβλημάτων στις μετρήσεις των προγραμμάτων, Spec2006 και Parsec, θα υπάρχει μεροληψία στα δεδομένα εισόδου των μεθόδων μηχανικής μάθησης. Επιπλέον τα δείγματα εισόδου στα μοντέλα μηχανικής μάθησης ήταν σχετικά μικρά. Γι' αυτούς τους λόγους θεωρούμε ότι τα ποσοστά επιτυχημένων προβλέψεων των μοντέλων μηχανικής μάθησης δεν ήταν απόλυτα αντιπροσωπευτικά της πραγματικότητας.

Τέλος, μοντελοποιήσαμε τη συμπεριφορά των προγραμμάτων με βάση την συνεκτέλεση με το stress benchmark και τον ανταγωνισμό στην L2 cache μνήμη. Χωρίσαμε τα προγράμματα με βάση τη συμπεριφορά τους όταν τρέχουν με άλλο πρόγραμμα παράλληλα στο ίδιο σύμπλεγμα πυρήνων. Εκείνα που παρουσίαζαν καθυστέρηση χρόνου πάνω από 25% σε σχέση με το αν πραγματοποιούσαν εκτέλεση μόνα τους στο ίδιο σύμπλεγμα πυρήνων τα θεωρήσαμε ακατάλληλα για παράλληλη εκτέλεση. Χρησιμοποιήσαμε τα μοντέλα μηχανικής μάθησης για να προβλέψουμε σε ποιον πυρήνα θα πρέπει να εκτελεστεί το κάθε πρόγραμμα. Τα ποσοστά επιτυχημένων προβλέψεων ήταν πολύ χαμηλά. Θεωρούμε ότι τα είδη των εισόδων που χρησιμοποιούμε στα μοντέλα μηχανικής μάθησης δεν μπορούν να καθορίσουν απόλυτα την παράλληλη συμπεριφορά των προγραμμάτων.

# Κεφάλαιο 7

# Επίλογος

## 7.1 Σύνοψη και Συμπεράσματα

Στο πρώτο κεφάλαιο της παρούσας διπλωματικής αναφερθήκαμε στους στόχους μας. Ας τους επαναφέρουμε για να αναλύσουμε τι πετύχαμε τελικά.

- Να περιγράψουμε τι είναι ετερογένεια, καθώς και την τεχνολογία ARM big.LITTLE.

Στο κεφάλαιο 2 περιγράψαμε αναλυτικά τι είναι ετερογένεια, για ποιους λόγους είναι αναγκαία η εφαρμογή της στην αρχιτεκτονική υπολογιστών, την τεχνολογία ARM big.LITTLE, καθώς και τις μεθόδους χρονοδρομολόγησης εργασιών σε τέτοιου είδους συστήματα.

- Να πραγματοποιήσουμε μετρήσεις πάνω στα benchmarks προγράμματα. Να μετρήσουμε τους χρόνους εκτέλεσης, τα ποσοστά των cache misses το instruction mix, την συμπεριφορά όταν συνεκτελούνται με άλλες εφαρμογές, και την ενεργειακή κατανάλωση αυτών των προγραμμάτων.

Στα κεφάλαια 4 και 5 έχουμε παραθέσει τα αποτελέσματα των μετρήσεών μας πάνω στις χρονικές μετρήσεις, το instruction mix, τα cache misses και τον αντίκτυπο συνεκτέλεσης, καθώς και την ενεργειακή κατανάλωση των Spec2006 και Parcec προγραμμάτων, αντίστοιχα.

- Να αναλύσουμε τη συμπεριφορά των benchmarks και να κατανοήσουμε ποιοι παράγοντες επηρεάζουν την επίδοση των προγραμμάτων στα δύο είδη πυρήνων.

Αναλύσαμε τη συμπεριφορά των Spec2006 και Parsec προγραμμάτων στα κεφάλαια 4 και 5 αντίστοιχα. Επιχειρήσαμε να αναλύσουμε ποιοι παράγοντες καθορίζουν την επίδοση των προγραμμάτων στα δύο είδη πυρήνων με βάση τις δικές μας μετρήσεις, αλλά χρησιμοποιώντας και πληροφορίες από άλλες έρευνες.

- Να κατανοήσουμε και να περιγράψουμε μοντέλα μηχανικής μάθησης.

Στο κεφάλαιο 6 περιγράψαμε τη μέθοδο Logistic Regression using Stochastic Gradient Descent, τη μέθοδο Decision Tree, τη μέθοδο Random Forest, τη μέθοδο kNN (k Nearest Neighbours) και το Multilayer Perceptron.

- Να μοντελοποιήσουμε τους επεξεργαστές ARM big.LITTLE, προβλέποντας με τη χρήση μοντέλων μηχανικής μάθησης σε ποιο είδος πυρήνα θα μπορεί να τρέξει μία μονομηματική εφαρμογή.

Αφού αναλύσαμε εις βάθος ποιοι παράγοντες καθορίζουν την επίδοση μίας εφαρμογής στα δύο είδη πυρήνων, επιχειρήσαμε να μοντελοποιήσουμε τους παραπάνω τύπους επεξεργαστών. Χρησιμοποιήσαμε σαν εισόδους στα μοντέλα μηχανικής μάθησης τα δεδομένα των μετρήσεών μας πάνω στα benchmarks. Στο κεφάλαιο 6 καταφέραμε να μοντελοποιήσουμε με υψηλή επιτυχία, μέσω των μοντέλων μηχανικής μάθησης, ποιο είδος πυρήνα θα επιλέγεται για την εκτέλεση μίας μονονηματικής εφαρμογής από τις σουίτες προγραμμάτων Spec2006 και Parsec. Τη μοντελοποίηση των προγραμμάτων πραγματοποιήσαμε με χρήση των συντελεστών EDP και $ED^2P$. Επιπλέον, προσπαθήσαμε να μοντελοποιήσουμε την συμπεριφορά των προγραμμάτων όταν συνεκτελούνται ταυτόχρονα με άλλο πρόγραμμα στο ίδιο σύμπλεγμα πυρήνων,όμως δεν μπορέσαμε να προβλέψουμε με υψηλή επιτυχία τη συμπεριφορά αυτή.

Συμπερασματικά θεωρούμε ότι τα μοντέλα μηχανικής μάθησης μπορούν να διαδραματίσουν σημαντικό ρόλο στη μοντελοποίηση επεξεργαστών και στη δημιουργία προγραμμάτων χρονοδρομολόγησης εργασιών του λειτουργικού συστήματος.

## 7.2 Μελλοντικές επεκτάσεις

Βασική επέκταση της παρούσας διπλωματικής θα ήταν η μοντελοποίηση πολυνηματικών εφαρμογών, η οποία θα προβλέπει τον κατάλληλο συνδυασμό πυρήνα-αριθμό threads-συχνότητας πυρήνων, ώστε να βελτιστοποιείται η τιμή του συντελεστή EDP ή του $ED^2P$. Ακόμα, μια τέτοια μοντελοποίηση θα μπορούσε λαμβάνει υπόψιν την επίδραση της συνεκτέλεσης άλλων προγραμμάτων.

Τέλος, θα είχε ιδιαίτερο ενδιαφέρον να υλοποιηθεί κάποια εφαρμογή είτε επιπέδου χρήστη είτε επιπέδου πυρήνα, η οποία θα υλοποιεί ενεργειακά αποδοτική χρονοδρομολόγηση εργασιών σε πραγματικό χρόνο. Η έρευνα [17] μας παρέχει ένα ρεαλιστικό σχέδιο υλοποίησης της μοντελοποίησης ετερογενών επεξεργαστών ARM που πραγματοποιήσαμε στην παρούσα διπλωματική.

# Βιβλιογραφία


[1] big.little technology: The future of mobilemaking very high performance available in a mobile envelope without sacrificing energy efficiency.

[2] M. Bhadauria, V. M. Weaver, and S. A. McKee. Understanding parsec performance on contemporary cmps. In *2009 IEEE International Symposium on Workload Characterization (IISWC)*, pages 98–107, Oct 2009.

[3] C. Bienia, S. Kumar, J. P. Singh, and K. Li. The parsec benchmark suite: Characterization and architectural implications. In *2008 International Conference on Parallel Architectures and Compilation Techniques (PACT)*, pages 72–81, Oct 2008.

[4] Jason Brownlee. A gentle introduction to k-fold cross-validation, May 2018.

[5] Peter Clarke. Benchmarking arm's big-little architecture. *EETimes*.

[6] Andrei Frumusanu and Ryan Smith. Arm a53/a57/t760 investigated - samsung galaxy note 4 exynos review, 2015.

[7] Andrei Frumusanu and Ryan Smith. Arm a53/a57/t760 investigated - samsung galaxy note 4 exynos review, 2015.

[8] Cosmin Gorgovan, Amanieu D'antras, and Mikel Luján. Mambo: A low-overhead dynamic binary modification tool for arm. *ACM Transactions on Architecture and Code Optimization*, 13:1–26, 04 2016.

[9] Peter Greenhalgh. Big.little processing with arm cortex™-a15 & cortex-a7 arm paper.

[10] John L. Henning. Spec cpu2006 benchmark descriptions. *SIGARCH Comput. Archit. News*, 34(4):1–17, September 2006.

[11] Taehoon Kim Hyun-Duk Cho, Kisuk Chung. Benefits of the big.little architecture.

[12] Anuja Nagpal. L1 and l2 regularization methods, October 2017.

[13] D Nemirovsky, T Arkose, Nikola Markovic, Mario Nemirovsky, O Unsal, Adrian Cristal, and M Valero. A general guide to applying machine learning to computer architecture. *Supercomputing Frontiers and Innovations*, 5:95–115, 01 2018.

[14] Sebastian Norena. Python model tuning methods using cross validation and grid search.

[15] Cslab Ntua. Προχωρημένα Θέματα Αρχιτεκτονική Υπολογιστών ΗΜΜΥ ΕΜΠ διαφάνειες 9,10.

[16] Cslab Ntua. Παράλληλος προγραμματισμός: Σχεδίαση και υλοποίηση παράλληλων προγραμμάτων, 2018.

[17] Irena Orsolic, Dario Pevec, Mirko Suznjevic, and Lea Skorin-Kapov. A machine learning approach to classifying youtube qoe based on encrypted network traffic. *Multimedia Tools and Applications*, 05 2017.

[18] Tribuvan K. Prakash and Lu Peng. Performance characterization of spec cpu 2006 benchmarks on intel core 2 duo processor.

[19] Karl Rosaen. k-fold cross validation.

[20] Hossein Sayadi, Nisarg Patel, Avesta Sasan, and Houman Homayoun. Machine learning-based approaches for energy-efficiency prediction and scheduling in composite cores architectures. pages 129–136, 11 2017.

[21] Law Amar Shan. Heterogeneous processing: a strategy for augmenting moore's law. Technical report, 2006.

[22] Brandon Skerritt. What is a decision tree in machine learning?

[23] Matthew Travers. Cpu power consumption experiments and results analysis of intel i7-4820k.

[24] Stavros Tzilis, Pedro Trancoso, and Ioannis Sourdis. Energy-efficient runtime management of heterogeneous multicores using online projection. *ACM Transactions on Architecture and Code Optimization*, 15:1–26, 01 2019.

[25] E. Vasilakis, I. Sourdis, V. Papaefstathiou, A. Psathakis, and M. G. H. Katevenis. Modeling energy-performance tradeoffs in arm big.little architectures. In *2017 27th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, pages 1–8, Sep. 2017.

[26] Evangelos Vasilakis. An instruction level energy characterization of arm processors.

[27] Wikipedia. Arm big.little.

[28] Wikipedia. Cpu power dissipation.

[29] Wikipedia. Id3 algorithm.

[30] Wikipedia. Moore's law.

[31] Wikipedia. Νόμος του Μουρ.

[32] Chris Yeh. Binary vs. multi-class logistic regression.

[33] Mohamed Zahran. Heterogeneous computing: Here to stay. *ACM Queue*, 14:31–42, 2016.

[34] Yuhao Zhu and Vijay Janapa Reddi. High-performance and energy-efficient mobile web browsing on big/little systems. In *Proceedings of the 2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*, HPCA '13, pages 13–24, Washington, DC, USA, 2013. IEEE Computer Society.

[35] Φανούριος Αραπίδης. Μοντελοποίηση Εφαρμογών σε Αρχιτεκτονικές numa με Τεχνικές Μηχανικής Μάθησης. Master's thesis, ΗΜΜΥ ΕΜΠ,, 2018.

[36] Αθανάσιος Κάτσιος. ποστήριξη ετερογενών αρχιτεκτονικών σε εικονικά περιβάλλοντα. Master's thesis, ΗΜΜΥ ΕΜΠ,, 2017.