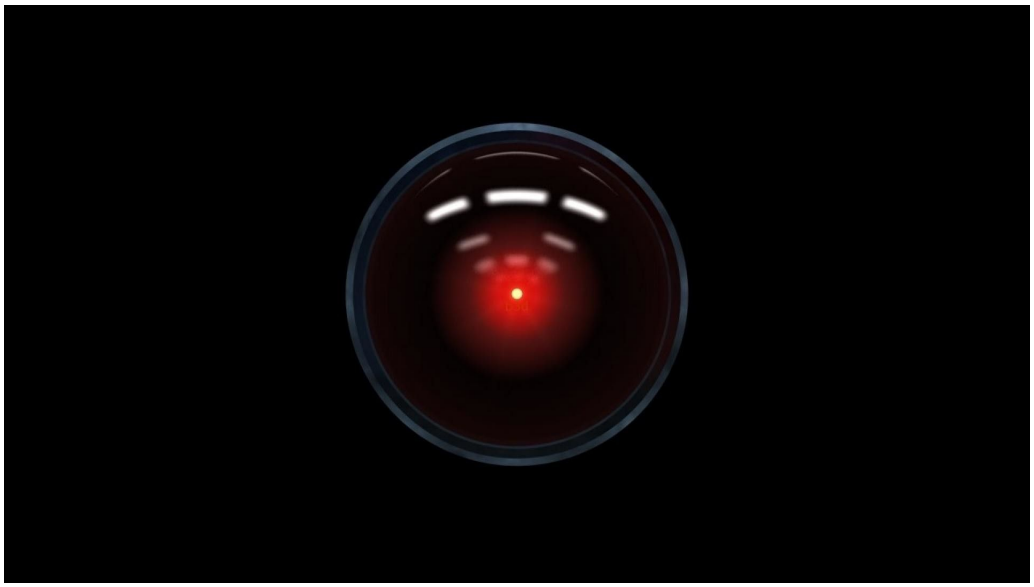
Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων

# Πρόβλεψη και Προβολή Επίδοσης Παράλληλων Εφαρμογών σε Συστήματα Μεγάλης Κλίμακας

Διπλωματικη Εργασια

του

**ΠΕΤΡΟΠΟΥΛΟΥ ΕΥΓΕΝΙΟΥ**



**Επιβλέπων:** Γκούμας Γεώργιος
Αναπληρωτής Καθηγητής

Αθήνα, Ιούνιος 2021

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

# Πρόβλεψη και Προβολή Επίδοσης Παράλληλων Εφαρμογών σε Συστήματα Μεγάλης Κλίμακας

Διπλωματικη Εργασια

του

**ΠΕΤΡΟΠΟΥΛΟΥ ΕΥΓΕΝΙΟΥ**

**Επιβλέπων:** Γκούμας Γεώργιος
Αναπληρωτής Καθηγητής

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 23η Ιουνίου 2021.

| *(Υπογραφή)* | *(Υπογραφή)* | *(Υπογραφή)* |
|---|---|---|
| ..... | ..... | ..... |
| Γκούμας Γεώργιος | Κοζύρης Νεκτάριος | Πνευματικάτος Διονύσιος |
| Αναπληρωτής Καθηγητής | Καθηγητής | Καθηγητής |

Αθήνα, Ιούνιος 2021

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών



## ΔΗΛΩΣΗ ΜΗ ΛΟΓΟΚΛΟΠΗΣ ΚΑΙ ΑΝΑΛΗΨΗΣ ΠΡΟΣΩΠΙΚΗΣ ΕΥΘΥΝΗΣ

Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ενυπογράφως ότι είμαι αποκλειστικός συγγραφέας της παρούσας Πτυχιακής Εργασίας, για την ολοκλήρωση της οποίας κάθε βοήθεια είναι πλήρως αναγνωρισμένη και αναφέρεται λεπτομερώς στην εργασία αυτή. Έχω αναφέρει πλήρως και με σαφείς αναφορές, όλες τις πηγές χρήσης δεδομένων, απόψεων, θέσεων και προτάσεων, ιδεών και λεκτικών αναφορών, είτε κατά κυριολεξία είτε βάσει επιστημονικής παράφρασης. Αναλαμβάνω την προσωπική και ατομική ευθύνη ότι σε περίπτωση αποτυχίας στην υλοποίηση των ανωτέρω δηλωθέντων στοιχείων, είμαι υπόλογος έναντι λογοκλοπής, γεγονός που σημαίνει αποτυχία στην Πτυχιακή μου Εργασία και κατά συνέπεια αποτυχία απόκτησης του Τίτλου Σπουδών, πέραν των λοιπών συνεπειών του νόμου περί πνευματικών δικαιωμάτων. Δηλώνω, συνεπώς, ότι αυτή η Πτυχιακή Εργασία προετοιμάστηκε και ολοκληρώθηκε από εμένα προσωπικά και αποκλειστικά και ότι, αναλαμβάνω πλήρως όλες τις συνέπειες του νόμου στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής άλλης πνευματικής ιδιοκτησίας.

*(Υπογραφή)*

.....
Ευγένιος Πετρόπουλος

23 Ιουνίου 2021

# Περίληψη

Ο σχεδιασμός υπερυπολογιστών επόμενης γενιάς αποτελεί μια πρόκληση για τους αρχιτέκτονες αυτών των συστημάτων. Η μοντελοποίηση και η πρόβλεψη της επίδοσης των παράλληλων εφαρμογών συνιστά μια ουσιαστική βοήθεια σε αυτήν τους την προσπάθεια. Ένα από τα σημαντικότερα προβλήματα που καλούνται να επιλύσουν τα μοντέλα πρόβλεψης είναι η επίδοση των παράλληλων εφαρμογών σε συστήματα μεγαλύτερης κλίμακας από τα υπάρχοντα, δηλαδή η προβολή αυτής (performance extrapolation). Αυτό αποτελεί ένα επιτακτικό ερώτημα, αφού βρισκόμαστε σε μια περίοδο μετάβασης όπου η ισχύς των συστημάτων θα περάσει από επιδόσεις της τάξης των PetaFLOPS σε επιδόσεις της τάξης των ExaFLOPS.

Επιχειρώντας να δώσουμε κάποιες απαντήσεις σε αυτά τα ερωτήματα, παρουσιάζουμε, σε αυτή την εργασία, τη μελέτη μας πάνω σε μια μέθοδο εμπειρικής μοντελοποίησης, η οποία έχει αναπτυχθεί στη βιβλιογραφία, με σκοπό να περιγράψει, μέσω απλών συναρτήσεων, τις ανάγκες μιας εφαρμογής σε πόρους ενός συστήματος, όπως η μνήμη, το πλήθος πράξεων κινητής υποδιαστολής, κ.ά. Αυτό μπορεί να βοηθήσει τόσο τους σχεδιαστές των συστημάτων νέας γενιάς, όσο και αυτούς που αναπτύσσουν παράλληλες εφαρμογές, να επικεντρωθούν σε καλύτερο σχεδιασμό πόρων των συστημάτων ή να ανακαλύψουν προβληματικούς κώδικες που δεν θα κλιμακώνουν ικανοποιητικά αντιστοίχως. Πέρα από την παρουσίαση αυτής της μεθοδολογίας, την εξετάζουμε σε τέσσερις μικρές εφαρμογές, οι οποίες υλοποιούν υπολογισμούς που απαντώνται συχνά σε παράλληλες εφαρμογές, ενώ προτείνουμε και μια εναλλακτική μέθοδο κατάταξης των προβλέψεων, συγκρίνοντάς τη με αυτή της βιβλιογραφίας.

Στη συνέχεια επιχειρούμε, με πανομοιότυπο τρόπο, να μοντελοποιήσουμε τον χρόνο εκτέλεσης αυτών των εφαρμογών ως συνάρτηση των αναγκών αυτής. Προτείνουμε έναν τρόπο πρόβλεψης του χρόνου εκτέλεσης χρησιμοποιώντας τα εμπειρικά μοντέλα που παράγουμε για τις ανάγκες αυτών, δημιουργώντας έτσι μοντέλα που παράγονται από εμπειρικές προβλέψεις. Επίσης, διεκπεραιώνουμε αυτή τη διαδικασία σε δύο διαφορετικά συστήματα, παλαιού και πιο σύγχρονου τύπου, αναδεικνύοντας έτσι τη διαφορά στις δυνατότητες μοντελοποίησης μεταξύ νέων και παλαιών αρχιτεκτονικών. Επιχειρούμε ακόμη να αναδείξουμε πιθανά οφέλη των μικρών εφαρμογών από διαφορετικές μελλοντικές σχεδιαστικές προσεγγίσεις χρησιμοποιώντας τεχνικές συν-σχεδίασης.

Τέλος, παρουσιάζουμε την ίδια προσπάθεια μοντελοποίησης αναγκών για μια πιο σύνθετη εφαρμογή, η οποία εκτελεί μια προσομοίωση πρακτόρων (agent-based modeling) σε γράφο. Μια τέτοια μοντελοποίηση αποτελεί πρόκληση καθώς οι αλγόριθμοι που δρουν σε γράφους εξαρτώνται σε μέγιστο βαθμό από τη φύση της εισόδου (γράφου) αλλά κι από τον τρόπο καταμερισμού της εργασίας, και διαφέρουν από την απλή παραμετροποίηση της εισόδου που απαντάται στις άλλες εφαρμογές που μελετούνται. Για αυτό τον λόγο, προτείνουμε δύο διαφορετικούς τρόπους αναπαράστασης των χαρακτηριστικών του γράφου ως είσοδο στα μοντέλα,

συγκρίνοντάς τες μεταξύ τους.

## Λέξεις Κλειδιά

Συστήματα μεγάλης κλίμακας, παράλληλες εφαρμογές, πρόβλεψη επίδοσης, προβολή επίδοσης, εμπειρική μοντελοποίηση, συν-σχεδίαση

# Abstract

Designing next generation's supercomputers can prove to be a challenging task for those involved. Modelling and predicting the efficiency of parallel applications can turn out to be of significant help in this process. One of the greatest problems faced by predictive models is performance extrapolation. Solving this problem is of great importance as we are currently faced with transitioning from PetaFLOPS to ExaFLOPS systems.

As an attempt to provide answers to the challenges mentioned above we present, in this thesis, a thorough study on an empirical modelling method presented in related research. This method aims to model an application's requirements that cover performance critical application aspects, like memory footprint, number of FLOPS, etc. This can prove to be extremely beneficial for system architects in making designing decisions, as well as for applications' developers in discovering scalability bottlenecks in their codes. After presenting this methodology, we put it to the test in four "mini-apps", which execute calculations common for parallel applications. Moreover we propose a different scoring function and compare the results with the one presented in the related work.

In addition, we provide a methodology to present execution time for these four "mini-apps" as a function of their requirements. We accomplish this by using the empirical models generated for these requirements, creating, in this way, models produced by empirical predictions. Moreover, we test this procedure in two different systems, an old one and a newer one, in order to highlight the differences in modelling capability between up-to-date and older architectures. Furthermore, we attempt to illustrate possible system design tradeoffs for these four applications using co-design techniques.

Finally we also present an attempt to model the same requirements for a more complicated application, which executes an agent-based modelling simulation on a given graph. That sort of modelling can prove to be challenging because the algorithms that operate on graphs depend, to an extensive degree, on the layout of the graph as well as the distribution of work, something that is different compared to the rather straightforward input parameterization on the other applications explored in this thesis. As a result, we propose two different representations of graph characteristics as model input and draw comparisons between the two.

## Keywords

large-scale systems, parallel applications, performance prediction, performance extrapolation, empirical modelling, co-design

# Ευχαριστίες

Η παρούσα διπλωματική εργασία σημαίνει και το πέρας της φοιτητικής μου ζωής στο ΕΜΠ. Θεωρώ πρέπον λοιπόν σε αυτό το σημείο να ευχαριστήσω μερικούς ανθρώπους που με βοήθησαν σε αυτό το ταξίδι. Πρώτον, θα ήθελα να ευχαριστήσω τον αναπληρωτή καθηγητή Γεώργιο Γκούμα που μου έδωσε την ευκαιρία να ασχοληθώ με αυτό το θέμα, επέβλεψε την εργασία και καλλιέργησε το ενδιαφέρον μου για το αντικείμενο μέσα από τα μαθήματά του. Επίσης, θα ήθελα να ευχαριστήσω ιδιαιτέρως την δρ. Νικέλα Παπαδοπούλου για την πολύτιμη καθοδήγηση και βοήθεια καθ' όλη τη διάρκεια της εκπόνησης της διπλωματικής εργασίας.

Σε μια πιο προσωπική νότα, θα ήθελα να ευχαριστήσω και όλους τους φίλους μου, οι οποίοι υπήρξαν συνοδοιπόροι στο φοιτητικό μου ταξίδι και με βοήθησαν (και συνεχίζουν να με βοηθούν) περισσότερο από όσο και οι ίδιοι γνωρίζουν.

Τέλος, θα ήθελα να εκφράσω την ευγνωμοσύνη μου προς την οικογένειά μου. Ιδιαίτερα προς τους γονείς μου Λένα και Πέτρο, την αδερφή μου Αθηνά και τον θείο μου Βασίλη οι οποίοι πάντοτε στηρίζουν τις αποφάσεις μου και συνδράμουν στη ζωή μου με αγάπη.

*Ευγένιος Πετρόπουλος*

# Περιεχόμενα

# Κατάλογος Σχημάτων

# Κατάλογος Πινάκων

# Πρόλογος

Η παρούσα διπλωματική εργασία εκπονήθηκε στο Εργαστήριο Υπολογιστικών Συστημάτων (CSLab) της σχολής Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Εθνικού Μετσόβιου Πολυτεχνείου στην Αθήνα, κατά το ακαδημαϊκό έτος 2020-2021.

Εικόνα Εξωφύλλου: ο υπερυπολογιστής HAL9000 από την ταινία 2001: A Space Odyssey (1968) του Stanley Kubrick

# Μέρος I

# Εισαγωγή

Κεφάλαιο 1

# Εισαγωγή

## 1.1 Αντικείμενο και Κίνητρο της Εργασίας

Τα συστήματα μεγάλης κλίμακας έχουν σημαντικό ρόλο στις επιτυχίες της επιστημονικής κοινότητας και χρησιμοποιούνται σε πληθώρα πεδίων, όπως κβαντική μηχανική, μετεωρολογικές προβλέψεις, εξαγωγή μοριακών μοντέλων, κ.ά. Οι υπερυπολογιστές έκαναν την εμφάνισή τους της δεκαετία του ′60 και η υπολογιστική τους ισχύ αυξάνεται χρόνο με το χρόνο, ενώ από το 2017 υπάρχουν συστήματα με δυνατότητα εκτέλεσης περισσότερων από $10^{17}$ FLOPS (Floating-point operations per second-πράξεις κινητής υποδιαστολής ανά δευτερόλεπτο), ή αλλιώς περισσότερα από 100 PetaFLOPS. Μάλιστα η λίστα Top500 [1], η οποία αναλαμβάνει την κατάταξη των 500 ισχυρότερων υπερυπολογιστών παγκοσμίως, αναφέρει, στην περίοδο συγγραφής αυτής της εργασίας, ότι ο ταχύτερος υπερυπολογιστής στον κόσμο, ο Fugaku της Fujitsu στο Kobe της Ιαπωνίας, έχει πετύχει ένα νέο παγκόσμιο ρεκόρ στα 442 PetaFLOPS.

Είναι λοιπόν εμφανές ότι βρισκόμαστε στην περίοδο μετάβασης από την εποχή των επιδόσεων της τάξης των PetaFLOPS σε αυτή των επιδόσεων της τάξης των ExaFLOPS, κάτι το οποίο θα σημαίνει και την άυξηση του πλήθους των πυρήνων και των κόμβων των υπερυπολογιστών. Προκειμένου να σχεδιαστούν τα νέα συστήματα αυτών των προδιαγραφών είναι επιτακτική η ανάγκη μοντελοποίησης και πρόβλεψης της επίδοσης των παράλληλων εφαρμογών που εκτελούνται σε αυτά. Μάλιστα, ένα από τα σημαντικότερα ερωτήματα τα οποία και καλούνται να απαντήσουν τα διάφορα μοντέλα πρόβλεψης αποτελεί η επίδοση των εφαρμογών σε συστήματα μεγαλύτερης κλίμακας από τα υπάρχοντα, δηλαδή η προβολή της επίδοσης των εφαρμογών (performance extrapolation). Επιπλέον, δεδομένου του πολύ υψηλού κόστους ενός συστήματος με δυνατότητες ExaFLOPS, σε συνδυασμό με το γεγονός ότι η σχεδίαση του αποτελεί διαδικασία αρκετών χρόνων ενώ το ίδιο παραμένει “κορυφαίο” για τρία με πέντε χρόνια, είναι ολοφάνερο ότι τα μηχανήματα αυτά πρέπει να σχεδιάζονται σύμφωνα με τις ανάγκες των εφαρμογών που θα τρέξουν σε αυτά, αλλά και να αποδίδουν τις αναμενόμενες επιδόσεις από τη στιγμή της εγκατάστασής τους.

Λόγω των παραπάνω συνθηκών κι αναγκών έχουν προταθεί στη βιβλιογραφία διάφοροι μέθοδοι μοντελοποίησης και πρόβλεψης της επίδοσης, στηριζόμενες σε αναλυτικά μοντέλα είτε και σε εμπειρική μοντελοποίηση. Σε αυτή την εργασία θα παρουσιάσουμε αρχικά μια τέτοια εμπειρική μέθοδο, που αναπτύχθηκε στο [2], η οποία κι αποσκοπεί στο να παρουσιάσει τις ανάγκες ορισμένων εφαρμογών σε συγκεκριμένους πόρους ως εύκολα αναγνώσιμες συναρτήσεις, εξαρτώμενες από το πλήθος των διαδικασιών που τρέχουν την εφαρμογή αλλά

κι από το μέγεθος του προβλήματος που αυτή καλείται να επιλύσει. Με αυτό τον τρόπο, οι σχεδιαστές τόσο των εφαρμογών όσο και των συστημάτων, μπορούν να πειραματιστούν με παραμέτρους όπως η διαχείριση της μνήμης, ο καταμερισμός των υπολογισμών, κ.ά. από τη μία και τη μνήμη του συστήματος, την αρχιτεκτονική του δικτύου, κ.ά. αντίστοιχα, προκειμένου να επιτύχουν καλύτερες επιδόσεις των εφαρμογών ή των συστημάτων. Εκτός από τις εφαρμογές οι οποίες μελετήθηκαν στο [2], παρουσιάζουμε τις δικές μας παρατηρήσεις για δύο ακόμα εφαρμογές, παρόμοιου τύπου, επιβεβαιώνοντας περαιτέρω τη σημασία μιας τέτοιας έρευνας. Επιπλέον, επεκτείνουμε αυτή την ιδέα, επιχειρώντας να μοντελοποιήσουμε με έναν παρόμοιο εμπειρικό τρόπο το χρόνο εκτέλεσης των ίδιων εφαρμογών, πειραματιζόμενοι σε διαφορετικά μηχανήματα. Αντί να προσπαθήσουμε να εκφράσουμε το χρόνο εκτέλεσης ως συνάρτηση του πλήθους διαδικασιών και του μεγέθους προβλήματος, επιχειρούμε να τον παρουσιάσουμε ως συνάρτηση των αναγκών των εφαρμογών στους πόρους που προκύπτουν από τα μοντέλα πρόβλεψης που περιγράφηκαν. Έτσι, "χτίζουμε" μια εμπειρική μοντελοποίηση του χρόνου εκτέλεσης της εφαρμογής πάνω στην εμπειρική μοντελοποίηση των πόρων που αυτή έχει ανάγκη. Τέλος, επιχειρούμε να δημιουργήσουμε μοντέλα πρόβλεψης για μια πιο σύνθετη εφαρμογή, η οποία επιλύει ένα πρόβλημα σε γράφο, οδηγούμενοι έτσι σε μοντελοποίηση πολυπαραμετρική ως προς την είσοδο ενός προβλήματος, σε αντίθεση με τη μονοδιάστατη είσοδο προβλημάτων που εξετάζεται στα αρχικά στάδια της εργασίας. Μια τέτοια έρευνα έχει επίσης σημασία καθώς τα προβλήματα σε γράφο αποτελούν μια σημαντική υποκατηγορία προβλημάτων όπου η πρόβλεψη επίδοσης είναι ιδιαίτερα προκλητική, αφού οι αντίστοιχοι αλγόριθμοι βασίζονται ιδιαιτέρως στην είσοδο αλλά, στην περίπτωση παράλληλων εφαρμογών, και στη διάσπαση του προβλήματος στους κόμβους υπολογισμού.

## 1.2 Διάρθρωση της Εργασίας

Η εργασία αυτή αποτελείται από έξι μέρη, καθένα με τα δικά του κεφάλαια.

- Το **μέρος 2** τιτλοφορείται ως "Θεωρητικό Υπόβαθρο-Εργαλεία" και περιλαμβάνει μια παρουσίαση θεωρητικών γνώσεων, χρήσιμες για την κατανόηση της εργασίας, καθώς και ορισμένες εξηγήσεις για τα εργαλεία και την αρχιτεκτονική των συστημάτων που χρησιμοποιήθηκαν στο πειραματικό μέρος αυτής.

- Στο **μέρος 3** παρουσιάζεται η εμπειρική μεθοδολογία που χρησιμοποιήθηκε για την κατασκευή μοντέλων πρόβλεψης της επίδοσης των εφαρμογών ως προς συγκεκριμένους πόρους του συστήματος. Επιπλέον αυτό το μέρος περιλαμβάνει και τα πειραματικά μας αποτελέσματα για τέσσερις εφαρμογές.

- Το **μέρος 4** περιλαμβάνει την επέκταση της παραπάνω εμπειρικής μεθοδολογίας με σκοπό την πρόβλεψη χρόνου εκτέλεσης των ίδιων τεσσάρων εφαρμογών, βασισμένη στις προβλέψεις για ανάγκες σε πόρους που παρουσιάζονται στο μέρος 3. Σε αυτό το μέρος εμπεριέχονται και οι αντίστοιχες πειραματικές μετρήσεις μας για τις εφαρμογές. Στο τέλος αυτού του μέρους επιχειρούμε μια μελέτη συν-σχεδίασης εξερευνώντας με ποιο τρόπο θα επωφελούνταν οι τέσσερις εφαρμογές από τρεις πιθανές αναβαθμίσεις συστήματος.

- Στο **μέρος 5** επιχειρούμε την εφαρμογή της μεθοδολογίας που παρουσιάστηκε στο μέρος 3 με σκοπό την πρόβλεψη των επιδόσεων μιας πιο σύνθετης εφαρμογής.
- Τέλος, στο **μέρος 6**, παρουσιάζονται συγκεντρωμένα ορισμένα γενικά συμπεράσματα από τις πειραματικές προσπάθειες αυτής της εργασίας.

# Μέρος II

# Θεωρητικό Υπόβαθρο-Εργαλεία

Κεφάλαιο 2

# Θεωρητικό Υπόβαθρο-Εργαλεία

## 2.1 Υπολογιστικά Συστήματα Μεγάλης Κλίμακας

Αντικείμενο της εργασίας είναι η πρόβλεψη και η προβολή της επίδοσης εφαρμογών που εκτελούνται σε συστήματα μεγάλης κλίμακας. Τέτοια συστήματα αποτελούνται από εγκαταστάσεις εκατοντάδων ή χιλιάδων υπολογιστικών κόμβων, διασυνδεδεμένων μέσω κάποιου δικτύου διασύνδεσης. Ακόμη, τα συστήματα αυτά, "επιστρατεύουν" πληθώρα λογισμικού, από λειτουργικό σύστημα μέχρι βιβλιοθήκες εφαρμογών, για τη λειτουργία τους.

Κύρια δομική μονάδα ενός τέτοιου υπολογιστικού συστήματος είναι οι κόμβοι. Σε αυτούς συνυπάρχει ένας αριθμός από επεξεργαστές οι οποίοι διασυνδέονται μεταξύ τους και διαθέτουν τη δική τους μνήμη. Η μνήμη αυτή μπορεί είτε να είναι κοινή για όλους είτε να μοιράζεται ανάμεσα τους, ενώ οι επεξεργαστές δύναται να διατηρούν κι έναν αριθμό από κρυφές μνήμες. Σε περίπτωση που ένας κόμβος φιλοξενεί περισσότερους από έναν κόμβους μνήμης, τότε έχουμε μια NUMA (Non-Uniform Memory Access) αρχιτεκτονικη, κάτι μπορεί να μειώσει τον χρόνο πρόσβασης στη μνήμη. Όσον αφορά τον τρόπο διασύνδεσης των επεξεργαστών συνήθως αυτοί συνδέονται με απευθείας συνδέσεις (point to point), γεγονός που μειώνει τον χρόνο επικοινωνίας μεταξύ τους.

Οι διάφοροι κόμβοι του συστήματος συνδέονται μεταξύ τους μέσω δικτύων διασύνδεσης υψηλών επιδόσεων. Αυτά έχουν αντίκτυπο στην επικοινωνία μεταξύ των κόμβων, κι επομένως στον τρόπο με τον οποίο υλοποιούνται οι ανάγκες επικοινωνίας των παράλληλων εφαρμογών. Υπάρχει πληθώρα διαφορετικών αρχιτεκτονικών σε ό,τι αφορά τα δίκτυα διασύνδεσης, οι οποίες και καθορίζουν σημαντικές παραμέτρους, όπως ο χρόνος απόκρισης, το εύρος ζώνης, το μέγεθος και τη δυνατότητα αποθήκευσης των διακοπτών αλλά και τους ίδιους τους μηχανισμούς δρομολόγησης.

Τελευταίο κομμάτι ενός υπολογιστικού συστήματος μεγάλης κλίμακας αποτελεί η στοίβα λογισμικού που το στελεχώνει. Αυτή περιλαμβάνει το λειτουργικό σύστημα, το λογισμικό διαχείρισης δικτύου, εργαλεία ανάπτυξης εφαρμογών, μεταγλωττιστές, βιβλιοθήκες παράλληλου προγραμματισμού κι επικοινωνιών, μαθηματικές βιβλιοθήκες και βιβλιοθήκες εφαρμογών. Η στοίβα λογισμικού επηρεάζει ποικιλοτρόπως τους παράγοντες που αφορούν την επίδοση μιας εφαρμογής, π.χ. ο διαχειριστής πόρων λαμβάνει αποφάσεις σχετικά με τη δέσμευση συγκεκριμένων πόρων για την απεικόνιση διεργασιών σε κόμβους κι επεξεργαστές, οι βιβλιοθήκες εφαρμογών μπορεί να πραγματοποιούν δικές τους πολιτικές διαχείρισης μνήμης, κ.ά.

## 2.2 Κλιμάκωση

Πρωτεύον ζήτημα των παράλληλων υπολογισμών είναι η ανάπτυξη αλγορίθμων και εφαρμογών που θα κλιμακώνουν ικανοποιητικά ενόσω οι παράλληλες αρχιτεκτονικές εξελίσσονται. Αυτό σημαίνει ότι οι αλγόριθμοι οφείλουν να εκμεταλλεύονται με τον βέλτιστο τρόπο τους πόρους των νέων συστημάτων προκειμένου να επιλύουν μεγαλύτερα προβλήματα σε μικρότερο χρόνο. Δύο ειδών "κλιμακώσεις" απατώνται συνήθως στη βιβλιογραφία για παράλληλες εφαρμογές: η δυνατή (strong) και η αδύναμη (weak).

Η **δυνατή** κλιμάκωση ορίζεται ως η προσπάθεια επίλυσης ενός προβλήματος γρηγορότερα αυξάνοντας το νούμερο των διαδικασιών που επιδίδονται στην επίλυση αυτού. Η μέγιστη, θεωρητική, επιτάχυνση ενός προγράμματος, χρησιμοποιώντας δυνατή κλιμάκωση, εκφράζεται, μέσω του νόμου του `Amdahl` [3], συνδέοντας την με τον βαθμό στον οποίο ένα πρόγραμμα μπορεί να παραλληλοποιηθεί, όπως βλέπουμε στην εξίσωση 2.1.

$$speedup = \frac{1}{(1-p) + \frac{p}{procs}} \tag{2.1}$$

όπου

- speedup η θεωρητική επιτάχυνση όλης της εφαρμογής
- procs ο αριθμός των διαδικασιών
- p το μέρος του χρόνου εκτέλεσης που καταλάμβανε το κομμάτι της εφαρμογής που επωφελείται από την αύξηση των διαδικασιών

Η **αδύναμη** κλιμάκωση ορίζεται ως η επίλυση ενός μεγάλου προβλήματος με αυξανόμενο αριθμό διαδικασιών, έτσι ώστε το πρόβλημα που ανατίθεται σε κάθε μία από αυτές να παραμένει σταθερό. Με αυτόν τον τρόπο ένα μεγαλύτερο μέρος διαδικασιών χρησιμοποιείται πιο αποτελεσματικά, αφού είναι εφικτή η καλύτερη επίδοση με καλύτερο καταμερισμό της εργασίας. Ωστόσο, το παραπάνω δεν αποτελεί μια εύκολη διαδικασία καθώς η αποδόμηση ενός προβλήματος σε μικρότερα και η ανάθεση αυτών σε διαφορετικές διαδικασίες γεννά αρκετές δυσκολίες.

Δυστυχώς το να επιτευχθεί ικανοποιητική κλιμάκωση, τόσο δυνατή ή αδύναμη, αποτελεί μια δύσκολη διαδικασία. Ο εντοπισμός σημείων σε εφαρμογές που ευθύνονται για την κακή κλιμάκωση της είναι επίπονη και δαπανηρή διαδικασία. Η αφαίρεση αυτών απαιτεί πληθώρα πειραμάτων σε μεγάλα συστήματα και δύσκολες παρεμβάσεις στον κώδικα των εφαρμογών. Αυτό γίνεται ακόμα πιο εμφανές όσο βαδίζουμε σε συστήματα επιδόσεων της τάξης των ExaFLOPS, καθώς οι εφαρμογές οφείλουν να αντιμετωπίσουν διαφορετικές προκλήσεις ταυτόχρονα, όπως νέες αρχιτεκτονικές. Τέλος, συνήθως τέτοια προβλήματα ανακύπτουν στα τελευταία στάδια της ανάπτυξης των εφαρμογών, όπου πλέον η αντιμετώπιση τους απαιτεί σημαντικές αλλαγές σε κώδικα ο οποίος έχει αναπτυχθεί ήδη για αρκετό καιρό.

## 2.3 Ανάλυση Επίδοσης

Η βελτίωση του χρόνου εκτέλεσης μιας εφαρμογής ή η αποδοτική χρήση των υφιστάμενων πόρων από πλευράς αυτής αποτελούν μια δύσκολη διεργασία. Προκειμένου να κατανοήσουμε τη συμπεριφορά παράλληλων εφαρμογών επιδιδόμαστε στην ανάλυση της επίδοσης αυτών, ούτως ώστε να αναδειχθούν όποιοι περιορισμοί υπάρχουν για την εφαρμογή ή για το σύστημα στο οποίο αυτή θα εκτελεστεί. Με αυτό τον τρόπο, οι άνθρωποι που αναπτύσσουν το λογισμικό ή τα μηχανήματα δύναται να επικεντρώσουν τις προσπάθειες τους στη στοχευμένη βελτίωση μερών που έχουν σημασία.

Ανάλογα με τον τρόπο με τον οποίο γίνεται η ανάλυση επίδοσης μπορούμε να κατηγοριοποιήσουμε τις προσεγγίσεις σε **πειραματικές, προσομοιώσεις, αναλυτικά μοντέλα ή εμπειρικές προσεγγίσεις**. Καθεμία από αυτές έχει τα δικά της πλεονεκτήματα ή μειονεκτήματα ανάλογα με την επίδραση στα αποτελέσματα του αριθμού των παραμέτρων που λαμβάνονται υπόψιν. Συνήθως μεγαλύτερο πλήθος παραμέτρων συνιστά και ακριβέστερα αποτελέσματα, ενώ λιγότερες παράμετροι οδηγούν σε αύξηση του σφάλματος της ανάλυσης [4].

### 2.3.1 Πειραματικές Προσεγγίσεις

Η κλασσική προσέγγιση για ανάλυση επίδοσης είναι η εκτέλεση της εφαρμογής στο σύστημα που θα τρέξει, η χρησιμοποίηση δηλαδή του κώδικα ως σημείο αναφοράς (benchmark) της επίδοσης του. Με πειραματικές προσεγγίσεις οι δημιουργοί των εφαρμογών καταγράφουν διαφορετικές μετρικές, όπως π.χ. ο χρόνος εκτέλεσης, και στη συνέχεια επικεντρώνονται στη βελτίωση των μερών του κώδικα που απαιτούν περισσότερους πόρους. Μια τέτοια προσέγγιση είναι περιορισμένη όσον αφορά τα περιθώρια βελτίωσης και δεν προσφέρει κάποια ξεκάθαρη προσδοκία για τις πραγματικές δυνατότητες της επίδοσης της εφαρμογής. Επιπλέον, μέσω αυτής της μεθόδου, είναι ιδιαίτερα δύσκολος ο διαχωρισμός των επιπτώσεων του κώδικα στην επίδοση από τις επιπτώσεις της αρχιτεκτονικής του συστήματος σε αυτήν.

### 2.3.2 Προσομοιώσεις

Μια πιο αφηρημένη προσέγγιση αποτελεί η προσομοίωση της εκτέλεσης της εφαρμογής με διαφορετικούς βαθμούς ακρίβειας. Αυτό επιτρέπει στον ενδιαφερόμενο να κατανοήσει και να ποσοτικοποιήσει την επίδραση της προσομοιούσας αρχιτεκτονικής στην επίδοση της εφαρμογής. Το μειονέκτημα μιας τέτοιας προσέγγισης είναι πως οι ακριβείς προσομοιώσεις είναι ιδιαίτερα πιο χρονοβόρες, συγκριτικά με την κανονική εκτέλεση του κώδικα, ενώ συνήθως οι γρήγορες προσομοιώσεις δεν αναδεικνύουν το περιβάλλον του προβλήματος με ακρίβεια, κι έτσι μπορεί να οδηγήσουν σε εσφαλμένες αποφάσεις περί βελτίωσης της εφαρμογής. Τέλος οι προσομοιώσεις προσφέρουν πλήθος από δεδομένα τα οποία πρέπει να οργανωθούν και να μελετηθούν διεξοδικά, περιπλέκοντας επιπλέον τη διαδικασία απόκτησης χρήσιμων αποτελεσμάτων.

### 2.3.3 Αναλυτική Μοντελοποίηση

Μια άλλη μέθοδος ανάδειξης προβλημάτων επίδοσης είναι η αναλυτική μοντελοποίηση αυτής. Κατά αυτή τη μέθοδο η επίδοση, όσον αφορά μια μετρική, εκφράζεται μέσω αναλυτικών εκφράσεων. Για παράδειγμα, ο χρόνος εκτέλεσης μπορεί να εκφράζεται μέσω αναλυτικής μοντελοποίησης συναρτήσει του αριθμού των διαδικασιών στις οποίες καταμερίζεται η εκτέλεση της εφαρμογής.

Συνήθως, προκειμένου να μοντελοποιήσουν την κλιμάκωση του κώδικα τους, οι άνθρωποι που τον αναπτύσσουν εκτελούν δοκιμές σε μικρότερη κλίμακα προκειμένου να καταλήξουν σε λεγόμενους "φλοιούς", οι οποίοι αποτελούν τα μέρη του κώδικα που αναμένουν να επηρεάζουν την απόδοση του σε μεγαλύτερη κλίμακα. Έπειτα, χρησιμοποιώντας λογική επιδίδονται στη χρονοβόρα διαδικασία της δημιουργίας αναλυτικών μοντέλων που θα περιγράφουν τη κλιμάκωση του κάθε "φλοιού". Το πρόβλημα της παραπάνω τακτικής είναι προφανές καθώς η ανάπτυξη μοντέλων, όντας χρονοβόρα, δεν γίνεται για παραπάνω από έναν αριθμό "φλοιών", οδηγώντας έτσι σε παραμέληση μερών του κώδικα τα οποία μπορεί με τη σειρά τους να επιδεικνύουν κακή κλιμάκωση.

Μια εφαρμογή της μοντελοποίησης είναι η διαδικασίας της "συν-σχεδίασης" (co-design) [5]. Ο όρος αυτός περιγράφει την ενοποιημένη ανάπτυξη υλικού, υπολογιστικών εφαρμογών και του αντίστοιχου λογισμικού. Κάτι τέτοιο κρίνεται αναγκαίο καθώς, δεδομένου του μεγάλου κόστους των συστημάτων μεγάλης κλίμακας, η αρχιτεκτονική τους οφείλει να ανταποκρίνεται στις απαιτήσεις των εφαρμογών που εκτελούνται σε αυτά, ενώ και οι ίδιες οι εφαρμογές πρέπει να σχεδιάζονται έτσι ώστε να μπορούν να εκτελεστούν σε συστήματα τα οποία είναι εφικτό να δημιουργηθούν.

### 2.3.4 Εμπειρική Μοντελοποίηση

Συνδυάζοντας ιδέες από την αναλυτική μοντελοποίηση και την πειραματική προσέγγιση, η εμπειρική μοντελοποίηση στοχεύει στην παραγωγή αναλυτικών μοντέλων που εκφράζουν το χρόνο εκτέλεσης ή την χρήση πόρων βασιζόμενα σε πειραματικές μετρήσεις, οι οποίες αποκτώνται με διαδοχικές εκτελέσεις στις οποίες μεταβάλλονται οι παράμετροι των μοντέλων (όπως π.χ. το πλήθος διαδικασιών ή το μέγεθος του προβλήματος).

Η εμπειρική μοντελοποίηση πρέπει ωστόσο να έρθει αντιμέτωπη με μια σειρά προκλήσεων. Τα μοντέλα οφείλουν να ανταποκρίνονται σε πληθώρα παραμέτρων, ενώ ο αριθμός των απαιτούμενων μετρήσεων μπορεί να είναι περιοριστικός, ιδίως αν πρέπει να ληφθούν υπόψιν πολλές παράμετροι. Επιπλέον, η συλλογή μετρήσεων και η δημιουργία αντίστοιχων εμπειρικών μοντέλων αναφέρεται στο σύνολο της εφαρμογής, και όχι σε κομμάτια αυτής.

## 2.4 Ανάλυση Μηχανημάτων-Εργαλείων

### 2.4.1 Μηχανήματα που χρησιμοποιήθηκαν στην εργασία

Προκειμένου να αποκτήσουμε τις πειραματικές παρατηρήσεις που χρειάστηκαν για την εξαγωγή των μοντέλων πρόβλεψης, πραγματοποιήσαμε μετρήσεις σε τρία συστήματα. Σε

αυτό το σημείο θα παρουσιάσουμε στους κατώθι πίνακες τα τεχνικά χαρακτηριστικά αυτών των τριών συστοιχιών.

### Συστοιχία Clones

Αυτή αποτελεί μια απλού τύπου συστοιχία με σχετικά παλιά μηχανήματα. Αποτελείται από 26 συνολικά κόμβους, τα τεχνικά χαρακτηριστικά καθενός μπορούν να βρεθούν παρακάτω:

| `Reference Name` | `clone` |
|---|---|
| CPU | 2· Intel Xeon E5335 |
| Processor Base Frequency | 2.00GHz |
| Number of Cores | $2 \cdot 4$ |
| Number of Threads | $2 \cdot 4$ |
| L1(data) cache/core | 32KB |
| L2 cache/core | 4MB |
| RAM | 2-8GB (depending on node) |
| OS | GNU Linux |

Πίνακας 2.1: *Τεχνικά χαρακτηριστικά κάθε κόμβου της συστοιχίας clones*

### Sandman

Πρόκειται για μια πιο σύγχρονη πλατφόρμα από αυτή των clones και η οποία χρησιμοποιήθηκε για να μετρήσουμε το πλήθος εντολών πρόσβασης στη μνήμη (Loads & Stores) που εκτελούν οι εφαρμογές, αλλά και για την εξ ολοκλήρου εκτέλεση της πιο σύνθετης εφαρμογής στο μέρος 5. Παρακάτω μπορούμε να δούμε τα τεχνικά χαρακτηριστικά αυτής.

| `Reference Name` | `Sandman` |
|---|---|
| CPU | 4· Intel Xeon E5-4620 |
| Processor Base Frequency | 2.20GHz |
| Number of Cores | $4 \cdot 8$ |
| Number of Threads | $4 \cdot 16$ |
| L1(data) cache/core | 32KB |
| L2 cache/core | 256KB |
| L3 cache/socket | 16MB |
| RAM | 256GB |
| OS | Debian 8.8 |

Πίνακας 2.2: *Τεχνικά χαρακτηριστικά κάθε κόμβου της συστοιχίας sandman*

### Μηχάνημα 3

Τέλος χρησιμοποιήσαμε μια ακόμη συστοιχία, με πιο μοντέρνα χαρακτηριστικά, προκειμένου να λάβουμε πιο αξιόπιστες μετρήσεις για τον χρόνο εκτέλεσης των εφαρμογών που παρουσιάζεται στο μέρος 4.

| Reference Name | node συστοιχίας |
|---|---|
| CPU | 2· Intel Xeon E5-2697 v3 |
| Processor Base Frequency | 2.60GHz |
| Number of Cores | $2 \cdot 14$ |
| Number of Threads | $2 \cdot 28$ |
| L1(data) cache/core | 32KB |
| L2 cache/core | 256KB |
| L3 cache/socket | 35MB |
| RAM | 64-256GB |
| OS | GNU Linux |

Πίνακας 2.3: *Τεχνικά χαρακτηριστικά κάθε κόμβου της συστοιχίας Μηχάνημα 3*

### 2.4.2 Εργαλεία που χρησιμοποιήθηκαν στην εργασία

Στην παρούσα εργασία επιστρατεύουμε ορισμένα εργαλεία για την συλλογή των μετρήσεων που χρειαζόμαστε προκειμένου να καταλήξουμε στα μοντέλα πρόβλεψης που παρουσιάζονται.

#### Score-P

Το Score-P [6] αποτελεί ένα εργαλείο που επιτρέπει στον χρήση να λαμβάνει μετρήσεις μέσω της συλλογής ιχνών (tracing) από παράλληλες εφαρμογές κατά την εκτέλεσή τους κι ανήκει στη σουίτα της Scalasca [7]. Η συλλογή ιχνών [8] στην περίπτωση των παράλληλων εφαρμογών είναι μια περίπλοκη διαδικασία καθώς εξαρτάται σε μεγάλο βαθμό από την αλληλεπίδραση μεταξύ των διεργασιών αλλά κι από το ίδιο το σύστημα στο οποίο θα εκτελεστεί. Στόχος του Score-P είναι να απλουστεύσει την ανάλυση της συμπεριφοράς των παράλληλων εφαρμογών, επιτρέποντας έτσι την εύρεση "μελανών σημείων" στον κώδικα αυτών. Επιπλέον, το Score-P επιτρέπει και την οπτικοποίηση των ιχνών που συλλέγονται ενώ ακόμη χρησιμοποιεί τον ίδιο instrumenter με άλλα εργαλεία ανάλυσης, όπως το Tau [9], ξεπερνώντας έτσι την ανάγκη για πολλαπλές επαναλήψεις της διαδικασίας ανάλυσης της εφαρμογής.

#### PAPI

Το PAPI (Performance Application Programming Interface) [10] προσφέρει στον χρήστη "φορητή" (portable), δηλαδή ανεξάρτητη του μηχανήματος ή του λειτουργικού συστήματος, πρόσβαση σε μετρητές επίδοσς υλικού (hardware performance counters), οι οποίοι απαντώνται στους περισσότερους μοντέρνους επεξεργαστές. Το PAPI αντιστοιχίζει αυτούς τους μετρητές σε αυτό που ονομάζει "γεγονότα" (events), επιτρέποντας στον χρήστη να λάβει τις απαραίτητες μετρήσεις είτε μέσω ενός API υψηλού επιπέδου είτε μέσω ενός χαμηλού επιπέδου, το οποίο κι επιτρέπει μεγαλύτερη ελευθερία ως προς τη λήψη μετρήσεων σε συγκεκριμένα κομμάτια του κώδικα κ.ά.

## 2.5 Σχετική Βιβλιογραφία

Η ανάγκη μοντελοποίησης των επιδόσεων εφαρμογών που τρέχουν σε συστήματα μεγάλης κλίμακας αποτελεί, όπως αναδεικνύεται κι από τις παρατηρήσεις μας στην ενότητα 1.1, ένα ση-

μαντικό αντικείμενο για την επιστημονική κοινότητα, και για αυτό τον λόγο έχει μελετηθεί εκτενώς στη βιβλιογραφία. Πιο συγκεκριμένα, έχει προταθεί πληθώρα τεχνικών πρόβλεψης και μοντελοποίησης, που στηρίζονται είτε σε αναλυτικά, είτε σε εμπειρικά μοντέλα, επιχειρώντας προβλέψεις σε συγκεκριμένες αρχιτεκτονικές κι εφαρμογές ή αγνοώντας αυτές. Στο [4] παρουσιάζονται ιδέες γύρω από αναλυτικά μοντέλα, καθώς και τα περιθώρια λαθών στη μοντελοποίηση όταν μεταβάλλεται το πλήθος των παραμέτρων ενδιαφέροντος.

Έχει επιχειρηθεί αρκετές φορές μια πρόβλεψη της επίδοσης εφαρμογών με τρόπους που δεν λαμβάνουν υπόψιν την αρχιτεκτονική του συστήματος και οι οποίες δεν επιχειρούν προβολή των προβλέψεων τους σε μεγαλύτερα συστήματα [11], [12]. Προκειμένου να επιτευχθεί η προβολή επίδοσης (performance extrapolation) των εφαρμογών έχουν προταθεί διάφορες μέθοδοι, όπως η συλλογή δεδομένων για την εφαρμογή μέσω εκτελέσεων της σε λίγους πυρήνες και η προβολή της σε μεγαλύτερο πλήθος μέσω απλών στατιστικών μεθόδων, π.χ. παλινδρόμηση (reggression) [13]. Άλλη μέθοδος που έχει προταθεί είναι η μερική εκτέλεση της εφαρμογής σε διάφορες αρχιτεκτονικές και η κατάληξη σε συμπεράσματα μέσω της σύγκρισης των αποτελεσμάτων [14]. Επιπλέον έχει προταθεί η εμπειρική μοντελοποίηση μερών πολύπλοκων εφαρμογών, με σκοπό την ανάδειξη της επίδοσης αυτών [4], μια ιδέα που επεκτείνεται στο [15], με σκοπό την εμπειρική μοντελοποίηση βιβλιοθηκών που απαντώνται με μεγαλύτερη συχνότητα σε παράλληλες εφαρμογές.

Τέλος, έχει αποπειραθεί η εξαγωγή απλών μοντέλων μέσω εμπειρικών μεθόδων με σκοπό την εκτίμηση των αναγκών μιας εφαρμογής σε πόρους [2]. Οι μέθοδοι αυτοί, βασιζόμενοι σε εμπειρικές παρατηρήσεις για τη μορφή των εξισώσεων που περιγράφουν τις προβλέψεις, επιχειρούν να συνδέσουν τις παρατηρήσεις με μοντέλα πολλών παραμέτρων, όπως το μέγεθος του προβλήματος, το πλήθος των πυρήνων, κ.ά. [16]. Αυτό έχει ως απώτερο σκοπό να συμβάλλει στις διαδικασίες “συν-σχεδίασης” (co-design) [5] των συστημάτων νέας γενιάς, λαμβάνοντας υπόψιν τη φύση και τις ανάγκες των εφαρμογών που τρέχουν σε αυτά.

# Μέρος III

# Μοντελοποίηση Αναγκών

Κεφάλαιο 3

# Μοντελοποίηση αναγκών σε πόρους

Σε αυτό το κεφάλαιο θα παρουσιάσουμε τον τρόπο εξαγωγής των εμπειρικών μοντέλων τα οποία κι εκφράζουν τις προϋποθέσεις που ερευνώνται για την πρόβλεψη της επίδοσης των εφαρμογών που εξετάζονται. Η διαδικασία που περιγράφεται είναι η ίδια που χρησιμοποιήθηκε τόσο για την εξαγωγή των μοντέλων που περιγράφουν τις ανάγκες σε πόρους των εφαρμογών όσο και για τα μοντέλα που επιχειρούν να προβάλουν τον χρόνο εκτέλεσης τους.

## 3.1 Ανάγκες σε Πόρους Εφαρμογών

Όσον αφορά την προσπάθεια για μοντελοποίηση των αναγκών των εφαρμογών, εδώ επιλέγεται η καταγραφή μετρικών οι οποίες επικεντρώνονται ξεκάθαρα στις εφαρμογές καθαυτές, χωρίς να γίνεται κάποια υπόθεση για το hardware των μηχανημάτων. Αυτό έχει σημασία για τη χρησιμότητα των αποτελεσμάτων, καθώς αυτές οι μετρικές μας αναδεικνύουν πιθανά "κολλήματα" των εφαρμογών, ανεξάρτητα από την αρχιτεκτονική του μηχανήματος που τις τρέχει. Επίσης, η προσπάθεια για δημιουργία μοντέλων που θα εξαρτώνταν από την εκάστοτε αρχιτεκτονική αυξάνει με την εκτέλεση εφαρμογών σε συστήματα διαφορετικών παραμέτρων, ενώ κάποιο μοντέλο που περιγράφει τις ανάγκες μιας εφαρμογής παραμένει το ίδιο ανεξάρτητα από το μηχάνημα που θα κληθεί να την εκτελέσει [2].

Επομένως εκφράζουμε τις ανάγκες σε πόρους των εφαρμογών ως συναρτήσεις της μορφής $r_i(n, p)$, όπου ως $r_i$ θεωρούμε την ανάγκη r ως προς έναν συγκεκριμένο πόρο i, ως n εννοούμε το μέγεθος του προβλήματος ανά διαδικασία και ως p εκφράζουμε το πλήθος των διαδικασιών. Στην περίπτωσή μας το πλήθος αυτό ταυτίζεται με τον αριθμό των πυρήνων που τρέχουν την εφαρμογή.

Στον πίνακα που ακολουθεί παρουσιάζουμε της μετρικές που συλλέξαμε καθώς και τους πόρους που κάθε μία από αυτές εκφράζει:

| Πόρος | Μετρική |
|---|---|
| Αποτύπωμα Μνήμης | Αριθμός Bytes που χρησιμοποιήθηκαν |
| Υπολογισμοί | Αριθμός Πράξεων Κινητής Υποδιαστολής FLOPS |
| Επικοινωνία | Αριθμός Bytes που εστάλησαν/παραλήφθηκαν |
| Πρόσβαση στη Μνήμη | Αριθμός Εντολών Loads/Stores |

Πίνακας 3.1: *Πόροι και αντίστοιχες μετρικές αυτών*

## 3.2 Εξαγωγή Δεδομένων

Σε αυτό το σημείο είναι σημαντικό να περιγράψουμε τον τρόπο με τον οποίο εξάγουμε τα δεδομένα για τις μετρικές που περιγράφηκαν παραπάνω. Συγκεκριμένα για κάθε εφαρμογή χρειαζόμαστε μια υλοποίηση της που χρησιμοποιεί το MPI, ενώ για την καταγραφή των μετρικών χρησιμοποιήθηκαν τα εργαλεία `Score-P` [6] και `PAPI` [10]. Όλες οι μετρικές που θα παρουσιαστούν αναφέρονται και μετρήθηκαν για μια διαδικασία, μιας και οι αντίστοιχοι πόροι, όπως η μνήμη, οι συνδέσεις δικτύου, κτλ., αυξάνονται σχεδόν αναλογικά με τον αριθμό των διαδικασιών σε ένα μηχάνημα. Παρουσιάζουμε παρακάτω αναλυτικά τον τρόπο καταγραφής των μετρικών:

- *Αποτύπωμα στη μνήμη:* προκειμένου να λάβουμε τον αριθμό των bytes που χρησιμοποιήθηκαν από μια διαδικασία χρησιμοποιούμε το **getrusage()**. Το αποτύπωμα στη μνήμη αποτελεί έναν πολύ σημαντικό παράγοντα, καθώς μπορεί να φανεί αποτρεπτικός ως προς την εξ ολοκλήρου εκτέλεση μιας εφαρμογής.

- *Επικοινωνία:* χρησιμοποιούμε το εργαλείο `Score-P` προκειμένου να λάβουμε το πλήθος των bytes που εστάλησαν/παραλήφθησαν από τις διαδικασίες που τρέχουν την εφαρμογή μέσω των διαφόρων κλήσεων του MPI.

- *Υπολογισμοί:* κάνουμε χρήση του εργαλείου `PAPI` με σκοπό να καταγράψουμε το πλήθος των πράξεων κινητής υποδιαστολής (FLOPS) που γίνονται από κάθε διαδικασία.

- *Πρόσβαση στη μνήμη:* και σε αυτή την περίπτωση χρησιμοποιούμε το `PAPI` έτσι ώστε να μετρήσουμε τον αριθμό των συνολικών εντολών load και store στο σύνολο του προγράμματος.

## 3.3 Παραγωγή Μοντέλων

### 3.3.1 Μέθοδος παραγωγής των μοντέλων

Σε αυτή την υπό-ενότητα θα παρουσιάσουμε αναλυτικά τον τρόπο με τον οποίο παράγονται τα εμπειρικά μοντέλα μας. Η ιδέα πίσω από τη μέθοδο παραγωγής τον μοντέλων μας βρίσκεται στο εργαλείο `Extra-P` [17], [16] το οποίο και πρώτο-δημιουργήθηκε προκειμένου να βοηθήσει προγραμματιστές ώστε να βρίσκουν πιθανά "μελανά σημεία" στους κώδικες τους που μπορεί να οδηγήσουν σε αδυναμία εκτέλεσης αυτών.

Προκειμένου να εξηγήσουμε πώς καταλήγουμε σε μοντέλα που περιγράφονται ως συναρτήσεις των (n,p) θα εξηγήσουμε πρωτίστως πώς μπορούμε να λάβουμε ένα μαθηματικό μοντέλο για μία μόνο παράμετρο. Μια σημαντική ιδέα ως προς αυτό είναι η *'Κανονική Φόρμα Μοντέλων Απόδοσης' (Performance Model Normal Form, PMNF)*[16], η οποία και παρουσιάζεται στην εξίσωση 3.1.

$$f(x) = \sum_{k=1}^{n} c_k \cdot x^{i_k} \cdot log_2^{j_k}(x) \tag{3.1}$$

Η εξίσωση αυτή περιγράφει την επίδραση μιας παραμέτρου $x$ σε μια ποσότητα ενδιαφέροντος *f(x)*, συνήθως μια μετρική απόδοσης όπως ο χρόνος εκτέλεσης ή κάποιος από τους πόρους

που περιγράφηκαν παραπάνω. Μέσω αυτής της εξίσωσης μπορούμε να λάβουμε έναν χώρο αναζήτησης προκειμένου να καταλήξουμε στη συνάρτηση η οποία και περιγράφει καλύτερα το σύνολο μετρήσεων που έχουμε συλλέξει. Προκειμένου να καταλήξουμε σε μια *υπόθεση* πρέπει να αναθέσουμε τιμές στα $i_k$ και $j_k$, καθώς και να υπολογίσουμε τον συντελεστή $c_k$. Οι εκθέτες λαμβάνουν τιμές από τα σύνολα $I = \{0, 1/4, 1/2, \ldots, 3\}$ για το $i_k$ και $J = \{0, 1, 2\}$ για το $j_k$ αντίστοιχα, καθώς γνωρίζουμε ότι κανένα από τα δύο σύνολα δεν χρειάζεται να είναι ιδιαίτερα μεγάλο για να επιτύχουμε καλό "ταίριασμα" του μοντέλου στα πραγματικά δεδομένα[16], κάτι που επιβεβαιώνουμε και με τη δική μας εμπειρία. Ακόμη, για λόγους ευκολίας, επιλέγουμε να μην εξετάσουμε άθροισμα διαφορετικών προβλέψεων, δηλαδή θέτουμε $n = 1$ στην εξίσωση 3.1. Στη συνέχεια, για να καταλήξουμε σε μια υπόθεση, μένει να προσδιορίσουμε το συντελεστή κάτι το οποίο κάνουμε αυτόματα με χρήση *παλινδρόμησης (Regression)*. Έχουμε τώρα στα χέρια μας $|I| \cdot |J|$ διαφορετικές υποθέσεις τις οποίες και πρέπει να αξιολογήσουμε με κάποιο τρόπο ώστε να καταλήξουμε στο μοντέλο που περιγράφει καλύτερα τα δεδομένα μας, χρειαζόμαστε με άλλα λόγια μια συνάρτηση αξιολόγησης. Για λόγους που θα γίνουν καλύτερα κατανοητοί στη συνέχεια, επιλέξαμε να μελετήσουμε και να παρουσιάσουμε μοντέλα που παράγονται από δύο διαφορετικές συναρτήσεις αξιολόγησης:

$$score_1 = R^2 \tag{3.2}$$

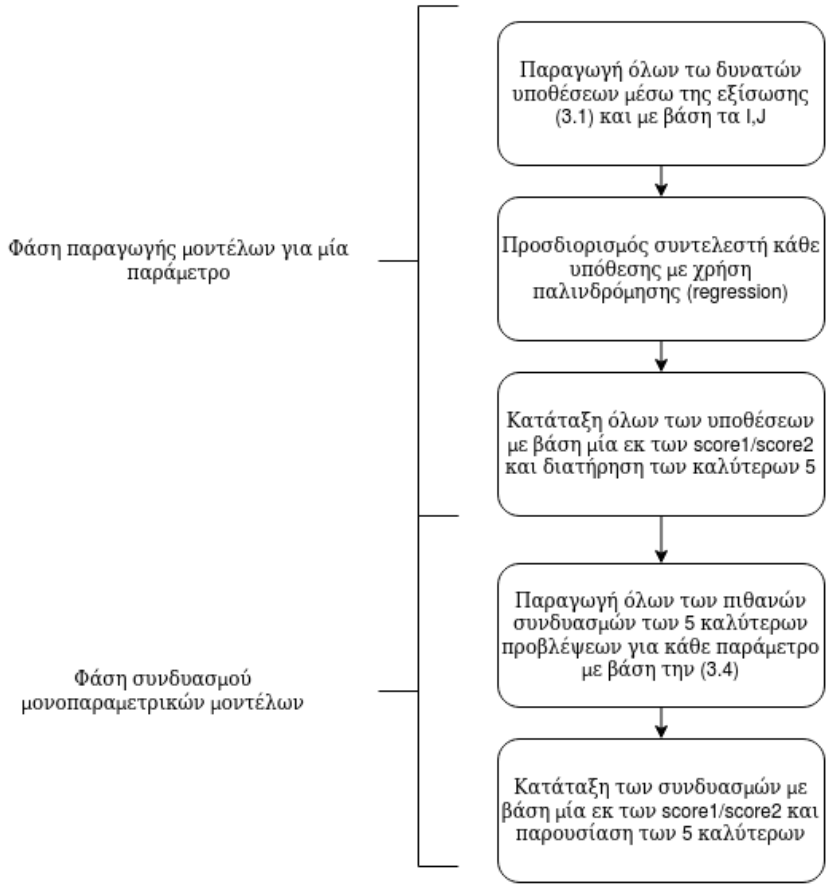
και

$$score_2 = R^2 \cdot RCC \cdot pred_{<0.35} \tag{3.3}$$

Λαμβάνουμε έτσι, αξιολογώντας είτε με βάση την 3.2 είτε με βάση την 3.3, μια ακολουθία ταξινομημένων μοντέλων μιας παραμέτρου για τα δεδομένα μας, από τα οποία και κρατάμε τα καλύτερα πέντε. Ακολουθούμε αυτή τη διαδικασία και για τις δύο παραμέτρους μας n και p ξεχωριστά. Προκειμένου να καταλήξουμε σε κάποιο μοντέλο που θα εξαρτάται κι από τις δύο παραμέτρους ακολουθούμε την παρατήρηση ότι "τα καλύτερα μοντέλα μιας παραμέτρου θα πρέπει να συνδυάζονται με κάποιο τρόπο δίνοντας τα καλύτερα μοντέλα και για τις δύο παραμέτρους" [16]. Έτσι πλέον η διαδικασία της εύρεσης το κατάλληλου μοντέλου πολλαπλών παραμέτρων εκφυλίζεται στην παραγωγή κάθε πιθανού αθροίσματος ή γινομένου των μοντέλων μιας παραμέτρου *(Εξίσωση 3.4)* και στην επιλογή, χρησιμοποιώντας μία εκ των 3.2 ή 3.3, του καλύτερου ταιριάσματος.

$$f(x_1, \ldots, x_m) = \sum_{k=1}^{n} c_k \cdot \prod_{l=1}^{m} x_l^{i_k l} \cdot log_2^{j_k l}(x_l) \tag{3.4}$$

Παρακάτω παραθέτουμε μια σχηματική απεικόνιση της διαδικασίας εξαγωγής των μοντέλων που περιγράφηκε παραπάνω:




Σχήμα 3.1: *Σχηματική αναπάρασταση της διαδικασίας εξαγωγής των μοντέλων πρόβλεψης*

### 3.3.2 Σύγκριση δύο διαφορετικών τρόπων αξιολόγησης μοντέλων

Σε αυτό το σημείο είναι σωστό να σταθούμε στους λόγους για τους οποίους επιλέξαμε να αξιολογήσουμε τα μοντέλα μας με δύο διαφορετικές συναρτήσεις αξιολόγησης, εξηγώντας τις διαφορές αυτών. Ενδιαφερόμαστε για τη δυνατότητα ενός μοντέλου να προβλέψει με ακρίβεια την τάση που ακολουθούν οι μετρήσεις που έχουμε λάβει καθώς και για το ελάχιστο δυνατό σφάλμα που λαμβάνουμε. Είναι γνωστό ότι καμία μετρική αξιολόγησης δεν δύναται να αποφανθεί απόλυτα σχετικά με την ακρίβεια και την καλή προσαρμογή ενός μοντέλου, παρ΄ όλα αυτά διαφορετικές μετρικές αναδεικνύουν διαφορετικές ιδιότητες της ακρίβειας ή της προσαρμογής ενός μοντέλου στην τάση των μετρήσεων μας, ενώ ο συνδυασμός ποικίλων μετρικών αξιολόγησης μπορεί να αποδειχθεί χρήσιμος για τη σύγκριση των μοντέλων.

*Στην ανάλυση που ακολουθεί, ορίζουμε ως $x_i$ την $i$-οστή παρατήρηση από τις μετρήσεις μας για μια μετρική $x$, ενώ ως $\hat{x}_i$ ορίζουμε την εκτίμηση του μοντέλου για την αντίστοιχη παρατήρηση.*

Ορίζουμε το σχετικό σφάλμα πρόβλεψης 3.5 ως εξής:

$$e_i = \frac{\hat{x}_i - x_i}{x_i} \tag{3.5}$$

Προκειμένου να εκτιμήσουμε την ακρίβεια ενός μοντέλου πρόβλεψης, εξετάζουμε το ποσοστό των προβλέψεων με σχετικό σφάλμα μικρότερο ή ίσο του 0.35 3.6, δηλαδή:

$$Pred_{0.35} = \frac{\#predictionswith|e_i| \leqq 0.35}{\#predictions} \tag{3.6}$$

Η μετρική $Pred_{0.35}$ λαμβάνει τιμές μεταξύ 0 και 1, με τις υψηλότερες τιμές να υποδηλώνουν και υψηλότερη ακρίβεια. Με αυτό τον τρόπο μπορούμε να προσδιορίσουμε την ακρίβεια της πρόβλεψης μας σε ένα σταθερό κατώφλι για το σχετικό σφάλμα (0.35 σε αυτή την περίπτωση). Έτσι, αν έχουμε μια τιμή κοντά στο 1, μπορούμε να αποφανθούμε ότι η ακρίβεια του μοντέλου είναι ιδιαίτερα ικανοποιητική για το συγκεκριμένο σύνολο σημείων, καθώς τα περισσότερα από τα σχετικά σφάλματα κυμαίνονται σε ένα εύρος $\pm 35\%$.

Μια ακόμη μετρική που εξετάζουμε είναι ο συντελεστής προσδιορισμού (coefficient of determination) $R^2$ 3.7:

$$R^2 = 1 - \frac{\sum_{i=1}^{m} (x_i - \hat{x_i})^2}{\sum_{i=1}^{m} (x_i - \bar{x_i})^2} \tag{3.7}$$

Αυτός λαμβάνει υψηλότερη τιμή για $R^2 = 1$, γεγονός που υποδηλώνει βέλτιστη προσαρμογή, αλλά δύναται να λάβει κι αρνητικές τιμές. Ο συντελεστής μας πληροφορεί για πόσο καλά το μοντέλο μας προσεγγίζει τις πραγματικές τιμές και σε υψηλές αποτιμήσεις του δηλώνει ότι τα προβλεπόμενα δεδομένα ταιριάζουν καλά με τα πραγματικά, μαρτυρώντας ακρίβεια και καλή προσαρμογή.

Τελευταία μετρική την οποία εξετάζουμε είναι ο συντελεστής συσχέτισης βαθμών (Rank Correlation Coefficient) $RCC$ 3.8:

$$RCC = \sum_{1 \leqq i \leqq m} \sum_{1 \leqq j \leqq m} concordant_{ij} / \frac{m(m-1)}{2} \tag{3.8}$$

όπου:

$$concordant_{ij} = \begin{cases} 1, & \text{αν } x_i < x_j \text{ και } \hat{x_i} < \hat{x_j} \\ 1, & \text{αν } x_i > x_j \text{ και } \hat{x_i} > \hat{x_j} \\ 0, & \text{αλλιώς} \end{cases} \tag{3.9}$$

Η μετρική $RCC$ λαμβάνει τιμές μεταξύ 0 και 1, με τις υψηλότερες τιμές να δηλώνουν και καλύτερη προσαρμογή του μοντέλου στις πραγματικές τιμές. Όπως υποδηλώνουν και οι 3.8, 3.9, η μετρική αυτή μαρτυρά πόσο καλά προβλέπεται η διάταξη των δεδομένων και μια υψηλή τιμή υποδηλώνει ένα μοντέλο το οποίο μπορεί να αναδείξει τις αυξομειώσεις στις τιμές των παρατηρήσεων, ανάλογα και με τις διαφορετικές ρυθμίσεις των πειραμάτων. Είναι μάλιστα εμφανής και η διαφορά μεταξύ $R^2$ και $RCC$, καθώς η πρώτη αξιολογεί τόσο την ακρίβεια του μοντέλου όσο και την προσαρμογή του, ενώ η δεύτερη ερευνά μόνο την προσαρμογή [18].

Αφού λοιπόν εξηγήσαμε αναλυτικά τις μετρικές αξιολόγησης μπορούμε να μελετήσουμε τις δύο συναρτήσεις που παρουσιάζονται στις 3.2 και 3.3. Η πρώτη μέθοδος επιστρατεύει μόνο την μετρική $R^2$ προκειμένου να αποφανθεί για την καταλληλότητα ενός μοντέλου, σε αντίθεση με τη δεύτερη μέθοδο η οποία βασίζεται σε ένα γινόμενο των τριών μετρικών. Η αξιολόγηση των μοντέλων με βάση μόνο τον συντελεστή 3.7 παρουσιάζεται ευρέως στη βιβλιογραφία, [2], [17], [16], και, ενώ φυσικά οδηγεί σε αρκετά ικανοποιητικά αποτελέσματα, όπως θα παρουσιάσουμε

και στα επόμενα κεφάλαια, φαίνεται να υστερεί απέναντι στον δεύτερο τρόπο αξιολόγησης. Αυτό μπορεί να αιτιολογηθεί από την παραπάνω ανάλυση των τριών μετρικών, καθώς ένας συνδυασμός αυτών μπορεί να αναδείξει τα διαφορετικά χαρακτηριστικά των μοντέλων τα οποία κι αξιολογεί η κάθε μετρική και είναι θεμιτά για την ταυτοποίηση της καλύτερης πρόβλεψης. Αν, για παράδειγμα, λαμβάναμε υπόψιν μόνο τον συντελεστή συσχέτισης $R^2$ τότε είναι πιθανό να καταλήγαμε σε ένα μοντέλο το οποίο να υστερεί έναντι ενός άλλου, το οποίο θα λάμβανε υπόψιν και το $RCC$, σχετικά με την προσαρμογή στην τάση των μετρήσεων.

Κεφάλαιο 4

# Μοντελοποίηση αναγκών τεσσάρων εφαρμογών

Στο κεφάλαιο αυτό θα παρουσιάσουμε τα αποτελέσματα μας για την μοντελοποίηση τεσσάρων εφαρμογών, με βάση τη μεθοδολογία που παρουσιάσθηκε εκταινώς στο προηγούμενο κεφάλαιο. Για τη μελέτη μας επιλέξαμε τις εξής εφαρμογές:

- *LULESH* [19]
- *Kripke* [20]
- *Pennant* [21]
- *AMG2013* [22]

Λάβαμε τις μετρήσεις μας στην ουρά clones και στο μηχάνημα sandman.

Στη συνέχεια θα παρουσιάσουμε ξεχωριστά για κάθε εφαρμογή τα αποτελέσματα των μετρήσεων και της μοντελοποίησής μας. Σε κάθε μοντέλο που ακολουθεί ως p έχει θεωρηθεί το πλήθος των πυρήνων που τρέχουν το πρόβλημα ενώ ο ορισμός για το μέγεθος προβλήματος ανά διαδικασία n διαφέρει ανάλογα με την εφαρμογή.

## 4.1 LULESH

Το LULESH αποτελεί μια ευρέως μελετημένη εφαρμογή σε προσπάθειες για σχεδιασμό νέων μηχανημάτων με exascale δυνατότητες. Η εφαρμογή μοντελοποιεί "Λαγκρανσιανές" υδροδυναμικές εξισώσεις τριών διαστάσεων σε ένα μη-δομημένο πλέγμα [23]. Ένα μεγάλο εύρος των υπολογιστικών προσομοιώσεων για επιστημονικά ή μηχανικά προβλήματα περιλαμβάνει τη μοντελοποίηση υδροδυναμικής, κάτι το οποίο δικαιολογεί και τη σημασία της μοντελοποίησης αυτής της εφαρμογής. Στη συνέχεια της ενότητας της εφαρμογής ως n έχει θεωρηθεί ο συνολικός όγκος του προβλήματος (πλέγματος), διαιρεμένος ισόποσα σε κάθε διαδικασία p.

### 4.1.1 Flops

Η πρώτη μετρική την οποία θα παρουσιάσουμε είναι ο αριθμός των πράξεων κινητής υποδιαστολής που εκτελεί η εφαρμογή, δηλαδή η ανάγκη της σε υπολογισμούς. Στο σχήμα 4.1 βλέπουμε τις παρατηρήσεις που συλλέξαμε καθώς και τα μοντέλα που παράχθηκαν για την εν

λόγω μετρική, με χρήση της πρώτης συνάρτησης αξιολόγησης 3.2, ενώ τα αντίστοιχα μοντέλα ως συναρτήσεις των $(n, p)$ παρουσιάζονται στον πίνακα 4.1.


Σχήμα 4.1: *Τα 5 καλύτερα μοντέλα πρόβλεψης του αριθμού των FLOPS για την εφαρμογή LULESH, με χρήση της 3.2*

| | Μοντέλο | Σχετικό Σφάλμα |
|---|---|---|
| Μοντέλο 1 | $6.23 \cdot 10^6 \cdot p^{0.75} \cdot n^{0.75}$ | 0.74 |
| Μοντέλο 2 | $2.98 \cdot 10^6 \cdot p^{0.5} \cdot log_2(p) \cdot n^{0.75}$ | 0.75 |
| Μοντέλο 3 | $3.44 \cdot 10^5 \cdot p^{0.75} \cdot n^{0.5} \cdot {log_2}^2(n)$ | 0.68 |
| Μοντέλο 4 | $1.65 \cdot 10^5 \cdot p^{0.5} \cdot log_2(p) \cdot n^{0.5} \cdot {log_2}^2(n)$ | 0.69 |
| Μοντέλο 5 | $3.43 \cdot 10^4 \cdot p^{0.5} \cdot {log_2}^2(p) \cdot n^{0.75} \cdot log_2(n)$ | 0.16 |

Πίνακας 4.1: *Τα 5 καλύτερα μοντέλα για τη μετρική FLOPS με score* $= R^2$

Αντίστοιχα, στο σχήμα 4.2 και στον πίνακα 4.2 μπορούμε να δούμε τα ίδια μεγέθη με χρήση της 3.3 συνάρτησης αξιολόγησης, η οποία χρησιμοποιεί τον συνδυασμό των μετρικών αξιολόγησης.


Σχήμα 4.2: *Τα 5 καλύτερα μοντέλα πρόβλεψης του αριθμού των FLOPS για την εφαρμογή LULESH, με χρήση της 3.3*

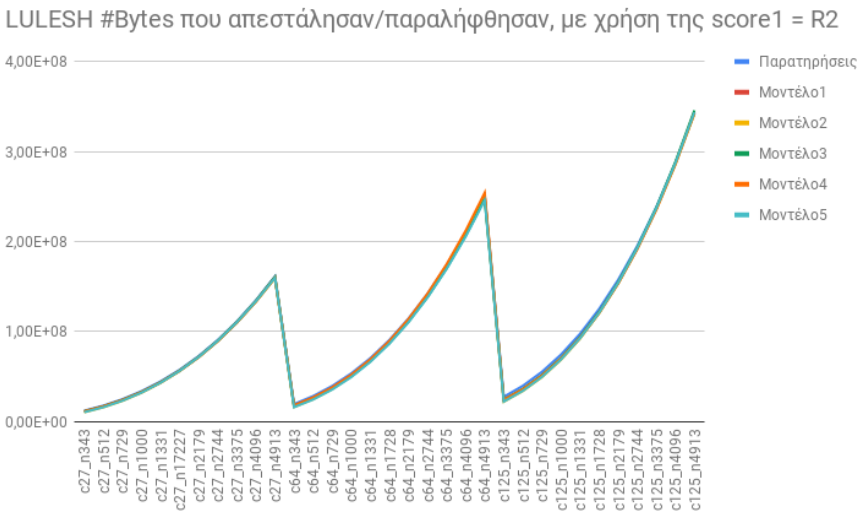
| | Μοντέλο | Σχετικό Σφάλμα |
|---|---|---|
| Μοντέλο 1 | $5.07 \cdot 10^4 \cdot p^{0.5} \cdot {log_2}^2(p) \cdot n$ | 0.01 |
| Μοντέλο 2 | $2.81 \cdot 10^3 \cdot p^{0.5} \cdot {log_2}^2(p) \cdot n^{0.75} \cdot {log_2}^2(n)$ | 0.01 |
| Μοντέλο 3 | $1.06 \cdot 10^5 \cdot p^{0.75} \cdot log_2(p) \cdot n$ | 0.005 |
| Μοντέλο 4 | $1.22 \cdot 10^4 \cdot p \cdot n^{0.75} \cdot {log_2}^2(n)$ | 0.02 |
| Μοντέλο 5 | $5.86 \cdot 10^3 \cdot p^{0.75} \cdot log_2(p) \cdot n^{0.75} \cdot {log_2}^2(n)$ | 0.02 |

Πίνακας 4.2: *Τα 5 καλύτερα μοντέλα για τη μετρική FLOPS με score* $= R^2 \cdot RCC \cdot pred_{<0.35}$

Παρατηρώντας τα γραφήματα γίνεται αντιληπτό ότι και οι δύο μέθοδοι αξιολόγησης δίνουν ικανοποιητικές προβλέψεις για το προς μελέτη μέγεθος. Ακόμη, είναι εμφανές ότι οι προβλέψεις μας αγνοούν πιθανές απρόβλεπτες συμπεριφορές στις μετρήσεις, όπως αυτή που παρατηρούμε για αριθμό πυρήνων (c) ίσο με 125 και μέγεθος προβλήματος (n) ίσο με 1728, και οι οποίες οφείλονται στα μηχανήματα της ουράς "clones". Τέλος, οι ανάγκες για υπολογισμό που περιγράφουν τα μοντέλα και των δύο πινάκων, λόγω των μικρών εκθετών στις συναρτήσεις των n,p, δεν φαίνεται να αποτελούν σπουδαίο εμπόδιο στην εφικτή εκτέλεση της εφαρμογής σε συστήματα μεγαλύτερης κλίμακας

### 4.1.2 Αριθμός Bytes που ελήφθησαν και απεστάλησαν ανά διαδικασία

Η δεύτερη μετρική που παρουσιάζουμε είναι αυτή που εκφράζει τις ανάγκες της εφαρμογής σε πράξεις επικοινωνίας, δηλαδή ο αριθμός των Bytes δεδομένων που "αντέλλαξαν" οι διαδικασίες κατά την εκτέλεση του LULESH. Στο σχήμα 4.3 και στον πίνακα 4.3 βλέπουμε τα πέντε καλύτερα μοντέλα που παράχθηκαν, σύμφωνα με την πρώτη μέθοδο αξιολόγησης 3.2.

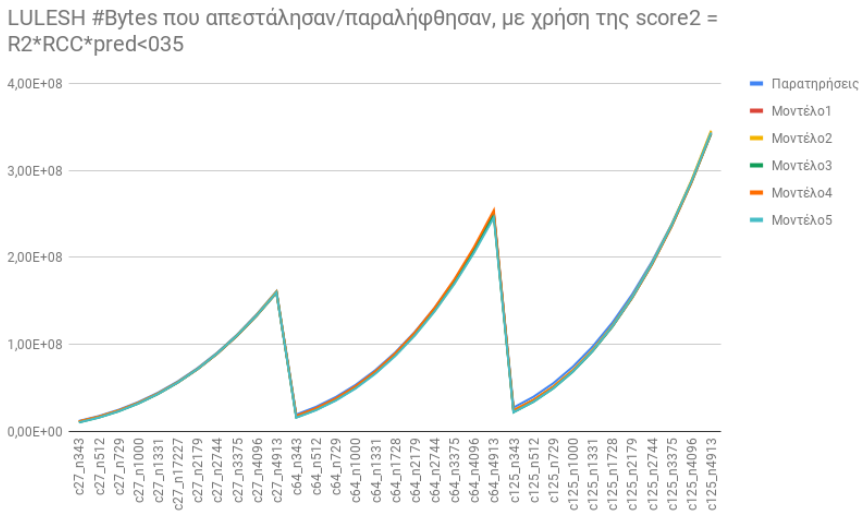



Σχήμα 4.3: *Τα 5 καλύτερα μοντέλα πρόβλεψης του αριθμού των Bytes επικοινωνίας για την εφαρμογή LULESH, με χρήση της 3.2*

| | Μοντέλο | Σχετικό Σφάλμα |
|---|---|---|
| Μοντέλο 1 | $3.01 \cdot 10^3 \cdot p^{0.25} \cdot log_2(p) \cdot n$ | 0.03 |
| Μοντέλο 2 | $1.66 \cdot 10^2 \cdot p^{0.25} \cdot log_2(p) \cdot n^{0.75} \cdot {log_2}^2(n)$ | 0.05 |
| Μοντέλο 3 | $6.3 \cdot 10^3 \cdot p^{0.5} \cdot n$ | 0.03 |
| Μοντέλο 4 | $1.44 \cdot 10^3 \cdot {log_2}^2(p) \cdot n$ | 0.03 |
| Μοντέλο 5 | $3.48 \cdot 10^2 \cdot p^{0.5} \cdot n^{0.75} \cdot {log_2}^2(n)$ | 0.05 |

Πίνακας 4.3: *Τα 5 καλύτερα μοντέλα για το πλήθος Bytes επικοινωνίας με score* $= R^2$

Αντίστοιχα, στο σχήμα 4.4 και στον πίνακα 4.4 μπορούμε να δούμε τα ίδια μεγέθη με χρήση της 3.3 συνάρτησης αξιολόγησης, η οποία χρησιμοποιεί τον συνδυασμό των μετρικών αξιολόγησης.




Σχήμα 4.4: *Τα 5 καλύτερα μοντέλα πρόβλεψης του αριθμού των Bytes επικοινωνίας για την εφαρμογή LULESH, με χρήση της 3.3*

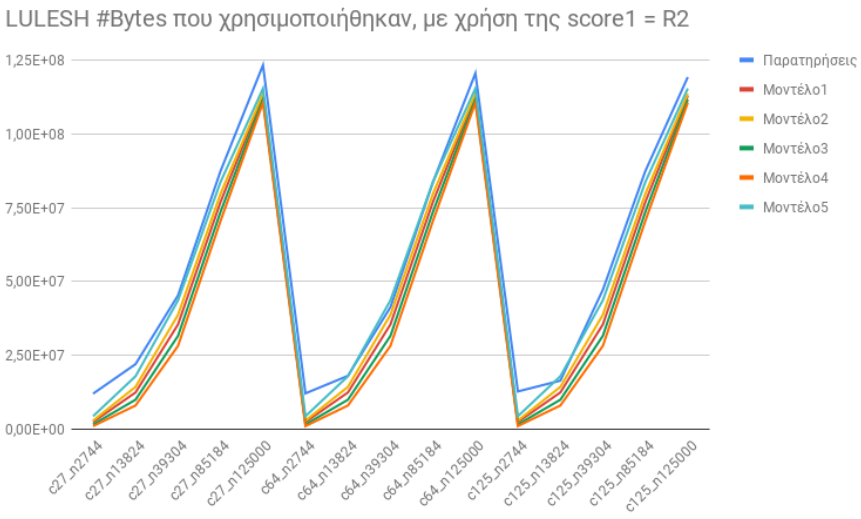
| | Μοντέλο | Σχετικό Σφάλμα |
|---|---|---|
| Μοντέλο 1 | $3.01 \cdot 10^3 \cdot p^{0.25} \cdot log_2(p) \cdot n$ | 0.03 |
| Μοντέλο 2 | $6.3 \cdot 10^3 \cdot p^{0.5} \cdot n$ | 0.03 |
| Μοντέλο 3 | $1.66 \cdot 10^2 \cdot p^{0.25} \cdot log_2(p) \cdot n^{0.75} \cdot {log_2}^2(n)$ | 0.05 |
| Μοντέλο 4 | $1.44 \cdot 10^3 \cdot {log_2}^2(p) \cdot n$ | 0.03 |
| Μοντέλο 5 | $3.48 \cdot 10^2 \cdot p^{0.5} \cdot n^{0.75} \cdot {log_2}^2(n)$ | 0.05 |

Πίνακας 4.4: *Τα 5 καλύτερα μοντέλα για το πλήθος Bytes επικοινωνίας με score* $= R^2 \cdot RCC \cdot pred_{<0.35}$

Τα παραπάνω σχήματα και πίνακες μας δείχνουν ότι και με τις δύο μεθόδους καταφέρνουμε μια ιδιαίτερα καλή πρόβλεψη για την επικοινωνία των διαδικασιών κατά την εκτέλεση του LULESH. Ιδιαίτερο ενδιαφέρον έχει επίσης το γεγονός ότι και οι δύο συναρτήσεις αξιολόγησης μας δίνουν τα ίδια πέντε μοντέλα με μία μοναδική διαφορά στην κατάταξή τους, καθώς στον πίνακα 4.4 βλέπουμε ότι το μοντέλο2 έχει αλλάξει θέση με το μοντέλο3 του πίνακα 4.3. Όπως και στα flops κι εδώ μπορούμε να αποφανθούμε πως το γεγονός ότι οι ανάγκες σε επικοινωνία μοντελοποιούνται από το γινόμενο του πλήθους διαδικασιών και του μεγέθους προβλήματος ανά διαδικασία, δεν αποτελεί ανάχωμα στην ανάγκη για πόρους επικοινωνίας όταν μεταφέρουμε την εκτέλεση της εφαρμογής σε μεγαλύτερης κλίμακας συστήματα.

### 4.1.3 Αριθμός Bytes που χρησιμοποιήθηκαν στη μνήμη

Στη συνέχεια θα παρουσιάσουμε την τρίτη μετρική, το πλήθος Bytes που χρησιμοποιήθηκαν από όλες τις διαδικασίες στη μνήμη, που περιγράφει την ανάγκη σε μνήμη της εφαρμογής. Και σε αυτή την περίπτωση παρουσιάζουμε πρώτα τα αποτελέσματα μας με βάση τη συνάρτηση αξιολόγησης 3.2 στο σχήμα 4.5 και στον πίνακα 4.5.

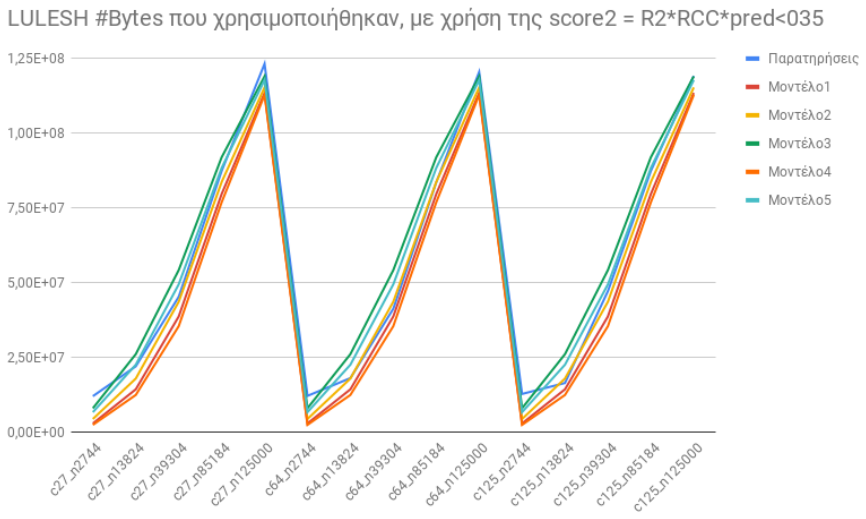



Σχήμα 4.5: *Τα 5 καλύτερα μοντέλα πρόβλεψης του αριθμού των Bytes που χρησιμοποιήθηκαν στη μνήμη για την εφαρμογή LULESH, με χρήση της 3.2*

| | Μοντέλο | Σχετικό Σφάλμα |
|---|---|---|
| Μοντέλο 1 | $9.02 \cdot 10^2 \cdot n$ | 0.3 |
| Μοντέλο 2 | $5.96 \cdot 10 \cdot n^{0.75} \cdot {log_2}^2(n)$ | 0.25 |
| Μοντέλο 3 | $5.27 \cdot 10 \cdot n \cdot log_2(n)$ | 0.36 |
| Μοντέλο 4 | $3.085 \cdot n \cdot {log_2}^2(n)$ | 0.42 |
| Μοντέλο 5 | $1.024 \cdot 10^3 \cdot n^{0.75} \cdot log_2(n)$ | 0.15 |

Πίνακας 4.5: *Τα 5 καλύτερα μοντέλα για το πλήθος Bytes που χρησιμοποιήθηκαν στη μνήμη με score* $= R^2$

Αντίστοιχα, στο σχήμα 4.6 και στον πίνακα 4.6 μπορούμε να δούμε τα ίδια μεγέθη με χρήση της 3.3 συνάρτησης αξιολόγησης, η οποία χρησιμοποιεί τον συνδυασμό των μετρικών αξιολόγησης.




Σχήμα 4.6: *Τα 5 καλύτερα μοντέλα πρόβλεψης του αριθμού των Bytes που χρησιμοποιήθηκαν στη μνήμη για την εφαρμογή LULESH, με χρήση της 3.3*

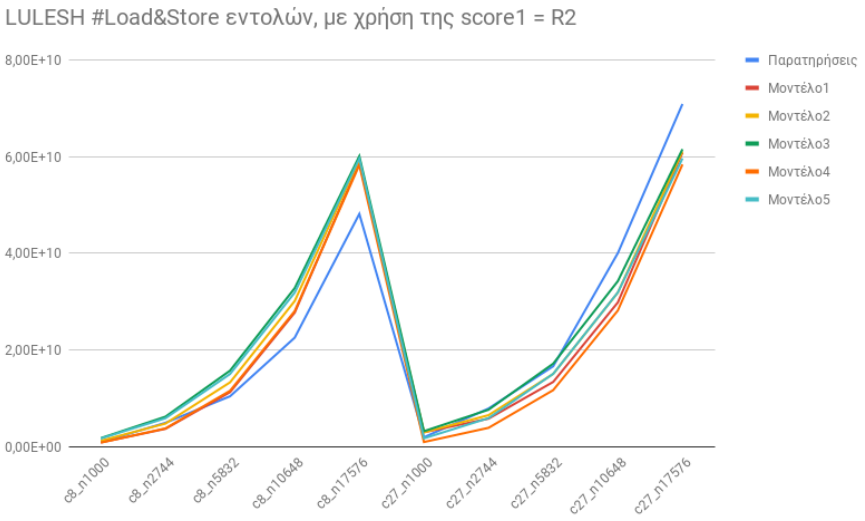
| | Μοντέλο | Σχετικό Σφάλμα |
|---|---|---|
| Μοντέλο 1 | $5.96 \cdot 10^1 \cdot n^{0.75} \cdot {log_2}^2(n)$ | 0.25 |
| Μοντέλο 2 | $1.024 \cdot 10^3 \cdot n^{0.75} \cdot log_2(n)$ | 0.15 |
| Μοντέλο 3 | $1.17 \cdot 10^3 \cdot n^{0.5} \cdot {log_2}^2(n)$ | 0.06 |
| Μοντέλο 4 | $9.02 \cdot 10^2 \cdot n$ | 0.3 |
| Μοντέλο 5 | $1.772814 \cdot 10^4 \cdot n^{0.75}$ | 0.03 |

Πίνακας 4.6: *Τα 5 καλύτερα μοντέλα για το πλήθος Bytes που χρησιμοποιήθηκαν στη μνήμη με score* $= R^2 \cdot RCC \cdot pred_{<0.35}$

Παρατηρώντας τα παραπάνω σχήματα γίνεται αντιληπτό ότι και οι δύο μέθοδοι αξιολόγησης είναι σε θέση να προβλέψουν αρκετά καλά τις ανάγκες της εφαρμογής σε μνήμη. Και σε αυτή τη μετρική παρατηρούμε ότι και με τους δύο τρόπους αξιολόγησης λαμβάνουμε, ως επί το πλείστον, κοινά μοντέλα, απλώς σε διαφορετική κατάταξη. Επίσης ενδιαφέρον έχει το γεγονός ότι τα μοντέλα για την ανάγκη του `LULESH` σε Bytes μνήμης ανά διαδικασία εξαρτώνται αποκλειστικά και μόνο από το μέγεθος του προβλήματος, κι όχι από τον αριθμό των διαδικασιών στις οποίες διαμοιράσεται το πρόβλημα. Τέλος, τα μοντέλα που παρουσιάζουμε μαρτυρούν ότι, αφού η ανάγκη σε μνήμη εξαρτάται κατά μη "απαγορευτικό" εκθετικό τρόπο από το μέγεθος του προβλήματος, η εφαρμογή θα μπορεί να τρέξει σε διαφορετικά συστήματα με σχετική άνεση, εν συναρτήσει με τις ανάγκες της σε μνήμη.

### 4.1.4 Πλήθος εντολών πρόσβασης στη μνήμη (Loads/Stores)

Ως τελευταία μετρική θα παρουσιάσουμε τον αριθμό των εντολών πρόσβασης στη μνήμη (load & store instructions) της εφαρμογής μας.

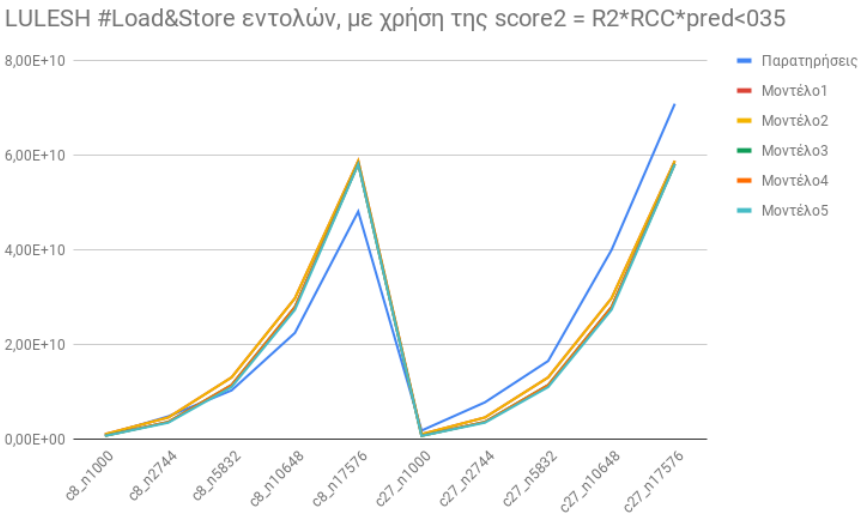



Σχήμα 4.7: *Τα 5 καλύτερα μοντέλα πρόβλεψης του αριθμού των εντολών πρόσβασης στη μνήμη για την εφαρμογή LULESH, με χρήση της 3.2*

| | Μοντέλο | Σχετικό Σφάλμα |
|---|---|---|
| Μοντέλο 1 | $2.52 \cdot 10^4 \cdot (p^{2.5} \cdot {log_2}^2(p) + n^{1.5})$ | 0.0129 |
| Μοντέλο 2 | $2.08 \cdot 10^4 \cdot (p^{2.5} \cdot {log_2}^2(p) + n^{1.25} \cdot log_2(n))$ | 0.1044 |
| Μοντέλο 3 | $1.72 \cdot 10^4 \cdot (p^{2.5} \cdot {log_2}^2(p) + n \cdot {log_2}^2(n))$ | 0.2746 |
| Μοντέλο 4 | $1.45 \cdot 10^3 \cdot (p^{2.5} \cdot {log_2}^2(p) + n^{1.25} \cdot {log_2}^2(n))$ | 0.158 |
| Μοντέλο 5 | $2.94 \cdot 10^5 \cdot (p \cdot log_2(p) + n^{1.25})$ | 0.1274 |

Πίνακας 4.7: *Τα 5 καλύτερα μοντέλα για το πλήθος των εντολών πρόσβασης στη μνήμη με* $score = R^2$

Ενώ τα αντίστοιχα μοντέλα για τη συνάρτηση αξιολόγησης 3.3 παρουσιάζονται στον πίνακα 4.8 και στο σχήμα 4.8:




Σχήμα 4.8: *Τα 5 καλύτερα μοντέλα πρόβλεψης του αριθμού των εντολών πρόσβασης στη μνήμη για την εφαρμογή LULESH, με χρήση της 3.3*

| | Μοντέλο | Σχετικό Σφάλμα |
|---|---|---|
| Μοντέλο 1 | $2.06 \cdot 10^4 \cdot (p^{0.75} \cdot log_2(p) + n^{1.25} \cdot log_2(n))$ | 0.0414 |
| Μοντέλο 2 | $2.06 \cdot 10^4 \cdot (p^{0.5} \cdot log_2(p) + n^{1.25} \cdot log_2(n))$ | 0.0414 |
| Μοντέλο 3 | $1.45 \cdot 10^3 (p^{0.75} \cdot log_2(p) + n^{1.25} \cdot {log_2}^2(n))$ | 0.1682 |
| Μοντέλο 4 | $1.45 \cdot 10^3 (p^{0.5} \cdot log_2(p) + n^{1.25} \cdot {log_2}^2(n))$ | 0.1682 |
| Μοντέλο 5 | $2.49 \cdot 10^4 (p^{0.75} \cdot log_2(p) + n^{1.5})$ | 0.1877 |

Πίνακας 4.8: *Τα 5 καλύτερα μοντέλα για το πλήθος των εντολών πρόσβασης στη μνήμη με* $score = R^2 \cdot RCC \cdot pred_{<0.35}$

Στην περίπτωση αυτής της μετρικής παρατηρούμε καλή μοντελοποίηση και από τις δύο μεθόδους. Ενδιαφέρον παρουσιάζει το γεγονός ότι έχουμε μοντέλα που εξαρτώνται από το άθροισμα συναρτήσεων των n και p. Όσον αφορά τα μοντέλα που αξιολογούνται μόνο μέσω του $R^2$, εκεί βλέπουμε υψηλούς εκθέτες για την εξάρτηση από το πλήθος των πυρήνων, κάτι που δεν μας δείχνουν τα μοντέλα που έχουν εξαχθεί με την άλλη συνάρτηση αξιολόγησης. Αντιθέτως οι συναρτήσεις του n στις οποίες έχουν καταλήξει και οι δύο μέθοδοι είναι οι ίδιες. Εάν λάβουμε υπόψιν τα μοντέλα του πίνακα 4.8 τότε μπορούμε να αναμένουμε πως η ασυμπτωτική συμπεριφορά της εφαρμογής μας σε μεγαλύτερης κλίμακας συστήματα δεν θα είναι "απαγορευτική" σε συχνότητα πρόσβασης στη μνήμη.

## 4.2 Kripke

Το `Kripke` αποτελεί έναν απλό κώδικα μοριακής μεταφοράς τριών διαστάσεων Sn. Αναπτύχθηκε έχοντας ως πρωταρχικό στόχο να ερευνήσει πώς η διάρθρωση των δεδομένων, τα προγραμματιστικά πρότυπα και οι αρχιτεκτονικές επηρεάζουν την υλοποίηση αλλά και την επίδοση ενός κώδικα Sn μεταφοράς[20]. Υλοποιεί ένα ασύγχρονα παράλληλο "sweep" αλγόριθμο, βασισμένο στο MPI. Στη συνέχεια της ενότητας του `Kripke` ως n έχει θεωρηθεί ο συνολικός όγκος του προβλήματος (πλέγματος), δια το πλήθος των διαδικασιών p. Πριν προχωρήσουμε στην ανάλυση των αναγκών, πρέπει να σημειωθεί ότι η εν λόγω εφαρμογή έχει μεγάλη επικάλυψη μεταξύ επικοινωνίας και υπολογισμού, κάτι που καθιστά δύσκολη τη μοντελοποίηση των εν λόγω δύο μετρικών, όπως θα γίνει αντιληπτό και στη συνέχεια.

### 4.2.1 Flops

Στο σχήμα 4.9 και τον πίνακα 4.9 μπορούμε να δούμε τα μοντέλα που εξήχθησαν για τη μετρική Flops με χρήση της 3.2.


Σχήμα 4.9: *Τα 5 καλύτερα μοντέλα πρόβλεψης του αριθμού των flops για την εφαρμογή Kripke, με χρήση της 3.2*

| | Μοντέλο | Σχετικό Σφάλμα |
|---|---|---|
| Μοντέλο 1 | $6.16 \cdot 10^5 \cdot n^{0.75} \cdot log_2(n)$ | 0.0179 |
| Μοντέλο 2 | $7.95 \cdot 10^6 \cdot n^{0.75}$ | 0.1972 |
| Μοντέλο 3 | $4.35 \cdot 10^5 \cdot n^{0.5} \cdot log_2{}^2(n)$ | 0.1747 |
| Μοντέλο 4 | $4.88 \cdot 10^4 \cdot n^{0.75} \cdot log_2{}^2(n)$ | 0.1144 |
| Μοντέλο 5 | $8.86 \cdot 10^5 \cdot n$ | 0.1029 |

Πίνακας 4.9: *Τα 5 καλύτερα μοντέλα για το πλήθος των flops με score* $= R^2$

Αντίστοιχα τα ίδια μοντέλα με χρήση της συνδυαστικής συνάρτησης αξιολόγησης 3.3 παρουσιάζονται στη συνέχεια:


Σχήμα 4.10: *Τα 5 καλύτερα μοντέλα πρόβλεψης του αριθμού των flops για την εφαρμογή Kripke, με χρήση της 3.3*

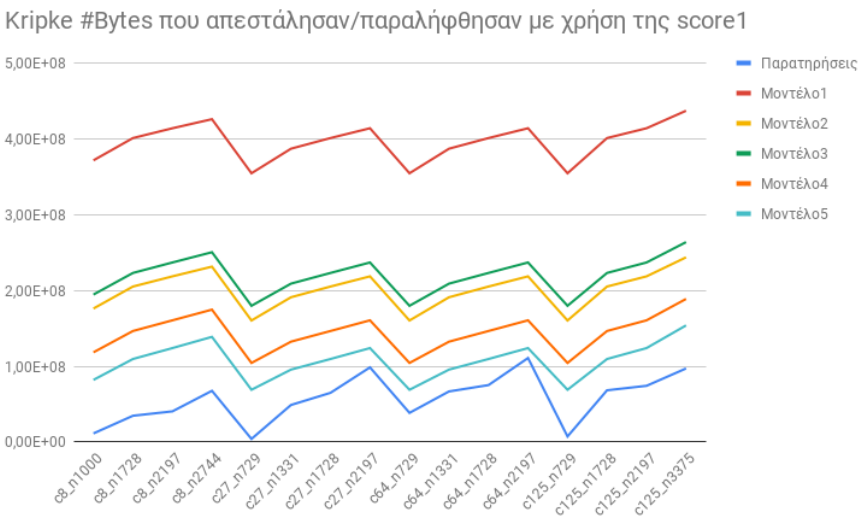
| | Μοντέλο | Σχετικό Σφάλμα |
|---|---|---|
| Μοντέλο 1 | $6.16 \cdot 10^5 \cdot n^{0.75} \cdot log_2(n)$ | 0.0179 |
| Μοντέλο 2 | $4.88 \cdot 10^4 \cdot n^{0.75} \cdot {log_2}^2(n)$ | 0.1144 |
| Μοντέλο 3 | $8.86 \cdot 10^5 \cdot n$ | 0.1029 |
| Μοντέλο 4 | $7.1 \cdot 10^4 \cdot n \cdot log_2(n)$ | 0.2075 |
| Μοντέλο 5 | $5.75 \cdot 10^3 \cdot n \cdot {log_2}^2(n)$ | 0.2901 |

Πίνακας 4.10: *Τα 5 καλύτερα μοντέλα για τον αριθμό των flops με score* $= R^2 \cdot RCC \cdot pred_{<0.35}$

Προτού σχολιάσουμε την απόδοση των μοντέλων μας, είναι σημαντικό να επισημάνουμε το γεγονός ότι οι παρατηρήσεις διαθέτουν μια, κατά τόπους, απρόβλεπτη συμπεριφορά, όπως π.χ. παρατηρούμε από τις κορυφές στα σημεία c27_n1000, c125_n1728. Αυτό οφείλεται μάλλον στα μηχανήματα της ουράς clones αλλά και στην προαναφερθείσα επικάλυψη μεταξύ υπολογισμού κι επικοινωνίας στην εφαρμογή. Παρά αυτή τη μη ομαλή συμπεριφορά των παρατηρήσεων, οι προβλέψεις των μοντέλων μας μοιάζουν να αποτυπώνουν τη γενική τάση των μετρήσεων. Αξίζει να σημειωθεί ότι τρία από τα πέντε μοντέλα καθενός εκ των πινάκων 4.9 και 4.10 είναι κοινά, ενώ όλα εξαρτώνται μόνο από το μέγεθος του προβλήματος ανά διαδικασία n. Μάλιστα η μη εμφάνιση του n με υψηλούς εκθέτες οδηγεί στο συμπέρασμα ότι η εφαρμογή αναμένεται να "κλιμακώνει" σχετικά ομαλά σε συστήματα μεγαλύτερης κλίμακας.

### 4.2.2 Αριθμός Bytes που ελήφθησαν και απεστάλησαν ανά διαδικασία

Στη συνέχεια θα παρουσιάσουμε τα μοντέλα μας για τη δεύτερη μετρική που περιγράφει τις ανάγκες της εφαρμογής σε επικοινωνία, ξεκινώντας και πάλι με τα αποτελέσματά μας με χρήση της συνάρτησης 3.2.

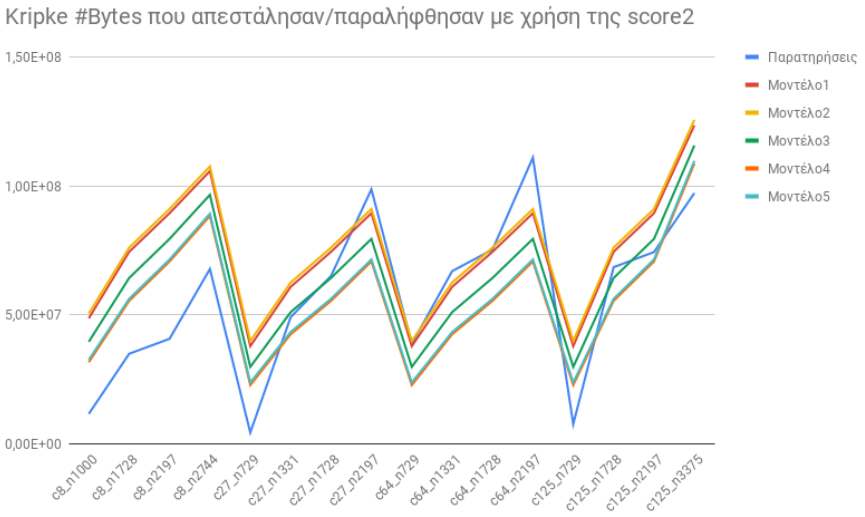



Σχήμα 4.11: *Τα 5 καλύτερα μοντέλα πρόβλεψης του αριθμού των Bytes που απεστάλησαν και ελήφθησαν για την εφαρμογή Kripke, με χρήση της 3.2*

| | Μοντέλο | Σχετικό Σφάλμα |
|---|---|---|
| Μοντέλο 1 | $3.73 \cdot 10^7 \cdot log_2(n)$ | 14.54 |
| Μοντέλο 2 | $1.77 \cdot 10^6 \cdot {log_2}^2(n)$ | 6.3953 |
| Μοντέλο 3 | $3.46 \cdot 10^7 \cdot n^{0.25}$ | 7.1879 |
| Μοντέλο 4 | $2.11 \cdot 10^6 \cdot n^{0.25} \cdot log_2(n)$ | 4.0188 |
| Μοντέλο 5 | $1.47 \cdot 10^5 \cdot n^{0.25} \cdot {log_2}^2(n)$ | 2.519 |

Πίνακας 4.11: *Τα 5 καλύτερα μοντέλα για τον αριθμό των Bytes που απεστάλησαν κι ελήφθησαν με score =* $R^2$

Ενώ τα αντίστοιχα μοντέλα με χρήση της 3.3 είναι:




Σχήμα 4.12: *Τα 5 καλύτερα μοντέλα πρόβλεψης του αριθμού των Bytes που απεστάλησαν και ελήφθησαν για την εφαρμογή Kripke, με χρήση της 3.3*

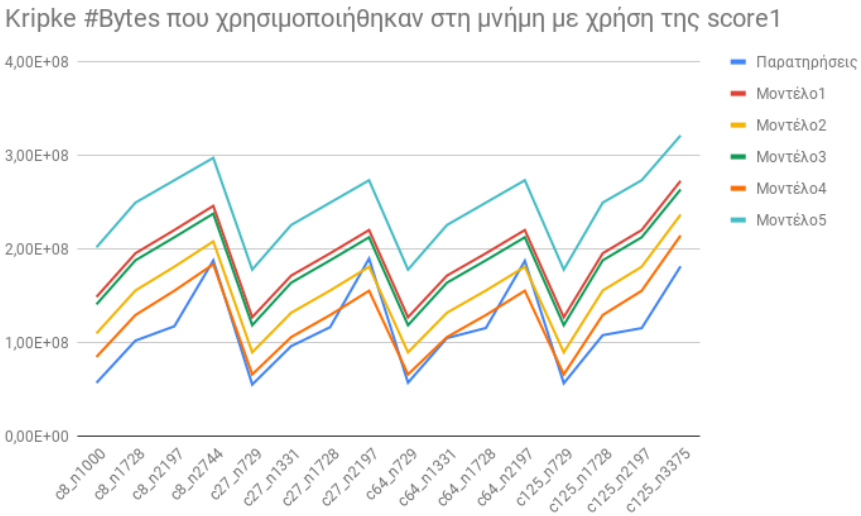
| | Μοντέλο | Σχετικό Σφάλμα |
|---|---|---|
| Μοντέλο 1 | $1.55 \cdot 10^4 \cdot n^{0.5} \cdot log_2{}^2(n)$ | 1.1578 |
| Μοντέλο 2 | $2.84 \cdot 10^5 \cdot n^{0.75}$ | 1.2372 |
| Μοντέλο 3 | $2.23 \cdot 10^4 \cdot n^{0.75} \cdot log_2(n)$ | 0.7943 |
| Μοντέλο 4 | $1.78 \cdot 10^3 \cdot n^{0.75} \cdot log_2{}^2(n)$ | 0.4747 |
| Μοντέλο 5 | $3.25 \cdot 10^4 \cdot n$ | 0.5155 |

Πίνακας 4.12: *Τα 5 καλύτερα μοντέλα για τον αριθμό των Bytes που απεστάλησαν κι ελήφθησαν με* $score = R^2 \cdot RCC \cdot pred_{<0.35}$

Οι ανάγκες του `Kripke` σε επικοινωνία είναι μια πρώτη ευκαιρία να παρατηρήσουμε τις διαφορές στην επιλογή μοντέλων των δύο συναρτήσεων αξιολόγησης. Συγκεκριμένα παρατηρούμε ότι αξιολογώντας τα μοντέλα μόνο βάση με το $R^2$ λαμβάνουμε προβλέψεις με αρκετά μεγαλύτερο σφάλμα συγκριτικά με τις μοντελοποιήσεις της συνδυαστικής αξιολόγησης. Αυτό φυσικά οφείλεται στο ότι η δεύτερη συνάρτηση αξιολόγησης λαμβάνει υπόψιν το σχετικό σφάλμα των προβλέψεων, "τιμωρώντας" έτσι προβλέψεις που μπορεί να έχουν υψηλή τιμή $R^2$, όπως αυτές που παρατηρούμε στο σχήμα 4.11, ωστόσο δεν δίνουν μικρό σχετικό σφάλμα, με αποτέλεσμα τις προφανείς αποκλίσεις του εν λόγω σχήματος συγκριτικά με τις παρατηρήσεις. Αντιθέτως, στο σχήμα 4.12, παρατηρούμε ότι έχουν "επιβραβευθεί" ως καλύτερες οι προβλέψεις οι οποίες έχουν και κατάλληλη προσαρμογή στη μορφή των παρατηρήσεων και δίνουν μικρό σφάλμα πρόβλεψης. Αξίζει να σημειώσουμε πως, και για το κομμάτι της επικοινωνίας του `Kripke`, παρατηρούμε εξάρτηση των αναγκών μόνο από το μέγεθος του προβλήματος ανά διαδικασία, χωρίς μάλιστα να έχουμε υψηλούς εκθέτες, κάτι που μας οδηγεί στο συμπέρασμα ότι η εφαρμογή θα συμπεριφέρεται ικανοποιητικά και σε μεγαλύτερου μεγέθους προβλήματα σε συστήματα μεγάλης κλίμακας. Τέλος, παρατηρούμε ότι και στο κομμάτι της επικοινωνίας, οι μετρήσεις που λαμβάνουμε από τα μηχανήματα φαίνεται να έχουν τυχαίες κορυφές σε απρόσμενα σημεία, κάτι που, όπως προ-αναφέρθηκε, οφείλεται και στην επικάλυψη μεταξύ επικοινωνίας και υπολογισμού στο `Kripke`.

### 4.2.3 Αριθμός Bytes που χρησιμοποιήθηκαν στη μνήμη

Ως τρίτη παρουσιάζουμε και για αυτή την εφαρμογή την μετρική που εκφράζει τις ανάγκες της σε μνήμη.

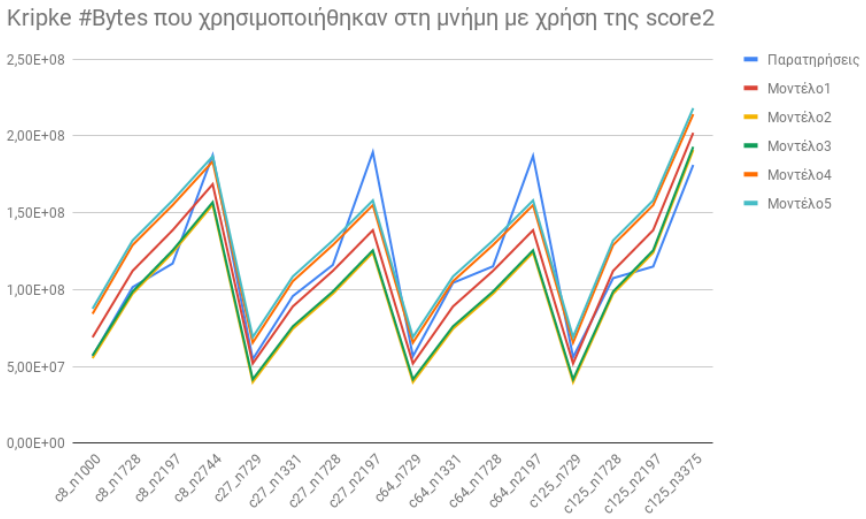



Σχήμα 4.13: *Τα 5 καλύτερα μοντέλα πρόβλεψης του αριθμού των Bytes που χρησιμοποιήθηκαν στη μνήμη για την εφαρμογή Kripke, με χρήση της 3.2*

| | Μοντέλο | Σχετικό Σφάλμα |
|---|---|---|
| Μοντέλο 1 | $4.69 \cdot 10^6 \cdot n^{0.5}$ | 0.8071 |
| Μοντέλο 2 | $3.47 \cdot 10^5 \cdot n^{0.5} \cdot log_2(n)$ | 0.4042 |
| Μοντέλο 3 | $2.52 \cdot 10^5 \cdot n^{0.25} \cdot log_2{}^2(n)$ | 0.7237 |
| Μοντέλο 4 | $2.68 \cdot 10^4 \cdot n^{0.5} \cdot log_2{}^2(n)$ | 0.1441 |
| Μοντέλο 5 | $3.6 \cdot 10^6 \cdot n^{0.25} \cdot log_2(n)$ | 1.3546 |

Πίνακας 4.13: *Τα 5 καλύτερα μοντέλα για τον αριθμό των Bytes που χρησιμοποιήθηκαν στη μνήμη με score =* $R^2$

Αντίστοιχα παρουσιάζουμε τα μοντέλα από τη συνδυαστική συνάρτηση αξιολόγησης στον πίνακα 4.14 και στο σχήμα 4.14.




Σχήμα 4.14: *Τα 5 καλύτερα μοντέλα πρόβλεψης του αριθμού των Bytes που χρησιμοποιήθηκαν στη μνήμη για την εφαρμογή Kripke, με χρήση της 3.3*

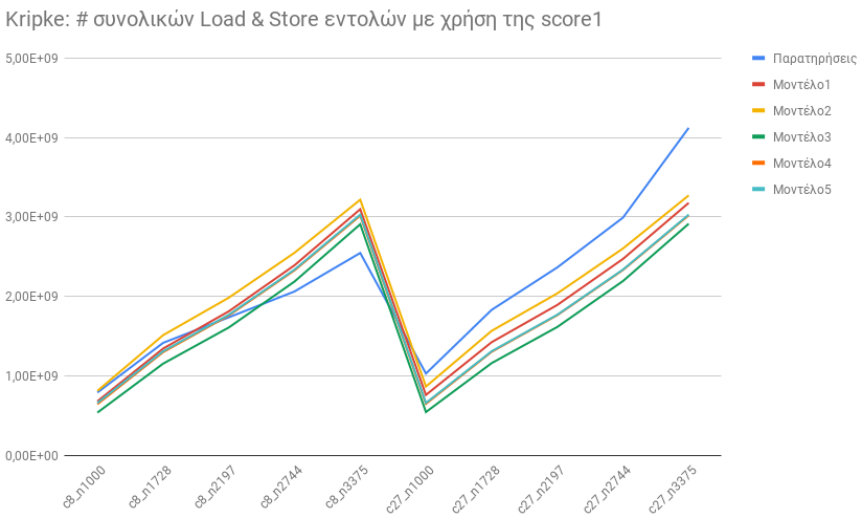
| | Μοντέλο | Σχετικό Σφάλμα |
|---|---|---|
| Μοντέλο 1 | $3.89 \cdot 10^4 \cdot n^{0.75} \cdot log_2(n)$ | 0.0147 |
| Μοντέλο 2 | $3.14 \cdot 10^3 \cdot n^{0.75} \cdot {log_2}^2(n)$ | 0.1546 |
| Μοντέλο 3 | $5.72 \cdot 10^4 \cdot n$ | 0.139 |
| Μοντέλο 4 | $2.68 \cdot 10^4 \cdot n^{0.5} \cdot {log_2}^2(n)$ | 0.1441 |
| Μοντέλο 5 | $4.93 \cdot 10^5 \cdot n^{0.75}$ | 0.1783 |

Πίνακας 4.14: *Τα 5 καλύτερα μοντέλα για τον αριθμό των Bytes που χρησιμοποιήθηκαν στη μνήμη με* $score = R^2 \cdot RCC \cdot pred_{<0.35}$

Όσον αφορά την ανάγκη της εφαρμογής σε μνήμη και σε αυτή την περίπτωση παρατηρούμε εξάρτηση του αριθμού Bytes που χρησιμοποιήθηκαν μόνο από το μέγεθος του προβλήματος ανά διαδικασία n. Ομοίως με τις προηγούμενες μετρικές, λόγω των μικρών εκθετών της παραμέτρου στα μοντέλα, αναμένουμε καλή "κλιμάκωση" της εφαρμογής σε μεγαλύτερα προβλήματα με περισσότερες διαδικασίες, δηλαδή σε μεγαλύτερα συστήματα. Σχετικά με τις διαφορές των δύο μεθόδων αξιολόγησης παρατηρούμε καλύτερη επίδοση των μοντέλων του σχήματος 4.14, αφού εκεί παρατηρούμε μικρότερο σφάλμα συγκριτικά με τις παρατηρήσεις. Στο σχήμα 4.13 μπορούμε να δούμε ότι το "Μοντέλο1" έχει αξιολογηθεί ως το καλύτερο, με το υψηλότερο $R^2$, ωστόσο τα μοντέλα 2,4 φαίνεται να έχουν μικρότερο σχετικό σφάλμα, δείγμα της αδυναμίας της πρώτης μεθόδου αξιολόγησης συγκριτικά με τη δεύτερη.

### 4.2.4 Πλήθος εντολών πρόσβασης στη μνήμη (Loads/Stores)

Τέλος παρουσιάζουμε τις μετρήσεις και τις προβλέψεις μας για το πλήθος των εντολών πρόσβασης στη μνήμη που εκτελεί η εφαρμογή.

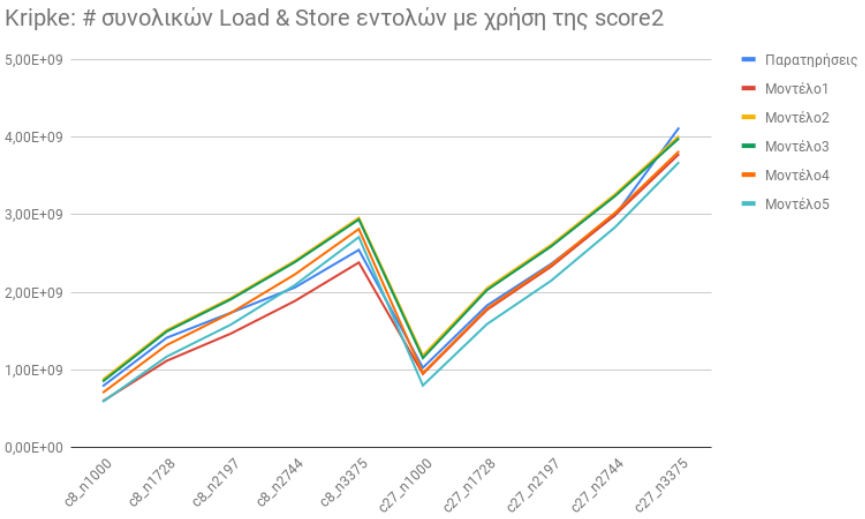



Σχήμα 4.15: *Τα 5 καλύτερα μοντέλα πρόβλεψης του αριθμού των Loads & Stores που εκτελέστηκαν για την εφαρμογή Kripke, με χρήση της 3.2*

| | Μοντέλο | Σχετικό Σφάλμα |
|---|---|---|
| Μοντέλο 1 | $1.20 \cdot 10^5 \cdot (p^2 + n^{1.25})$ | 0.0855 |
| Μοντέλο 2 | $8.12 \cdot 10^4 \cdot (p^2 + n \cdot log_2(n)$ | 0.0041 |
| Μοντέλο 3 | $9.65 \cdot 10^3 \cdot (p^2 + n^{1.25} \cdot log_2(n))$ | 0.2084 |
| Μοντέλο 4 | $6.51 \cdot 10^3 \cdot (p^2 + n \cdot {log_2}^2(n))$ | 0.1339 |
| Μοντέλο 5 | $1.18 \cdot 10^5 \cdot n^{1.25}$ | 0.1247 |

Πίνακας 4.15: *Τα 5 καλύτερα μοντέλα για τον αριθμό των Loads & Stores που εκτελέστηκαν με score* $= R^2$

Αντίστοιχα παρουσιάζουμε τα μοντέλα από τη συνδυαστική συνάρτηση αξιολόγησης στον πίνακα 4.16 και στο σχήμα 4.16.




Σχήμα 4.16: *Τα 5 καλύτερα μοντέλα πρόβλεψης του αριθμού των Loads & Stores που εκτελέστηκαν για την εφαρμογή Kripke, με χρήση της 3.3*

| | Μοντέλο | Σχετικό Σφάλμα |
|---|---|---|
| Μοντέλο 1 | $2.01 \cdot 10^4 \cdot log_2(p) \cdot n \cdot log_2(n)$ | 0.3536 |
| Μοντέλο 2 | $5.22 \cdot 10^5 \cdot p^{0.25} \cdot n$ | 0.2178 |
| Μοντέλο 3 | $2.87 \cdot 10^3 \cdot p^{0.25} \cdot n^{0.75} \cdot {log_2}^2(n)$ | 0.2274 |
| Μοντέλο 4 | $4.23 \cdot 10^4 \cdot p^{0.25} \cdot n \cdot log_2(n)$ | 0.3071 |
| Μοντέλο 5 | $6.27 \cdot 10^4 \cdot p^{0.25} \cdot n^{1.25}$ | 0.3643 |

Πίνακας 4.16: *Τα 5 καλύτερα μοντέλα για τον αριθμό των Loads & Stores που εκτελέστηκαν με score* $= R^2 \cdot RCC \cdot pred_{<0.35}$

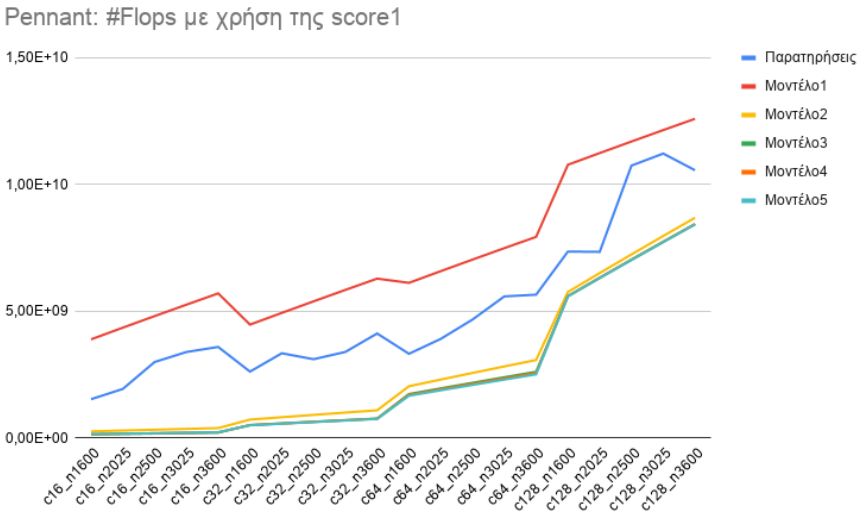
Σε αυτή τη μετρική παρατηρούμε ικανοποιητικά αποτελέσματα κι από τις δύο συναρτήσεις αξιολόγησης. Σχετικά με τα μοντέλα που παρουσιάζονται στους πίνακες 4.15 και 4.16 παρατηρούμε ότι αυτή είναι η μόνη μετρική του `Kripke` η οποία εξαρτάται κι από τις δύο παραμέτρους: n,p. Το γεγονός ότι η ανάγκη της εφαρμογής σε εντολές πρόσβασης στη μνήμη εξαρτάται από το γινόμενο, 4.16, ή το άθροισμα, 4.15, των δύο παραμέτρων μπορεί να οδηγήσει σε καθυστέρηση της εκτέλεσης της εφαρμογής, όσο αυτοί οι αριθμοί λαμβάνουν μεγαλύτερες τιμές.

## 4.3 Pennant

Το Pennant αποτελεί μια εφαρμογή που λειτουργεί σε γενικά μη δομημένα, δηλαδή με τυχαία πολύγωνα, πλέγματα δύο διαστάσεων και σχεδιάσθηκε για προχωρημένη έρευνα σε θέματα αρχιτεκτονικής πολυπύρηνων συστημάτων[21]. Στις υποενότητες που ακολουθούν ως μέγεθος του προβλήματος ανά διαδικασία n, έχει θεωρηθεί ο συνολικός όγκος του πλέγματος δια το πλήθος των διαδικασιών p.

### 4.3.1 Flops

Στο σχήμα 4.17 και τον πίνακα 4.17 μπορούμε να δούμε τις μετρήσεις αλλά και τις προβλέψεις για το πλήθος των πράξεων κινητής υποδιαστολής flops που εκτελεί η εφαρμογή.

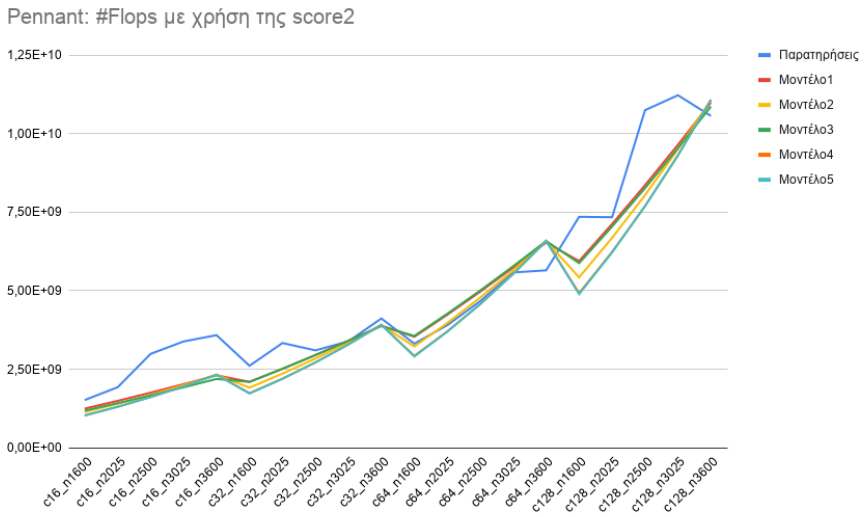



Σχήμα 4.17: *Τα 5 καλύτερα μοντέλα πρόβλεψης του αριθμού των flops που εκτελέστηκαν για την εφαρμογή Pennant, με χρήση της 3.2*

| | Μοντέλο | Σχετικό Σφάλμα |
|---|---|---|
| Μοντέλο 1 | $4.98 \cdot 10^6 \cdot (p^{1.5} + n^{0.25} \cdot log_2{}^2(n))$ | 0.5933 |
| Μοντέλο 2 | $5.55 \cdot 10^3 \cdot p^{1.5} \cdot n^{0.25} \cdot log_2{}^2(n)$ | 0.5662 |
| Μοντέλο 3 | $3.7 \cdot 10^2 \cdot p^1.25 \cdot log_2{}^2(p) \cdot n^{0.25} \cdot log_2{}^2(n)$ | 0.6282 |
| Μοντέλο 4 | $7.7 \cdot 10^2 \cdot (n^{0.25}) \cdot log_2{}^2(n) \cdot p^{1.5} \cdot log_2(p)$ | 0.6303 |
| Μοντέλο 5 | $1.6 \cdot 10^3 \cdot p^{1.75} \cdot n^{0.25} \cdot log_2{}^2(n)$ | 0.6328 |

Πίνακας 4.17: *Τα 5 καλύτερα μοντέλα για τον αριθμό των flops που εκτελέστηκαν με score* $= R^2$

Αντίστοιχα παρουσιάζουμε τα μοντέλα από τη συνδυαστική συνάρτηση αξιολόγησης στον πίνακα 4.18 και στο σχήμα 4.18.




Σχήμα 4.18: *Τα 5 καλύτερα μοντέλα πρόβλεψης του αριθμού των flops που εκτελέστηκαν για την εφαρμογή Pennant, με χρήση της 3.3*

| | Μοντέλο | Σχετικό Σφάλμα |
|---|---|---|
| Μοντέλο 1 | $3.45 \cdot 10^4 \cdot p^{0.75} \cdot n^{0.5} \cdot log_2{}^2(n)$ | 0.1127 |
| Μοντέλο 2 | $5.3 \cdot 10^4 \cdot p^{0.75} \cdot n^{0.75} \cdot log_2(n)$ | 0.1467 |
| Μοντέλο 3 | $1.64 \cdot 10^4 \cdot p^{0.5} \cdot log_2(p) \cdot n^{0.5} \cdot log_2{}^2(n)$ | 0.1221 |
| Μοντέλο 4 | $8.09 \cdot 10^4 \cdot p^{0.75} \cdot n$ | 0.1831 |
| Μοντέλο 5 | $4.48 \cdot 10^3 \cdot p^{0.75} \cdot n^{0.75} \cdot log_2{}^2(n)$ | 0.1848 |

Πίνακας 4.18: *Τα 5 καλύτερα μοντέλα για τον αριθμό των flops που εκτελέστηκαν με* $score = R^2 \cdot RCC \cdot pred_{<0.35}$

Παρατηρούμε ότι τα μοντέλα προβλέψεων για τα flops του `Pennant` ακολουθούν σε ικανοποιητικό βαθμό τις μετρήσεις μας, ειδικά αυτά του σχήματος 4.18. Και στις δύο περιπτώσεις παρατηρούμε εξάρτηση των μοντέλων κι από τις δύο παραμέτρους, στον πίνακα 4.17 το καλύτερο μοντέλο εξαρτάται από το άθροισμα ενώ στον πίνακα 4.18 όλα εξαρτώνται από το γινόμενό τους. Το γεγονός αυτό μας οδηγεί στο συμπέρασμα ότι η εφαρμογή μπορεί να οδηγηθεί σε υψηλές υπολογιστικές ανάγκες, όσο οι δύο αυτοί παράμετροι αυξάνονται.

Τα μοντέλα του πίνακα 4.18 είναι μια καλή ευκαιρία για περαιτέρω σχολιασμό της συνδυαστικής συνάρτησης αξιολόγησης. Παρακάτω μπορούμε να δούμε το βαθμό με τον οποίο αξιολογήθηκε κάθε μοντέλο, καθώς και τις επιμέρους τιμές των τριών μετρικών $R^2$, $RCC$, $pred_{<0.35}$:

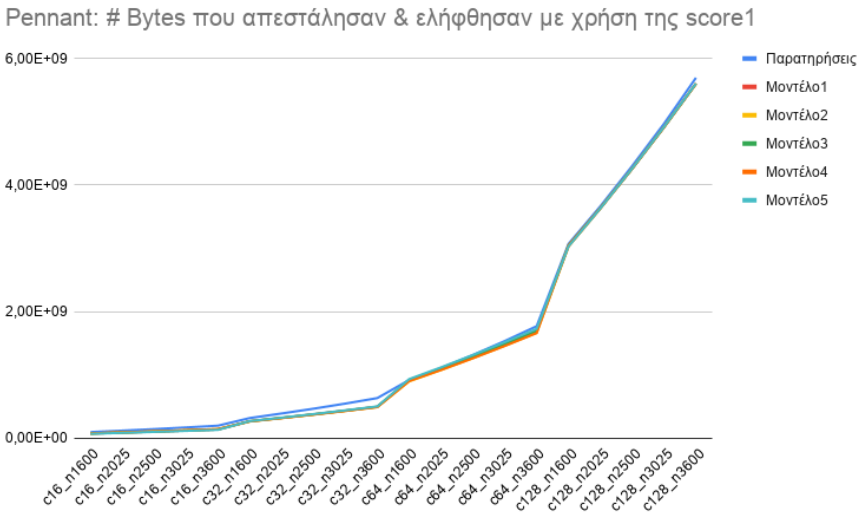
| Μοντέλο | score | $R^2$ | $RCC$ | $pred_{<0.35}$ |
|---|---|---|---|---|
| Μοντέλο 1 | 0,7119 | 0,9198 | 0,91053 | 0,85 |
| Μοντέλο 2 | 0,7073 | 0,9086 | 0,91579 | 0,85 |
| Μοντέλο 3 | 0,7056 | 0,9117 | 0,91053 | 0,85 |
| Μοντέλο 4 | 0,6989 | 0,8927 | 0,92105 | 0,85 |
| Μοντέλο 5 | 0,69854 | 0,8922 | 0,92105 | 0,85 |

Πίνακας 4.19: *Αναλυτικά οι τιμές των επιμέρους μετρικών για τα μοντέλα του 4.18*

Το πρώτο πράγμα που μπορούμε να σχολιάσουμε από τον παραπάνω πίνακα είναι το γεγονός ότι αν κατατάσσαμε αυτά τα μοντέλα με χρήση της $score_1$ τότε το "Μοντέλο3" θα άλλαζε θέση με το "Μοντέλο2", κάτι που τώρα δεν συμβαίνει μιας και το 3 έχει χαμηλότερη τιμή RCC σε σχέση με το 2. Επίσης, αν η αξιολόγηση των μοντέλων γινόταν με βάση μόνο το RCC τότε τα δύο χειρότερα μοντέλα (4,5) θα ήταν οι "νικητές" ασχέτως αν έχουν τα χειρότερα $R^2$ εκ των πέντε.

### 4.3.2 Αριθμός Bytes που ελήφθησαν και απεστάλησαν ανά διαδικασία

Στη συνέχεια θα παρουσιάσουμε τα μοντέλα μας για τη δεύτερη μετρική που περιγράφει τις ανάγκες της εφαρμογής σε επικοινωνία, ξεκινώντας και πάλι με τα αποτελέσματά μας με χρήση της συνάρτησης 3.2.




Σχήμα 4.19: *Τα 5 καλύτερα μοντέλα πρόβλεψης του αριθμού των Bytes που απεστάλησαν και ελήφθησαν για την εφαρμογή Pennant, με χρήση της 3.2*

| | Μοντέλο | Σχετικό Σφάλμα |
|---|---|---|
| Μοντέλο 1 | $6.6 \cdot 10^1 \cdot n^{0.5} \cdot log_2{}^2(n) \cdot p^{1.5} \cdot log_2(p)$ | 0.1225 |
| Μοντέλο 2 | $1.37 \cdot 10^2 \cdot n^{0.5} \cdot log_2{}^2(n) \cdot p^{1.75}$ | 0.1219 |
| Μοντέλο 3 | $1.189 \cdot 10^3 \cdot n^{0.75} \cdot p^{1.5} \cdot log_2(p)$ | 0.1215 |
| Μοντέλο 4 | $2.474 \cdot 10^3 \cdot n^{0.75} \cdot p^{1.75}$ | 0.121 |
| Μοντέλο 5 | $3.2 \cdot 10^1 \cdot n^{0.5} \cdot log_2{}^2(n) \cdot p^{1.25} \cdot log_2{}^2(p)$ | 0.1227 |

Πίνακας 4.20: *Τα 5 καλύτερα μοντέλα για τον αριθμό των Bytes που απεστάλησαν κι ελήφθησαν με score* $= R^2$

Ενώ τα αντίστοιχα μοντέλα με χρήση της 3.3 είναι:


Σχήμα 4.20: *Τα 5 καλύτερα μοντέλα πρόβλεψης του αριθμού των Bytes που απεστάλησαν και ελήφθησαν για την εφαρμογή Pennant, με χρήση της 3.3*

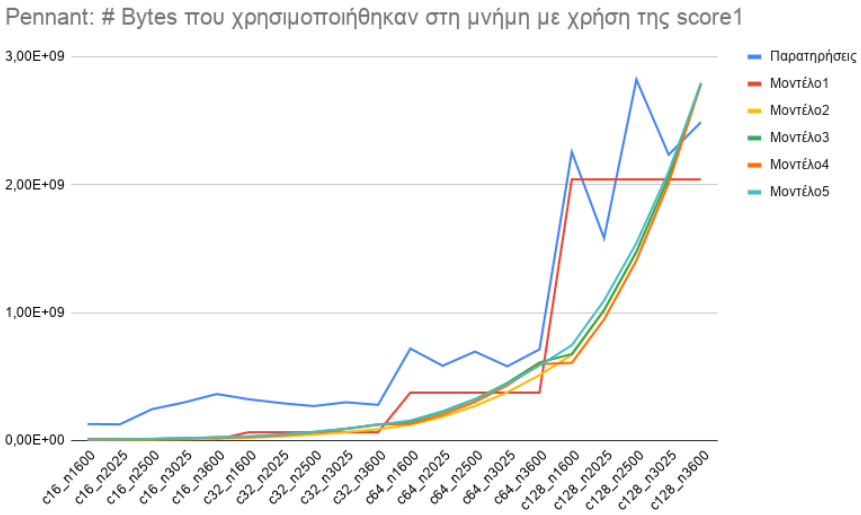
| | Μοντέλο | Σχετικό Σφάλμα |
|---|---|---|
| Μοντέλο 1 | $6.6 \cdot 10^1 \cdot n^{0.5} \cdot log_2{}^2(n) \cdot p^{1.5} \cdot log_2(p)$ | 0.1225 |
| Μοντέλο 2 | $1.37 \cdot 10^2 \cdot n^{0.5} \cdot log_2{}^2(n) \cdot p^{1.75}$ | 0.1219 |
| Μοντέλο 3 | $1.189 \cdot 10^3 \cdot n^{0.75} \cdot p^{1.5} \cdot log_2(p)$ | 0.1215 |
| Μοντέλο 4 | $2.474 \cdot 10^3 \cdot n^{0.75} \cdot p^{1.75}$ | 0.121 |
| Μοντέλο 5 | $3.2 \cdot 10^1 \cdot n^{0.5} \cdot log_2{}^2(n) \cdot p^{1.25} \cdot log_2{}^2(p)$ | 0.1227 |

Πίνακας 4.21: *Τα 5 καλύτερα μοντέλα για τον αριθμό των Bytes που απεστάλησαν κι ελήφθησαν με score* $= R^2 \cdot RCC \cdot pred_{<0.35}$

Γίνεται εμφανές από τα παραπάνω αποτελέσματα ότι οι προβλέψεις που κάνουν τα μοντέλα μας σχετικά με τις ανάγκες του `Pennant` για επικοινωνία μεταξύ των διαδικασιών είναι αρκετά ακριβή, μιας και η καμπύλη των παρατηρήσεων έχει εύκολα προβλέψιμη μορφή. Παρατηρούμε ακόμη ότι οι δύο τρόποι αξιολόγησης δίνουν ακριβώς τα ίδια πέντε μοντέλα. Αυτό οφείλεται στο γεγονός ότι όλα τα μοντέλα του 4.21 έχουν $RCC = 1$ και $pred_{<0.35} = 1$, με αποτέλεσμα η κατάταξη τους να εκφυλίζεται σε ταξινόμηση με βάση το $R^2$ που παρατηρείται. Η ακρίβεια μπορεί επίσης να δικαιολογηθεί από το γεγονός ότι στην υλοποίηση της εφαρμογής χρησιμοποιούνται αποκλειστικά "συγκεντρωτικές" συναρτήσεις επικοινωνίας του MPI, όπως `MPI_Allreduce`, `MPI_Gather` κ.ά., κάτι το οποίο οδηγεί σε εύκολο μοντελοποιήσιμο καθεστώς επικοινωνίας. Στο παραπάνω οφείλεται και η συνεχής και σταθερή αύξηση της μετρικής με την είσοδο περισσότερων διαδικασιών, κάτι που αποτυπώνεται και στους υψηλούς εκθέτες του p στα μοντέλα. Αυτοί είναι και ο λόγος για τον οποίο πιστεύουμε ότι η εφαρμογή μπορεί να οδηγηθεί σε σημαντικές καθυστερήσεις σε συστήματα μεγαλύτερης κλίμακας, κάτι που γίνεται εύκολα αντιληπτό κι από τις γραφικές 4.19 και 4.20.

### 4.3.3 Αριθμός Bytes που χρησιμοποιήθηκαν στη μνήμη

Ως τρίτη παρουσιάζουμε και για αυτή την εφαρμογή την μετρική που εκφράζει τις ανάγκες της σε μνήμη.

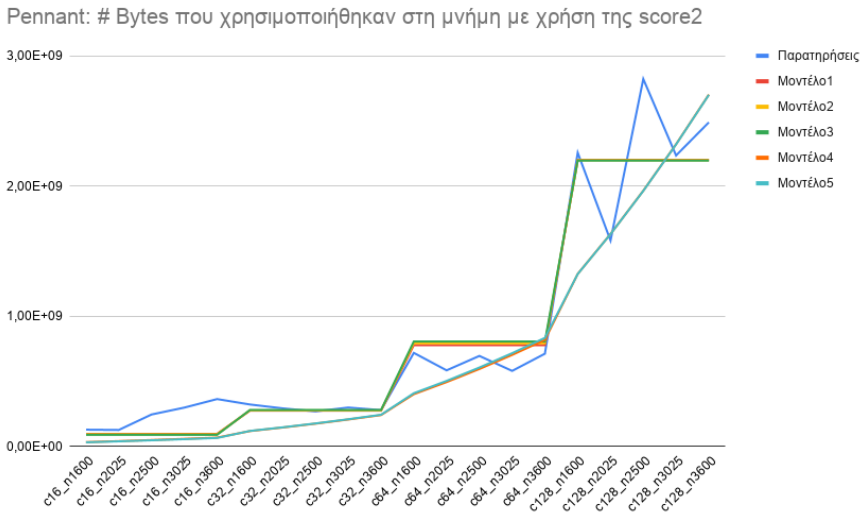



Σχήμα 4.21: *Τα 5 καλύτερα μοντέλα πρόβλεψης του αριθμού των Bytes που χρησιμοποιήθηκαν στη μνήμη για την εφαρμογή Pennant, με χρήση της 3.2*

| | Μοντέλο | Σχετικό Σφάλμα |
|---|---|---|
| Μοντέλο 1 | $2.543 \cdot 10^{3} \cdot p^{2} \cdot (log_2{}^{2}(p))$ | 0.5544 |
| Μοντέλο 2 | $1.151 \cdot 10^{-4} \cdot p^{2} \cdot log_2{}^{2}(p) \cdot n^{1.5} \cdot log_2{}^{2}(n)$ | 0.6598 |
| Μοντέλο 3 | $7 \cdot 10^{-3} \cdot n^{1.75} \cdot p^{1.75} \cdot log_2{}^{2}(p)$ | 0.6126 |
| Μοντέλο 4 | $1.23 \cdot 10^{-3} \cdot n^{1.75} \cdot log_2(n) \cdot p^{2} \cdot log_2(p)$ | 0.6287 |
| Μοντέλο 5 | $1.985 \cdot 10^{-2} \cdot n^{1.5} \cdot log_2(n) \cdot p^{2.25}$ | 0.6052 |

Πίνακας 4.22: *Τα 5 καλύτερα μοντέλα για τον αριθμό των Bytes που χρησιμοποιήθηκαν στη μνήμη με score =* $R^2$

Αντίστοιχα παρουσιάζουμε τα μοντέλα από τη συνδυαστική συνάρτηση αξιολόγησης στον πίνακα 4.23 και στο σχήμα 4.22.




Σχήμα 4.22: *Τα 5 καλύτερα μοντέλα πρόβλεψης του αριθμού των Bytes που χρησιμοποιήθηκαν στη μνήμη για την εφαρμογή Pennant, με χρήση της 3.3*

| | Μοντέλο | Σχετικό Σφάλμα |
|---|---|---|
| Μοντέλο 1 | $1.52 \cdot 10^6 \cdot p^{1.5}$ | 0.0897 |
| Μοντέλο 2 | $7.3 \cdot 10^5 \cdot p^{1.25} \cdot log_2(p)$ | 0.0875 |
| Μοντέλο 3 | $3.5 \cdot 10^5 \cdot p \cdot {log_2}^2(p)$ | 0.0851 |
| Μοντέλο 4 | $4.86 \cdot 10^1 \cdot p^{1.5} \cdot log_2(p) \cdot n^{0.75} \cdot log_2(n)$ | 0.335 |
| Μοντέλο 5 | $2.33 \cdot 10^1 \cdot p^{1.25} \cdot {log_2}^2(p) \cdot n^{0.75} \cdot log_2(n))$ | 0.3371 |

Πίνακας 4.23: *Τα 5 καλύτερα μοντέλα για τον αριθμό των Bytes που χρησιμοποιήθηκαν στη μνήμη με score* $= R^2 \cdot RCC \cdot pred_{<0.35}$

Όσον αφορά αυτή τη μετρική είναι ξεκάθαρη η εξάρτηση της από τον αριθμό των πυρήνων, κάτι που αποτυπώνεται και στα μοντέλα των δύο πινάκων με την "κυριαρχία" της παραμέτρου p. Το γεγονός αυτό φαίνεται έντονα στις προβλέψεις που έχουν προκύψει από τη δεύτερη συνάρτηση αξιολόγησης, κάτι που οφείλεται και στο γεγονός ότι τα μοντέλα αυτά επιζητούν όσο το δυνατό μικρότερο σφάλμα. Συγκεκριμένα βλέπουμε στον παρακάτω πίνακα αναλυτικά τις τιμές των μετρικών της συνδυαστικής αξιολόγησης:

| Μοντέλο | score | $R^2$ | $RCC$ | $pred_{<0.35}$ |
|---|---|---|---|---|
| Μοντέλο 1 | 0,55433 | 0,92714 | 0,74737 | 0,8 |
| Μοντέλο 2 | 0,4841 | 0,92527 | 0,74737 | 0,7 |
| Μοντέλο 3 | 0,483 | 0,92325 | 0,74737 | 0,7 |
| Μοντέλο 4 | 0,42403 | 0,89319 | 0,86316 | 0,55 |
| Μοντέλο 5 | 0,42342 | 0,8919 | 0,86316 | 0,55 |

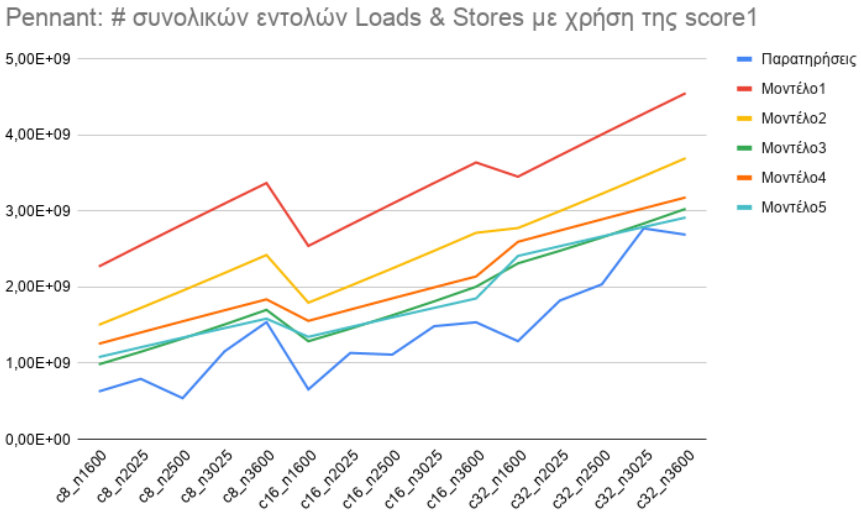
Πίνακας 4.24: *Αναλυτικά οι τιμές των επιμέρους μετρικών για τα μοντέλα του 4.23*

Παρατηρώντας τον πίνακα συνειδητοποιούμε ότι ο παραπάνω ισχυρισμός, που θέλει το γεγονός ότι η δεύτερη μέθοδος αξιολόγησης επιζητεί μικρότερο σφάλμα να είναι ο λόγος για την εξάρτηση των τριών πρώτων μοντέλων αποκλειστικά από το p, ευσταθεί. Αυτός είναι και

ο λόγος για τον οποίο η πρώτη μέθοδος αξιολόγησης δίνει μόνο ένα μοντέλο με μοναδική εξάρτηση από το p, μιας και ταξινομεί τις προβλέψεις με βάση το $R^2$. Τέλος, παρατηρούμε ότι τα μοντέλα τα οποία λαμβάνουν υπόψιν τους και το n έχουν καλύτερη επίδοση όσον αφορά το $RCC$.

### 4.3.4 Πλήθος εντολών πρόσβασης στη μνήμη (Loads/Stores)

Τέλος παρουσιάζουμε τις μετρήσεις και τις προβλέψεις μας για το πλήθος των εντολών πρόσβασης στη μνήμη που εκτελεί το `Pennant`.

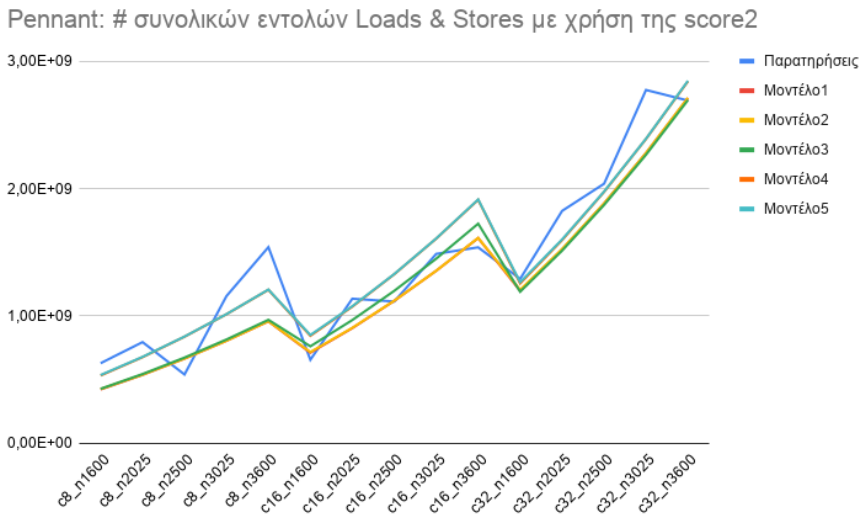



Σχήμα 4.23: *Τα 5 καλύτερα μοντέλα πρόβλεψης του αριθμού των Loads & Stores που εκτελέστηκαν για την εφαρμογή Pennant, με χρήση της 3.2*

| | Μοντέλο | Σχετικό Σφάλμα |
|---|---|---|
| Μοντέλο 1 | $3.01 \cdot 10^6 \cdot (p^{1.75} + n^{0.25} \cdot log_2{}^2(n))$ | 1.7074 |
| Μοντέλο 2 | $3.245 \cdot 10^6 \cdot (p^{1.75} + n^{0.5} \cdot log_2(n))$ | 0.9739 |
| Μοντέλο 3 | $3.385 \cdot 10^6 \cdot (p^{1.75} + n^{0.75})$ | 0.4474 |
| Μοντέλο 4 | $1.6 \cdot 10^6 \cdot (p^{1.5} \cdot log_2(p) + n^{0.25} \cdot log_2{}^2(n))$ | 0.652 |
| Μοντέλο 5 | $1.385 \cdot 10^6 \cdot (p^2 + n^{0.25} \cdot log_2{}^2(n))$ | 0.4548 |

Πίνακας 4.25: *Τα 5 καλύτερα μοντέλα για τον αριθμό των Loads & Stores που εκτελέστηκαν με score =* $R^2$

Αντίστοιχα παρουσιάζουμε τα μοντέλα από τη συνδυαστική συνάρτηση αξιολόγησης στον πίνακα 4.26 και στο σχήμα 4.24.




Σχήμα 4.24: *Τα 5 καλύτερα μοντέλα πρόβλεψης του αριθμού των Loads & Stores που εκτελέστηκαν για την εφαρμογή Pennant, με χρήση της 3.3*

| | Μοντέλο | Σχετικό Σφάλμα |
|---|---|---|
| Μοντέλο 1 | $3.105 \cdot 10^3 \cdot p^{0.75} \cdot n^{0.75} \cdot log_2{}^2(n)$ | 0.1162 |
| Μοντέλο 2 | $5.60 \cdot 10^4 \cdot p^{0.75} \cdot n$ | 0.1142 |
| Μοντέλο 3 | $1.661 \cdot 10^3 \cdot log_2{}^2(p) \cdot n^{0.75} \cdot log_2{}^2(n)$ | 0.0921 |
| Μοντέλο 4 | $3.684 \cdot 10^3 \cdot p^{0.25} \cdot log_2(p) \cdot n^{0.75} \cdot log_2{}^2(n)$ | 0.0256 |
| Μοντέλο 5 | $6.6481 \cdot 10^4 \cdot p^{0.25} \cdot log_2(p) \cdot n$ | 0.0286 |

Πίνακας 4.26: *Τα 5 καλύτερα μοντέλα για τον αριθμό των Loads & Stores που εκτελέστηκαν με score* $= R^2 \cdot RCC \cdot pred_{<0.35}$

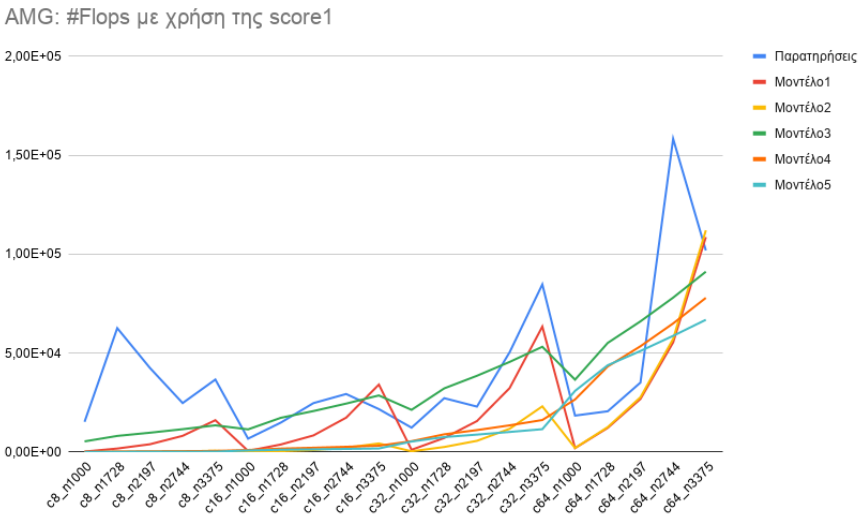
Τα παραπάνω δεδομένα μας δείχνουν μια αρκετά ικανοποιητική μοντελοποίηση, ειδικά για τα μοντέλα του πίνακα 4.26. Συγκεκριμένα βλέπουμε εξάρτηση κι από τις δύο παραμέτρους και υψηλή ανάγκη για εντολές πρόσβασης στη μνήμη, όσο τα (n,p) λαμβάνουν μεγαλύτερες τιμές. Αυτό μπορεί να οδηγήσει στη μη καλή κλιμάκωση της εφαρμογής σε συστήματα μεγαλύτερης κλίμακας, κάτι που υποδηλώνει ανάγκη για καλύτερη διαχείριση αυτού του πόρου από μεριάς υλοποίησης της εφαρμογής.

## 4.4 AMG2013

Το AMG2013 αποτελεί έναν παράλληλο αλγεβρικό πολυπλεγματικό "λύτη" για γραμμικά συστήματα που προκύπτουν από προβλήματα σε μη δομημένα πλέγματα. Αποτελεί έναν υψηλά συγχρονισμένο κώδικα στον οποίο τα μοτίβα επικοινωνίας ακολουθούν μεθόδους κοινούς σε πληθώρα παράλληλων επιστημονικών κωδίκων[22].

### 4.4.1 Flops

Στο σχήμα 4.25 και τον πίνακα 4.27 μπορούμε να δούμε τις μετρήσεις αλλά και τις προβλέψεις για το πλήθος των πράξεων κινητής υποδιαστολής flops που εκτελεί η εφαρμογή.

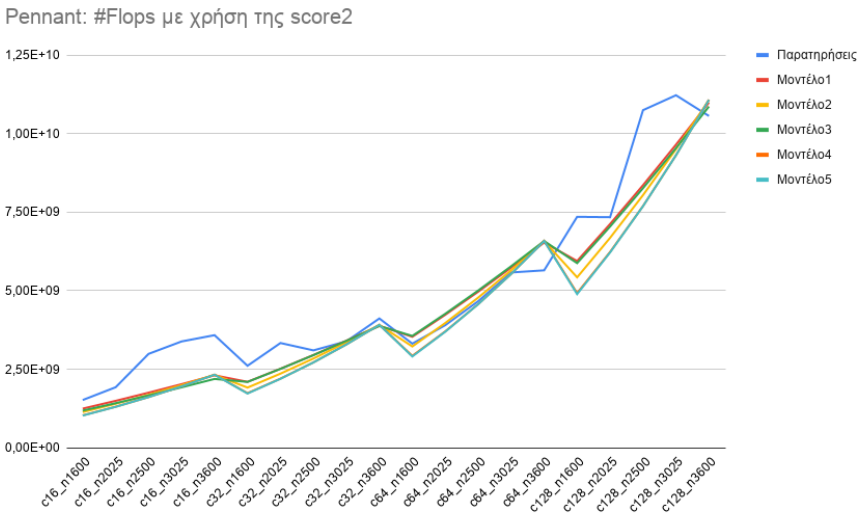



Σχήμα 4.25: *Τα 5 καλύτερα μοντέλα πρόβλεψης του αριθμού των flops που εκτελέστηκαν για την εφαρμογή AMG2013, με χρήση της 3.2*

| | Μοντέλο | Σχετικό Σφάλμα |
|---|---|---|
| Μοντέλο 1 | $2.02 \cdot 10^{-10} \cdot n^3 \cdot log_2{}^2(n) \cdot p^{0.25} \cdot log_2{}^2(p)$ | 0.5455 |
| Μοντέλο 2 | $4.07 \cdot 10^{-13} \cdot n^3 \cdot log_2{}^2(n) \cdot p^{1.75} \cdot log_2{}^2(p)$ | 0.7863 |
| Μοντέλο 3 | $2.022 \cdot n^{0.75} \cdot p^{0.25} \cdot log_2{}^2(p)$ | 0.0716 |
| Μοντέλο 4 | $6.11 \cdot 10^{-4} \cdot n^{0.75} \cdot log_2(n) \cdot p^2 \cdot log_2(p)$ | 0.5681 |
| Μοντέλο 5 | $6.66 \cdot 10^{-4} \cdot n^{0.5} \cdot log_2(n) \cdot p^2 \cdot log_2{}^2(p)$ | 0.5921 |

Πίνακας 4.27: *Τα 5 καλύτερα μοντέλα για τον αριθμό των flops που εκτελέστηκαν με score* $= R^2$

Αντίστοιχα παρουσιάζουμε τα μοντέλα από τη συνδυαστική συνάρτηση αξιολόγησης στον πίνακα 4.28 και στο σχήμα 4.26.




Σχήμα 4.26: *Τα 5 καλύτερα μοντέλα πρόβλεψης του αριθμού των flops που εκτελέστηκαν για την εφαρμογή AMG2013, με χρήση της 3.3*

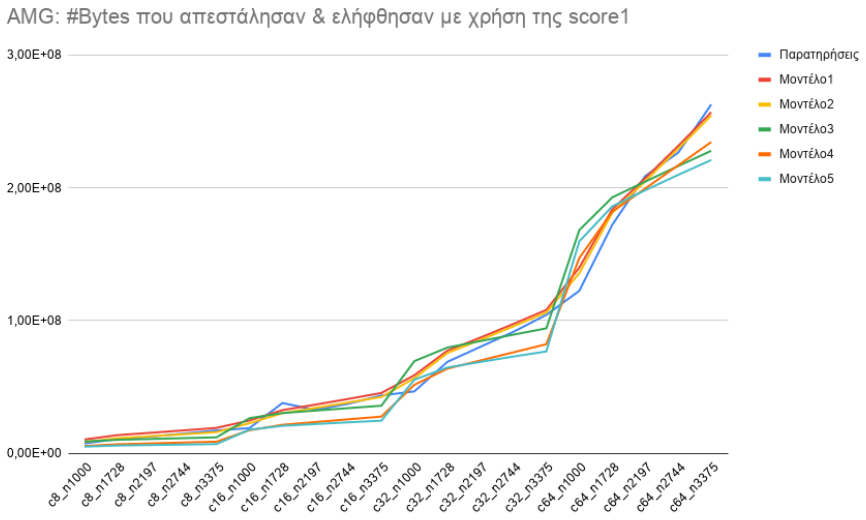
| | Μοντέλο | Σχετικό Σφάλμα |
|---|---|---|
| Μοντέλο 1 | $5.97 \cdot 10^{-3} \cdot p^{0.5} \cdot n^{1.5} \cdot log_2(n)$ | 0.0857 |
| Μοντέλο 2 | $2.79 \cdot 10^{-3} \cdot p^{0.25} \cdot log_2(p) \cdot n^{1.5} \cdot log_2(n)$ | 0.0276 |
| Μοντέλο 3 | $5.04 \cdot 10^{-4} \cdot p^{0.5} \cdot n^{1.5} \cdot {log_2}^2(n)$ | 0.0164 |
| Μοντέλο 4 | $9.09 \cdot 10^{-3} \cdot p^{0.5} \cdot n^{1.75}$ | 0.0217 |
| Μοντέλο 5 | $1.1 \cdot 10^{-4} \cdot {log_2}^2(p) \cdot n^{1.5} \cdot {log_2}^2(n)$ | 0.0824 |

Πίνακας 4.28: *Τα 5 καλύτερα μοντέλα για τον αριθμό των flops που εκτελέστηκαν με* $score = R^2 \cdot RCC \cdot pred_{<0.35}$

Αρχικά μπορούμε να σχολιάσουμε το γεγονός ότι τα μοντέλα τα οποία δημιουργήθηκαν με χρήση της συνδυαστικής συνάρτησης αξιολόγησης μοιάζουν, και σε αυτή την περίπτωση, να πετυχαίνουν καλύτερες προβλέψεις. Το πετυχαίνουν αυτό καθώς, χωρίς να εξαρτώνται μόνο από το $R^2$, δεν επηρεάζονται στον ίδιο βαθμό από κάποιες, φαινομενικά, τυχαίες κορυφές στις μετρήσεις, οι οποίες κι οφείλονται, κατά πάσα πιθανότητα, στα μηχανήματα της ουράς clones. Παρατηρούμε ακόμη εξάρτηση των μοντέλων, και των δύο πινάκων, από το γινόμενο και των δύο παραμέτρων, κάτι το οποίο μαρτυρά πως η εφαρμογή δεν αναμένεται να κλιμακώνει με επιθυμητό τρόπο σε συστήματα μεγαλύτερης κλίμακας.

### 4.4.2 Αριθμός Bytes που ελήφθησαν και απεστάλησαν ανά διαδικασία

Ως δεύτερη παρουσιάζουμε τη μετρική του πλήθους Bytes που ανταλλάχθηκαν κατά την επικοινωνία των διαδικασιών, ξεκινώντας κι εδώ με τα αποτελέσματα από τη συνάρτηση αξιολόγησης $score_1 = R^2$.

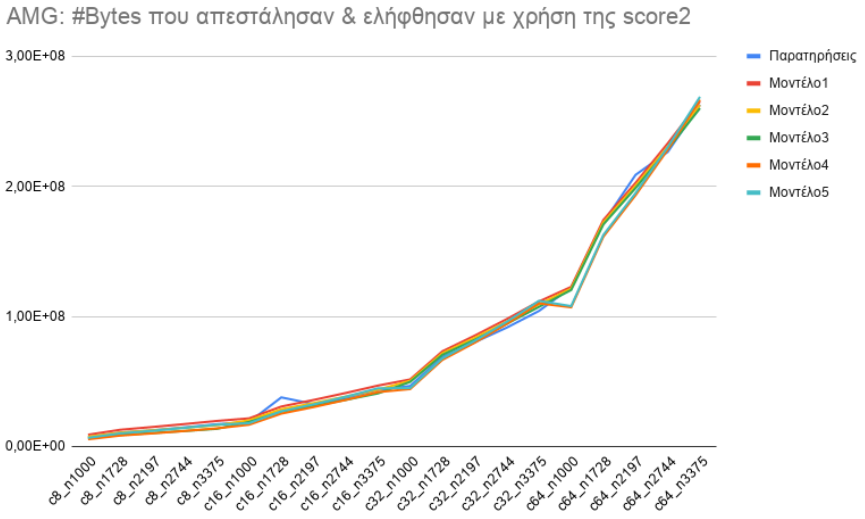



Σχήμα 4.27: *Τα 5 καλύτερα μοντέλα πρόβλεψης του αριθμού των Bytes που απεστάλησαν και ελήφθησαν για την εφαρμογή AMG2013, με χρήση της 3.2*

| | Μοντέλο | Σχετικό Σφάλμα |
|---|---|---|
| Μοντέλο 1 | $2.4422 \cdot 10^4 \cdot n^{0.5} \cdot p^{1.25}$ | 0.1192 |
| Μοντέλο 2 | $6.3265 \cdot 10^2 \cdot n^{0.25} \cdot log_2{}^2(n) \cdot p \cdot log_2(p)$ | 0.036 |
| Μοντέλο 3 | $3.6695 \cdot 10^4 \cdot n^{0.25} \cdot p^{0.75} \cdot log_2{}^2(p)$ | 0.0127 |
| Μοντέλο 4 | $2.416 \cdot 10^3 \cdot n^{0.25} \cdot log_2(n) \cdot p^{1.25} \cdot log_2(p)$ | 0.1939 |
| Μοντέλο 5 | $6.9777 \cdot 10^2 \cdot log_2{}^2(n) \cdot p \cdot log_2{}^2(p)$ | 0.2239 |

Πίνακας 4.29: *Τα 5 καλύτερα μοντέλα για τον αριθμό των Bytes που απεστάλησαν κι ελήφθησαν με score =* $R^2$

Ενώ τα αντίστοιχα μοντέλα με χρήση της 3.3 είναι:




Σχήμα 4.28: *Τα 5 καλύτερα μοντέλα πρόβλεψης του αριθμού των Bytes που απεστάλησαν και ελήφθησαν για την εφαρμογή AMG2013, με χρήση της 3.3*

| | Μοντέλο | Σχετικό Σφάλμα |
|---|---|---|
| Μοντέλο 1 | $2.152 \cdot 10^3 \cdot p^{1.25} \cdot n^{0.5} \cdot log_2(n)$ | 0.0802 |
| Μοντέλο 2 | $1 \cdot 10^3 \cdot n^{0.5} \cdot log_2(n) \cdot p \cdot log_2(p)$ | 0.0046 |
| Μοντέλο 3 | $4.69 \cdot 10^2 \cdot n^{0.5} \cdot log_2(n) \cdot p^{0.75} \cdot log_2{}^2(p)$ | 0.06 |
| Μοντέλο 4 | $7.39 \cdot 10^2 \cdot p^{0.75} \cdot log_2{}^2(p) \cdot n^{0.75}$ | 0.0921 |
| Μοντέλο 5 | $1.581 \cdot 10^3 \cdot p \cdot log_2(p) \cdot n^{0.75}$ | 0.0305 |

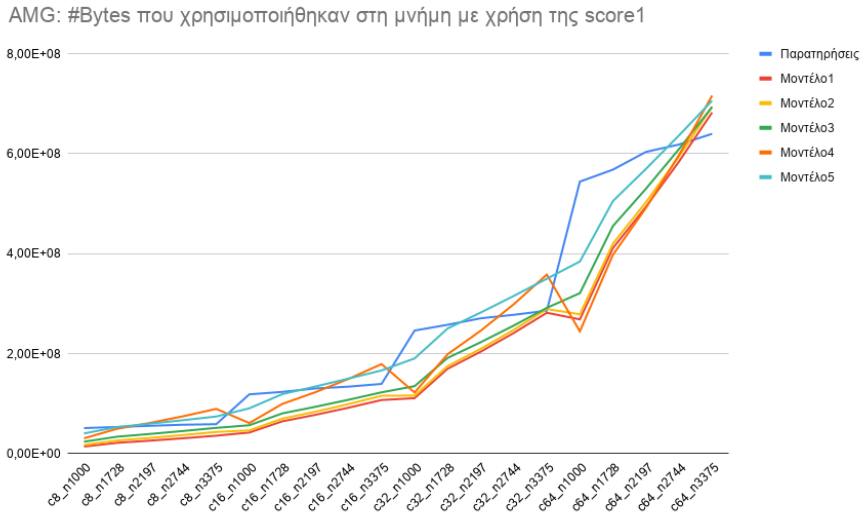
Πίνακας 4.30: *Τα 5 καλύτερα μοντέλα για τον αριθμό των Bytes που απεστάλησαν κι ελήφθησαν με score =* $R^2 \cdot RCC \cdot pred_{<0.35}$

Όσον αφορά τις ανάγκες της εφαρμογής σε επικοινωνία, παρατηρούμε ότι, και με τις δύο μεθόδους αξιολόγησης, τα μοντέλα μας είναι ικανά να προβλέψουν σε ικανοποιητικότατο βαθμό το πλήθος των Bytes που ανταλλάσσονται για επικοινωνία. Αυτό οφείλεται και σε μεγάλο βαθμό στο γεγονός ότι το `AMG2013` χρησιμοποιεί συναρτήσεις "συγκεντρωτικής" επικοινωνίας, σαν αυτές που χρησιμοποιεί και το `Pennant` και οι οποίες περιγράφηκαν στην αντίστοιχη ενότητα αυτής της εφαρμογής. Τέλος, όσον αφορά το κατά πόσο οι ανάγκες του `AMG2013` σε επικοινωνία θα αποτελούν πιθανή αιτία καθυστέρησης της εφαρμογής όσο αυξάνουν οι

παράμετροι (n,p), βλέπουμε ότι, μιας και τα μοντέλα εξαρτώνται από το γινόμενο των δύο παραμέτρων, αυτό είναι κάτι το οποίο οι δημιουργοί της εφαρμογής θα πρέπει να αναμένουν.

### 4.4.3 Αριθμός Bytes που χρησιμοποιήθηκαν στη μνήμη

Παρουσιάζουμε εδώ τη μετρική που εκφράζει τις ανάγκες της εφαρμογής σε μνήμη.

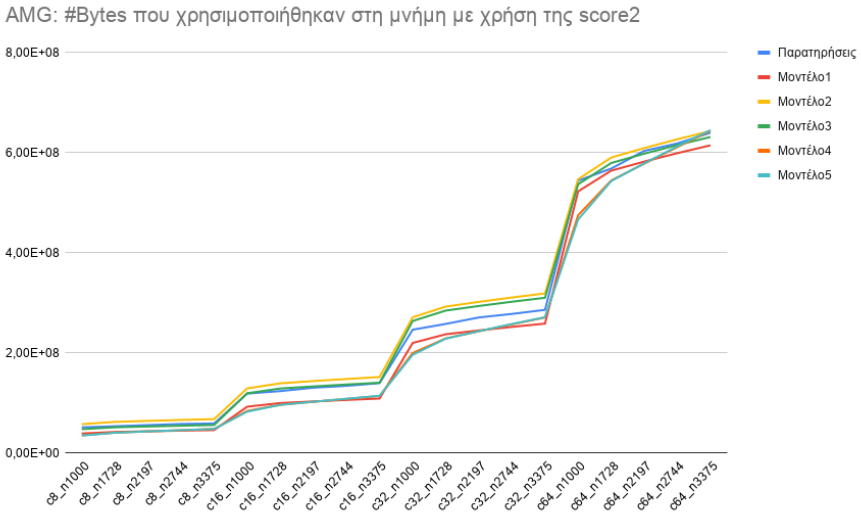



Σχήμα 4.29: *Τα 5 καλύτερα μοντέλα πρόβλεψης του αριθμού των Bytes που χρησιμοποιήθηκαν στη μνήμη για την εφαρμογή AMG2013, με χρήση της 3.2*

| | Μοντέλο | Σχετικό Σφάλμα |
|---|---|---|
| Μοντέλο 1 | $1.05 \cdot 10^2 \cdot p^{0.75} \cdot log_2{}^2(p) \cdot n^{0.5} \cdot log_2{}^2(n)$ | 0.3509 |
| Μοντέλο 2 | $4.084 \cdot 10^3 \cdot n^{0.75} \cdot p \cdot log_2(p)$ | 0.3045 |
| Μοντέλο 3 | $5.628 \cdot 10^3 \cdot n^{0.5} \cdot log_2(n) \cdot p^{1.25}$ | 0.2302 |
| Μοντέλο 4 | $2.157 \cdot 10^3 \cdot n^{0.75} \cdot log_2(n) \cdot p$ | 0.0667 |
| Μοντέλο 5 | $8.9545 \cdot 10^4 \cdot n^{0.5} \cdot p^{0.75} \cdot log_2(p)$ | 0.0086 |

Πίνακας 4.31: *Τα 5 καλύτερα μοντέλα για τον αριθμό των Bytes που χρησιμοποιήθηκαν στη μνήμη με score =* $R^2$

Αντίστοιχα παρουσιάζουμε τα μοντέλα από τη συνδυαστική συνάρτηση αξιολόγησης στον πίνακα και στο σχήμα που ακολουθούν.




Σχήμα 4.30: *Τα 5 καλύτερα μοντέλα πρόβλεψης του αριθμού των Bytes που χρησιμοποιήθηκαν στη μνήμη για την εφαρμογή AMG2013, με χρήση της 3.3*

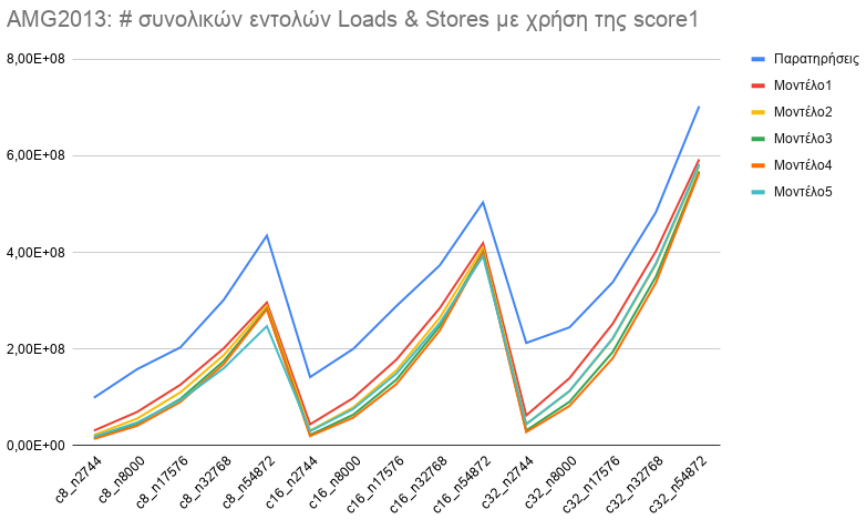
| | Μοντέλο | Σχετικό Σφάλμα |
|---|---|---|
| Μοντέλο 1 | $2.8976 \cdot 10^5 \cdot p^{1.25} \cdot log_2(n))$ | 0.1406 |
| Μοντέλο 2 | $4.04 \cdot 10^5 \cdot p^{0.75} \cdot log_2(p) \cdot log_2(n))$ | 0.0943 |
| Μοντέλο 3 | $1.87 \cdot 10^5 \cdot p^{0.5} \cdot {log_2}^2(p) \cdot log_2(n)$ | 0.0115 |
| Μοντέλο 4 | $4.666 \cdot 10^5 \cdot p^{1.25} \cdot n^{0.25}$ | 0.1518 |
| Μοντέλο 5 | $2.59 \cdot 10^4 \cdot p^{1.25} \cdot {log_2}^2(n)$ | 0.154 |

Πίνακας 4.32: *Τα 5 καλύτερα μοντέλα για τον αριθμό των Bytes που χρησιμοποιήθηκαν στη μνήμη με score* $= R^2 \cdot RCC \cdot pred_{<0.35}$

Παρατηρούμε, και σε αυτή τη μετρική, μια αρκετά ικανοποιητική μοντελοποίηση των παρατηρήσεων με τη μέθοδο εξαγωγής προβλέψεων που έχουμε περιγράψει. Πιο συγκεκριμένα, οι προβλέψεις που προκύπτουν με τη συνδυαστική συνάρτηση αξιολόγησης φαίνεται να προβλέπουν με μεγάλη επιτυχία τις ανάγκες της εφαρμογής σε μνήμη, ενώ και ο άλλος τρόπος αξιολόγησης δίνει ικανοποιητικά αποτελέσματα. Σε ό,τι αφορά τα μοντέλα των δύο πινάκων παρατηρούμε εξάρτηση του αριθμού Bytes στη μνήμη από το γινόμενο των δύο παραμέτρων κάτι το οποίο, όπως έχει προαναφερθεί, μπορεί να οδηγήσει την εν λόγω ανάγκη της εφαρμογής σε καθυστερήσεις στην εκτέλεση της εφαρμογής σε συστήματα μεγαλύτερης κλίμακας.

### 4.4.4 Πλήθος εντολών πρόσβασης στη μνήμη (Loads/Stores)

Τέλος παρουσιάζουμε τις μετρήσεις και τις προβλέψεις μας για το πλήθος των εντολών Loads & Stores που εκτελεί το `AMG2013`.

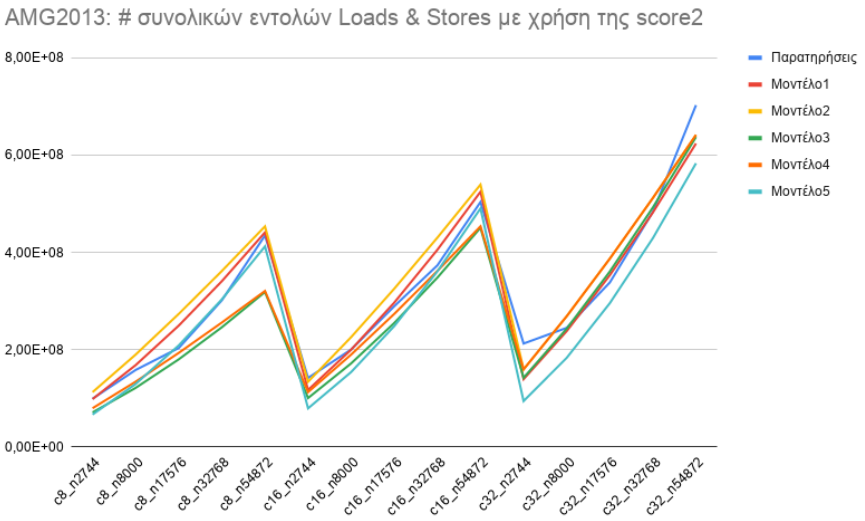



Σχήμα 4.31: *Τα 5 καλύτερα μοντέλα πρόβλεψης του αριθμού των Loads & Stores που εκτελέστηκαν για την εφαρμογή AMG2013, με χρήση της 3.2*

| | Μοντέλο | Σχετικό Σφάλμα |
|---|---|---|
| Μοντέλο 1 | $2.92 \cdot 10^4 \cdot p^{0.5} \cdot n^{0.75}$ | 0.3979 |
| Μοντέλο 2 | $1.81 \cdot 10^3 \cdot p^{0.5} \cdot n^{0.75} \cdot log_2(n)$ | 0.4659 |
| Μοντέλο 3 | $1.13 \cdot 10^2 \cdot p^{0.5} \cdot n^{0.75} \cdot {log_2}^2(n)$ | 0.5204 |
| Μοντέλο 4 | $1.82 \cdot 10^3 \cdot p^{0.5} \cdot n$ | 0.5421 |
| Μοντέλο 5 | $8.69 \cdot 10^2 \cdot p^{0.25} \cdot log_2(p) \cdot n^{0.75} \cdot log_2(n)$ | 0.4955 |

Πίνακας 4.33: *Τα 5 καλύτερα μοντέλα για τον αριθμό των Loads & Stores που εκτελέστηκαν με score =* $R^2$

Αντίστοιχα παρουσιάζουμε τα μοντέλα από τη συνδυαστική συνάρτηση αξιολόγησης στον πίνακα και στο σχήμα που ακολουθούν:




Σχήμα 4.32: *Τα 5 καλύτερα μοντέλα πρόβλεψης του αριθμού των Loads & Stores που εκτελέστηκαν για την εφαρμογή AMG2013, με χρήση της 3.3*

| | Μοντέλο | Σχετικό Σφάλμα |
|---|---|---|
| Μοντέλο 1 | $1.12 \cdot 10^6 \cdot p^{0.25} \cdot n^{0.5}$ | 0.0026 |
| Μοντέλο 2 | $7.11 \cdot 10^4 \cdot p^{0.25} \cdot n^{0.25} \cdot log_2{}^2(n)$ | 0.0871 |
| Μοντέλο 3 | $4.81 \cdot 10^5 \cdot p^{0.5} \cdot n^{0.5}$ | 0.1433 |
| Μοντέλο 4 | $2.99 \cdot 10^4 \cdot p^{0.5} \cdot n^{0.25} \cdot log_2{}^2(n)$ | 0.0858 |
| Μοντέλο 5 | $6.65 \cdot 10^4 \cdot p^{0.25} \cdot n^{0.5} \cdot log_2(n)$ | 0.1742 |

Πίνακας 4.34: *Τα 5 καλύτερα μοντέλα για τον αριθμό των Loads & Stores που εκτελέστηκαν με score* $= R^2 \cdot RCC \cdot pred_{<0.35}$

Παρατηρώντας τα παραπάνω σχήματα αξίζει να σχολιάσουμε αρχικά ότι οι μετρήσεις μοιάζουν εύκολα μοντελοποιήσιμες με αποτέλεσμα και οι δύο μέθοδοι αξιολόγησης των μοντέλων να μας δίνουν ικανοποιητικά αποτελέσματα. Πιο συγκεκριμένα για τη συνάρτηση αξιολόγησης $score_1 = R^2$, παρατηρούμε ότι και τα πέντε μοντέλα που έχουν επιλεγεί παρουσιάζουν ιδιαίτερα καλή προσαρμογή στις παρατηρήσεις, παρόλο που φαίνεται να αποκλίνουν από αυτές, κάτι που δικαιολογεί και μεγαλύτερες τιμές σχετικού σφάλματος και λογικά θα μπορούσε να αποφευχθεί με την επιλογή μεγαλύτερων συντελεστών. Όσον αφορά τα μοντέλα του 4.32 παρατηρούμε κι εδώ ικανοποιητική προσαρμογή, καθώς και μικρή απόκλιση, δηλαδή χαμηλές τιμές σφαλμάτων, από τις μετρήσεις. Τέλος, οι πίνακες 4.33 και 4.34 δεν έχουν κάποιο κοινό μοντέλο, κάτι που διαφαίνεται κι από τις γραφικές λόγω των μεγαλύτερων σφαλμάτων των μοντέλων της $score_1$. Και τα δέκα μοντέλα που παρουσιάζονται στους δύο πίνακες εξαρτώνται από το γινόμενο των δύο παραμέτρων κάτι που, παρά τους μικρούς εκθέτες αυτών, μαρτυρά πως οι απαιτήσεις της εφαρμογής σε εντολές πρόσβασης στη μνήμη δύναται να οδηγήσει την εκτέλεση της εφαρμογής σε συμφόρηση σε μεγαλύτερης κλίμακας συστήματα και προβλήματα.

# Μέρος IV

# Χρόνος Εκτέλεσης και Συν-σχεδίαση

Κεφάλαιο 5

# Μοντελοποίηση του χρόνου εκτέλεσης της εφαρμογής με βάση τις ανάγκες αυτής σε πόρους

Σε αυτό το κεφάλαιο θα επεκτείνουμε την ιδέα της παραγωγής μοντέλων πρόβλεψης που παρουσιάσθηκε στα προηγούμενα κεφάλαια με σκοπό την εξαγωγή προβλέψεων για το χρόνο εκτέλεσης των εφαρμογών συναρτήσει των αναγκών αυτών σε πόρους.

## 5.1 Διαδικασία εξαγωγής των μοντέλων

Κίνητρο για τις διαδικασίες που θα παρουσιασθούν σε αυτό το κεφάλαιο αποτελεί η προσπάθεια να εκφραστεί ο χρόνος εκτέλεσης μιας εφαρμογής ως συνάρτηση των αναγκών αυτής στους πόρους που εξετάστηκαν στα προηγούμενα κεφάλαια.

Ορμώμενοι από το παραπάνω, και με σκοπό την παραγωγή μοντέλων πρόβλεψης για τους χρόνους εκτέλεσης, ορίζουμε πλέον τα μοντέλα μας ως συναρτήσεις της μορφής:

$$Time(flops, bytes\_sent\_received, bytes\_used, loads\_stores) \tag{5.1}$$

Δεδομένου ότι όλες οι εφαρμογές που μελετάμε χρησιμοποιούν το MPI, λαμβάνουμε μετρήσεις για τον χρόνο εκτέλεσης με χρήση της συνάρτησης `MPI_Wtime()`. Προκειμένου να δώσουμε τιμές στις παραμέτρους του παραπάνω τύπου συνάρτησης 5.1 χρησιμοποιούμε τα όσα παρουσιάσαμε στα προηγούμενα κεφάλαια: Βασιζόμενοι στα μοντέλα που παρουσιάσθηκαν για κάθε πόρο, λαμβάνουμε τιμές για την εκάστοτε μετρική από το καλύτερο μοντέλο που έχουμε για αυτή μέσω της πρότερής μας εργασίας και στη συνέχεια χρησιμοποιούμε αυτές τις τιμές αντιστοιχίζοντας τες με τα $(n, p)$ των προηγούμενων κεφαλαίων σε μια πανομοιότυπη διαδικασία παραγωγής μοντέλων. Σημειώνουμε επίσης ότι και σε αυτή τη διαδικασία παράγουμε μοντέλα τόσο με τη συνάρτηση αξιολόγησης 3.2 ($score_1 = R^2$) όσο και με τη συνδυαστική συνάρτηση αξιολόγησης 3.3 ($score_2 = R^2 \cdot RCC \cdot pred_{<0.35}$). Μια σχηματική αναπαράσταση της μεθόδου εξαγωγής των μοντέλων χρόνου παρουσιάζεται παρακάτω:


Σχήμα 5.1: *Βήματα παραγωγής μοντέλων χρόνου*

Αποτέλεσμα είναι η παραγωγή μοντέλων πρόβλεψης για τον χρόνο εκτέλεσης των τεσσάρων εφαρμογών τα οποία μπορούν να αναδείξουν το πόσο δύναται να επηρεαστεί αυτός από τις ανάγκες του εκάστοτε κώδικα σε κάποιο συγκεκριμένο πόρο και να φανερώσει συσχετίσεις ανάμεσα στα δύο. Μια τέτοια διαδικασία είναι σημαντική γιατί δύναται να αναδείξει κάποια ανάγκη μιας εφαρμογής σε συγκεκριμένους πόρους, όπως υπολογισιμότητα, μνήμη, κτλ., ως υπαίτιο για χρονικές καθυστερήσεις στην εκτέλεση της εφαρμογής.

## 5.2 Χρόνος εκτέλεσης εφαρμογών

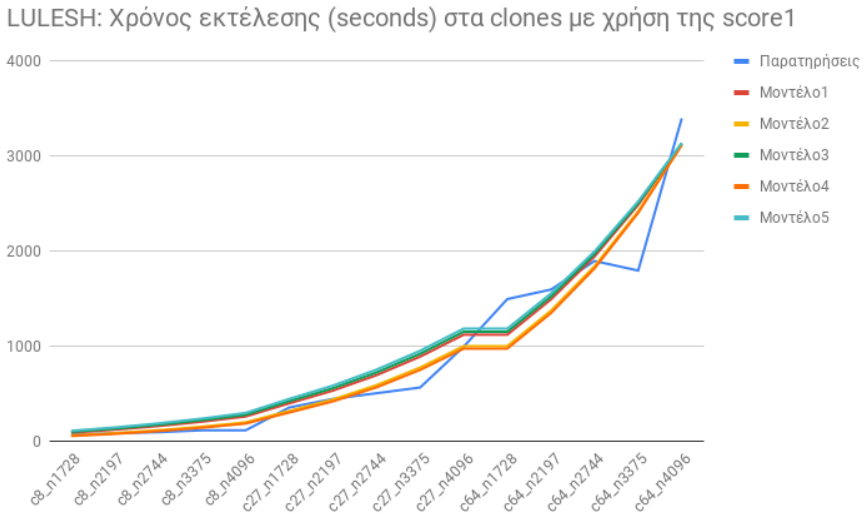
Σε αυτή την ενότητα θα προχωρήσουμε στην παρουσίαση και τον σχολιασμό των μοντέλων για το χρόνο εκτέλεσης των τεσσάρων εφαρμογών μας, τα οποία κι εξάγονται με τον τρόπο που αναλύθηκε προηγουμένως. Λάβαμε μετρήσεις τόσο στα μηχανήματα της ουράς clones όσο και στο μηχάνημα 3. Είναι σωστό να τονίσουμε ότι τα clones αποτελούν παλαιά συστοιχία μηχανημάτων, το δίκτυο τον οποίων επιβαρύνεται από έντονη κίνηση εισόδου/εξόδου. Για αυτό τον λόγο, είναι αρκετά δύσκολο να λάβουμε ικανοποιητικά αποτελέσματα σχετικά με τον χρόνο εκτέλεσης μιας εφαρμογής, ωστόσο τα παρουσιάζουμε σε αυτή την ενότητα με σκοπό την σύγκριση αυτών με τα αποτελέσματα ενός πιο σύγχρονου μηχανήματος.

*Σε όλους τους πίνακες της ενότητας έχει γίνει η εξής αντιστοίχηση μεγεθών:*

- *f αντιστοιχεί στον αριθμό των πράξεων κινητής υποδιαστολής (FLOPS)*
- *sr αντιστοιχεί στον αριθμό Bytes που απεστάλησαν/ελήφθησαν ανά διαδικασία*
- *ls αντιστοιχεί στον αριθμό εντολών πρόσβασης στη μνήμη (Loads & Stores)*
- *b αντιστοιχεί στον αριθμό Bytes που χρησιμοποιήθηκαν στη μνήμη*

### 5.2.1 LULESH

Θα ξεκινήσουμε την παρουσίαση των αποτελεσμάτων μας με την εφαρμογή LULESH. Στα παρακάτω σχήματα και πίνακες παρουσιάζουμε τα αποτελέσματά μας για τα μηχανήματα clones με χρήση των δύο διαφορετικών συναρτήσεων αξιολόγησης.




Σχήμα 5.2: *Τα 5 καλύτερα μοντέλα πρόβλεψης του χρόνου εκτέλεσης για την εφαρμογή* `LULESH` *στα μηχανήματα clones, με χρήση της* $score_1$ *3.2*


Σχήμα 5.3: *Τα 5 καλύτερα μοντέλα πρόβλεψης του χρόνου εκτέλεσης για την εφαρμογή* `LULESH` *στα μηχανήματα clones, με χρήση της* $score_2$ *3.3*

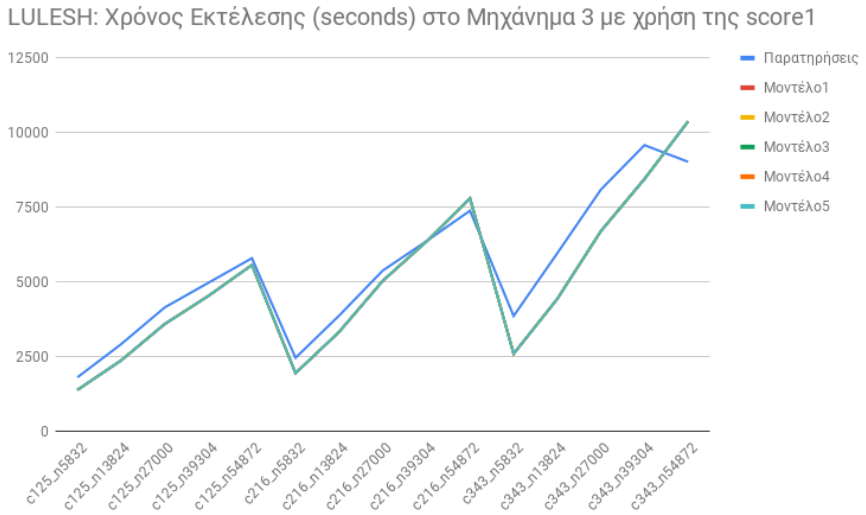
| $score_1$ | Σχετικό Σφάλμα |
|---|---|
| $1.24 \cdot 10^{-16} \cdot f^{1.5} \cdot log_2{}^2(f)$ | 0.3126 |
| $3.1 \cdot 10^{-16} \cdot f^{1.75}$ | 0.0826 |
| $4.48 \cdot 10^{-15} \cdot f^{1.5} \cdot log_2(f)$ | 0.3788 |
| $8.61 \cdot 10^{-18} \cdot f^{1.75} \cdot log_2(f)$ | 0.0389 |
| $1.62 \cdot 10^{-13} \cdot f^{1.5}$ | 0.4498 |

Πίνακας 5.1: *Τα 5 καλύτερα μοντέλα για χρόνο εκτέλεσης της εφαρμογής* `LULESH` *στα μηχανήματα clones με χρήση της* $score_1$

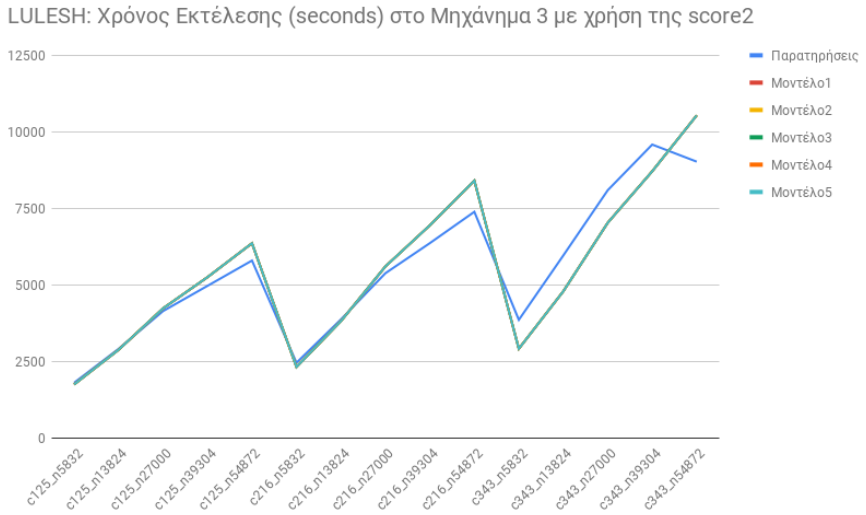
| $score_2$ | Σχετικό Σφάλμα |
|---|---|
| $1.04 \cdot 10^{-10} \cdot f^{1.25}$ | 0.0086 |
| $2.9 \cdot 10^{-12} \cdot f^{1.25} \cdot log_2(f)$ | 0.0663 |
| $8.11 \cdot 10^{-14} \cdot f^{1.25} \cdot log_2{}^2(f)$ | 0.1189 |
| $4 \cdot 10^{-11} \cdot f \cdot log_2{}^2(f)$ | 0.2765 |
| $2.11 \cdot 10^{-13} \cdot f^{1.5}$ | 0.2861 |

Πίνακας 5.2: *Τα 5 καλύτερα μοντέλα για χρόνο εκτέλεσης της εφαρμογής* `LULESH` *στα μηχανήματα clones με χρήση της* $score_2$

Στη συνέχεια παρουσιάζουμε τα μοντέλα και τα αποτελέσματά μας στο μηχάνημα 3




(α′) $score_1$




(β′) $score_2$

Σχήμα 5.4: *Τα 5 καλύτερα μοντέλα πρόβλεψης του χρόνου εκτέλεσης για την εφαρμογή* ***LULESH*** *στο μηχάνημα 3, με χρήση των 3.2 και 3.3*

| $score_1$ | Σχετικό Σφάλμα |
|---|---|
| $4.0667 \cdot 10^{-9} \cdot f^{0.75} \cdot log_2{}^2(f)$ | 0.1174 |
| $4.065 \cdot 10^{-9} \cdot (sr^{0.75} \cdot log_2(sr) + f^{0.75} \cdot log_2{}^2(f) + b^{0.25} + ls^{0.25})$ | 0.1174 |
| $4.065 \cdot 10^{-9} \cdot (sr^{0.75} \cdot log_2(sr) + f^{0.75} \cdot log_2{}^2(f) + b^{0.25} \cdot log_2(b) + ls^{0.25})$ | 0.1174 |
| $4.065 \cdot 10^{-9} \cdot (sr^{0.75} \cdot log_2(sr) + f^{0.75} \cdot log_2{}^2(f) + b^{0.25} \cdot log_2(b) + ls^{0.25} \cdot log_2(ls))$ | 0.1174 |
| $4.065 \cdot 10^{-9} \cdot (sr^{0.75} \cdot log_2(sr) + f^{0.75} \cdot log_2{}^2(f) + b^{0.5} + ls^{0.25})$ | 0.1174 |

Πίνακας 5.3: *Τα 5 καλύτερα μοντέλα για χρόνο εκτέλεσης της εφαρμογής* ***LULESH*** *στο μηχάνημα 3 με χρήση της* $score_1$

| $score_2$ | Σχετικό Σφάλμα |
|---|---|
| $3.1657 \cdot 10^{-6} \cdot f^{0.5} \cdot {log_2}^2(f)$ | 0.0108 |
| $3.1656 \cdot 10^{-6} \cdot (sr^{0.5} + f^{0.5} \cdot {log_2}^2(f) + b^{0.5} + ls^{0.25} \cdot log_2(ls))$ | 0.0108 |
| $3.165 \cdot 10^{-6} \cdot (sr^{0.5} + f^{0.5} \cdot {log_2}^2(f) + b^{0.5} \cdot log_2(b) + ls^{0.25} \cdot log_2(ls))$ | 0.0108 |
| $3.165 \cdot 10^{-6} \cdot (sr^{0.5} + f^{0.5} \cdot {log_2}^2(f) + b^{0.5} + ls^{0.5})$ | 0.0108 |
| $3.165 \cdot 10^{-6} \cdot (sr^{0.5} + f^{0.5} \cdot {log_2}^2(f) + b^{0.75} + ls^{0.25} \cdot log_2(ls))$ | 0.0108 |

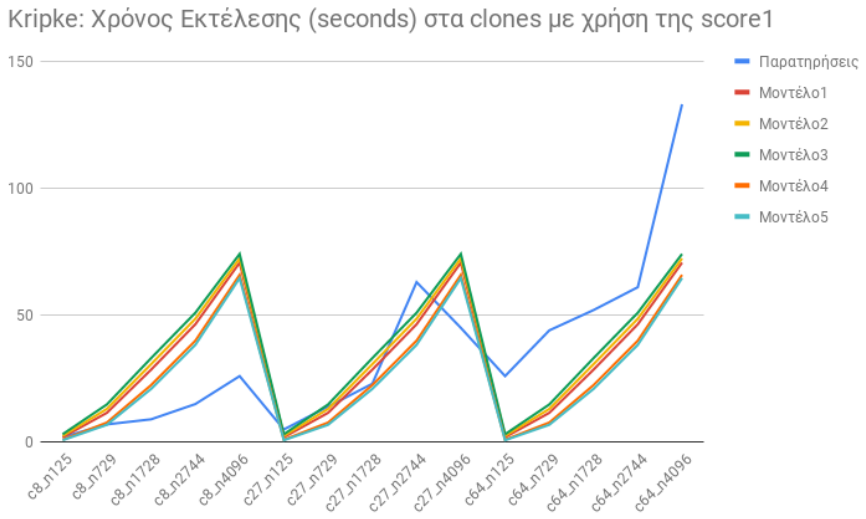
Πίνακας 5.4: *Τα 5 καλύτερα μοντέλα για χρόνο εκτέλεσης της εφαρμογής* `LULESH` *στο μηχάνημα 3 με χρήση της* $score_2$

Όσον αφορά τις παρατηρήσεις των δύο διαφορετικών μηχανημάτων, παρατηρούμε ότι αυτές διαφέρουν αρκετά μεταξύ τους. Συγκεκριμένα για τα δεδομένα από τη συστοιχία clones, είναι εμφανές ότι διαθέτουν αρκετά σημεία με δύσκολα μοντελοποιήσιμες τιμές, όπως π.χ. το σημείο *c64_n3375*, τα οποία δυσκολεύουν την παραγωγή σχετικών προβλέψεων. Επιπροσθέτως, στις μετρήσεις από τα clones παρατηρούμε μια "εκθετική" αύξηση του χρόνου εκτέλεσης, λογικά αποτέλεσμα των περιορισμών της συστοιχίας. Αυτή η εκθετική τάση ίσως δικαιολογεί και το γεγονός ότι στον πίνακα 5.1 παρατηρούμε μοντέλα που βασίζονται μόνο στη μετρική FLOPS. Ανατρέχοντας κανείς στον πίνακα 4.1 παρατηρεί ότι το αντίστοιχο μοντέλο έχει αποκλειστικά εκθετική εξάρτηση από τα μεγέθη (n,p), σε αντίθεση με τα μοντέλα των άλλων μετρικών που παρουσιάζουν και λογαριθμικές εξαρτήσεις ή κι ανεξαρτησία από ένα εκ των δύο μεγεθών. Αυτός μπορεί να είναι ένας λόγος για τον οποίο η συνάρτηση αξιολόγησης $score_1 = R^2$ μοντελοποιεί τον χρόνο εκτέλεσης μόνο με βάση τις προβλέψεις για τα FLOPS. Σχετικά με τις διαφορές των πινάκων 5.1 και 5.2 παρατηρούμε ορισμένα κοινά μοντέλα, ενώ και οι δύο συναρτήσεις αξιολόγησης συσχετίζουν το χρόνο εκτέλεσης μόνο με το πλήθος των FLOPS.

Η κατάσταση είναι διαφορετική στις παρατηρήσεις που λαμβάνουμε από το μηχάνημα 3 και, ως αποτέλεσμα, στα μοντέλα που παρουσιάζονται για αυτές. Στα σχήματα 5.4α′ και 5.3 παρατηρούμε χρόνους οι οποίοι δείχνουν μια πιο συγκεκριμένη τάση σε σύγκριση με αυτούς των clones. Αυτό αποτυπώνεται και στην καλή μοντελοποίηση που παρατηρούμε. Στις περιπτώσεις των πινάκων 5.3 και 5.4 παρατηρούμε πως αφενός έχουμε και στους δύο ως καλύτερο ένα μοντέλο που εξαρτάται μόνο από το πλήθος των FLOPS που εκτιμάται ότι θα εκτελέσει η εφαρμογή, αφετέρου αυτό είναι κάτι που δεν ισχύει για όλα τα υπόλοιπα μοντέλα. Και στους δύο πίνακες παρατηρούμε μοντέλα που εξαρτώνται από ένα άθροισμα των διαφόρων προβλέψεων, όπου έχουμε μικρές αλλαγές σε εκθέτες ή εκφράσεις αυτών ανά μοντέλο. Πιο συγκεκριμένα, οι δύο συναρτήσεις αξιολόγησης, δίνουν κοινή έκφραση για την εξάρτηση από το πλήθος των Bytes επικοινωνίας και τον αριθμό FLOPS σε όλα τα μοντέλα τους, γεγονός που μάλλον αναδεικνύει σωστή πρόβλεψη για τη σχέση μεταξύ του χρόνου εκτέλεσης και των μεγεθών αυτών. Τέλος, το γεγονός ότι ο χρόνος εκτέλεσης εξαρτάται από το άθροισμα αυτών των μετρικών, σε συνδυασμό με την ανάλυση που έγινε για κάθε μετρική στο προηγούμενο κεφάλαιο, μας οδηγεί στο συμπέρασμα ότι η εφαρμογή δεν θα απαιτεί υπέρογκο χρόνο εκτέλεσης ακόμα και σε μεγαλύτερης κλίμακας προβλήματα και συστήματα.

### 5.2.2 Kripke

Συνεχίζουμε σε αυτή την υπό-ενότητα με την παρουσίαση των αντίστοιχων αποτελεσμάτων για την εφαρμογή `Kripke`, ξεκινώντας με τα μοντέλα μας από την ουρά μηχανημάτων clones.




(α′) $score_1$


(β′) $score_2$

Σχήμα 5.5: *Τα 5 καλύτερα μοντέλα πρόβλεψης του χρόνου εκτέλεσης για την εφαρμογή* `Kripke` *στα μηχανήματα clones, με χρήση των 3.2 και 3.3*

| $score_1$ | Σχετικό Σφάλμα |
|---|---|
| $4.418 \cdot 10^{-9} \cdot ls^{0.75} \cdot log_2{}^2(ls)$ | 0.0228 |
| $1.44 \cdot 10^{-7} \cdot ls^{0.75} \cdot log_2(ls)$ | 0.1007 |
| $4.71 \cdot 10^{-6} \cdot ls^{0.75}$ | 0.1903 |
| $1.67 \cdot 10^{-8} \cdot ls$ | 0.1745 |
| $5.1368 \cdot 10^{-10} \cdot ls \cdot log_2(ls)$ | 0.2214 |

Πίνακας 5.5: *Τα 5 καλύτερα μοντέλα για χρόνο εκτέλεσης της εφαρμογής* `Kripke` *στα μηχανήματα clones με χρήση της* $score_1$

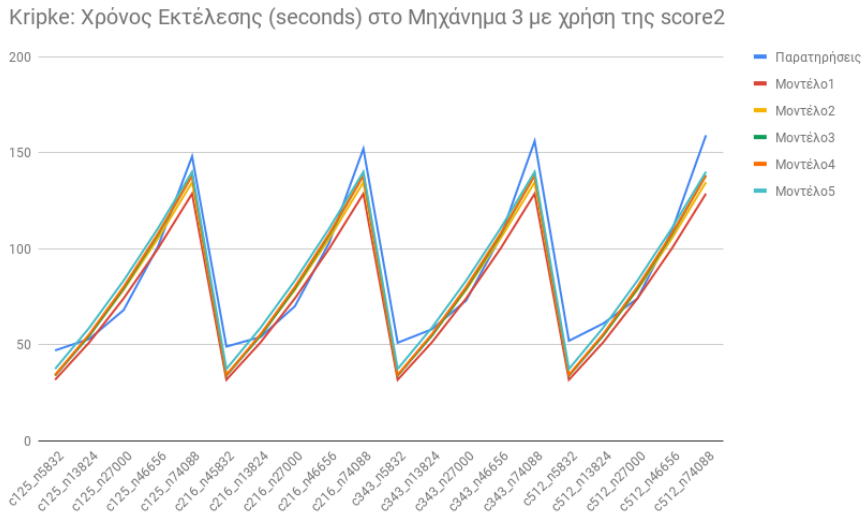
| $score_2$ | Σχετικό Σφάλμα |
|---|---|
| $3.58 \cdot 10^{-9} \cdot ls^{0.75} \cdot log_2{}^(ls)$ | 0.3026 |
| $3.58 \cdot 10^{-9} \cdot (sr^{0.25} \cdot log_2(sr) + f^{0.25} + b^{0.25} + ls^{0.75} \cdot log_2{}^(ls))$ | 0.3026 |
| $3.58 \cdot 10^{-9} \cdot (sr^{0.25} \cdot log_2(sr) + f^{0.25} + b^{0.25} \cdot log_2(b) + ls^{0.75} \cdot log_2{}^(ls))$ | 0.3026 |
| $3.58 \cdot 10^{-9} \cdot (sr^{0.25} \cdot log_2(sr) + f^{0.25} \cdot log_2(f) + b^{0.25} + ls^{0.75} \cdot log_2{}^(ls))$ | 0.3026 |
| $3.58 \cdot 10^{-9} \cdot (sr^{0.25} \cdot log_2(sr) + f^{0.25} \cdot log_2(f) + b^{0.25} \cdot log_2(b) + ls^{0.75} \cdot log_2{}^(ls))$ | 0.3026 |

Πίνακας 5.6: *Τα 5 καλύτερα μοντέλα για χρόνο εκτέλεσης της εφαρμογής* `Kripke` *στα μηχανήματα clones με χρήση της* $score_2$

Όπως αναφέρθηκε και στη σχετική ανάλυση της εφαρμογής του `LULESH`, τα μηχανήματα της ουράς clones διαθέτουν ορισμένα μειονεκτήματα τα οποία και καθιστούν τις μετρήσεις που λαμβάνουμε σε αυτά ακατάλληλες για την εξαγωγή ασφαλών αποτελεσμάτων σχετικά με μοντέλα πρόβλεψης συμπεριφοράς. Ωστόσο αξίζει να σχολιασθούν τα παραπάνω αποτελέσματα αλλά και να συγκριθούν με αυτά από το μηχάνημα 3 που ακολουθούν. Αρχικά, παρατηρούμε ότι με χρήση της συνάρτησης αξιολόγησης $score_1 = R^2$ λαμβάνουμε μοντέλα που εξαρτώνται μόνο από το πλήθος εντολών πρόσβασης στη μνήμη. Αυτό δικαιολογείται αν ανατρέξουμε στους αντίστοιχους πίνακες των μετρικών του `Kripke`, όπου και θα συνειδητοποιήσουμε ότι η μόνη μετρική που εξαρτάται κι από το μέγεθος προβλήματος n αλλά και από το πλήθος διαδικασιών p, είναι αυτή του αριθμού Loads & Stores. Για αυτό το λόγο, και μιας και είναι εμφανής η εξάρτηση των παρατηρήσεων από την παράμετρο p, παρατηρούμε εξάρτηση μόνο από το ls. Όσον αφορά τα μοντέλα του πίνακα 5.6 εδώ αν και παρατηρούμε εξάρτηση από όλες τις μετρικές, το πλήθος Loads & Stores μοιάζει κι εδώ να είναι το πιο σημαντικό, αφού οι υπόλοιπες μετρικές εμφανίζονται με σχετικά μικρούς εκθέτες.

Στη συνέχεια ακολουθούν τα αντίστοιχα μοντέλα που εξάγαμε στο μηχάνημα 3


(α′) $score_1$




(β′) $score_2$

Σχήμα 5.6: *Τα 5 καλύτερα μοντέλα πρόβλεψης του χρόνου εκτέλεσης για την εφαρμογή* **`Kripke`** *στο μηχάνημα 3, με χρήση των 3.2 και 3.3*

| $score_1$ | Σχετικό Σφάλμα |
|---|---|
| $9.82 \cdot 10^{-24} \cdot b^{2.75}$ | 0.6219 |
| $3.237 \cdot 10^{-25} \cdot b^{2.75} \cdot log_2(b)$ | 0.6283 |
| $1.056 \cdot 10^{-26} \cdot (sr^3 + f^{1.5} \cdot {log_2}^2(f) + b^{2.75} \cdot {log_2}^2(b) + ls^{1.25} \cdot {log_2}^2(ls))$ | 0.6172 |
| $1.056 \cdot 10^{-26} \cdot (sr^3 + f^{1.5} \cdot {log_2}^2(f) + b^{2.75} \cdot {log_2}^2(b) + ls^{1.25} \cdot log_2(ls))$ | 0.6172 |
| $1.056 \cdot 10^{-26} \cdot (sr^3 + f^{1.5} \cdot {log_2}^2(f) + b^{2.75} \cdot {log_2}^2(b) + ls \cdot {log_2}^2(ls))$ | 0.6172 |

Πίνακας 5.7: *Τα 5 καλύτερα μοντέλα για χρόνο εκτέλεσης της εφαρμογής* **`Kripke`** *στο μηχάνημα 3 με χρήση της* $score_1$

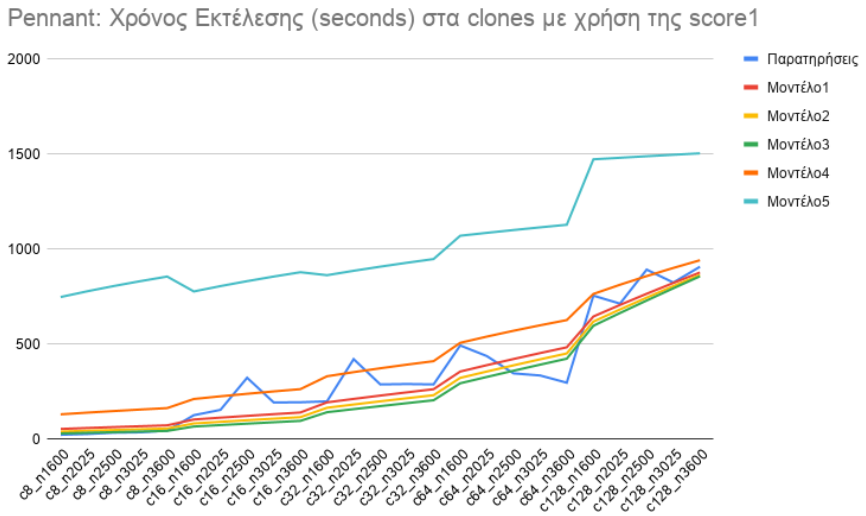
| $score_2$ | Σχετικό Σφάλμα |
|---|---|
| $3.14 \cdot 10^{-4} \cdot ls^{0.5}$ | 0.1238 |
| $4.415 \cdot 10^{-6} \cdot (sr^{0.75} + f^{0.5} \cdot log_2(f) + b^{0.5} \cdot log_2(b) + ls^{0.5} \cdot log_2(ls))$ | 0.0704 |
| $5.27 \cdot 10^{-6} \cdot (sr^{0.75} + f^{0.5} \cdot log_2(f) + b^{0.75} + ls^{0.5})$ | 0.0618 |
| $5.37 \cdot 10^{-7} \cdot (sr^{0.75} \cdot log_2(sr) + f^{0.5} \cdot log_2(f) + b^{0.5} \cdot {log_2}^2(b) + ls^{0.5} \cdot log_2(ls))$ | 0.0548 |
| $1.72 \cdot 10^{-6} \cdot (sr^{0.75} + f^{0.5} \cdot log_2(f) + b^{0.5} \cdot {log_2}^2(b) + ls^{0.5} \cdot log_2(ls))$ | 0.013 |

Πίνακας 5.8: *Τα 5 καλύτερα μοντέλα για χρόνο εκτέλεσης της εφαρμογής* `Kripke` *στο μηχάνημα 3 με χρήση της* $score_2$

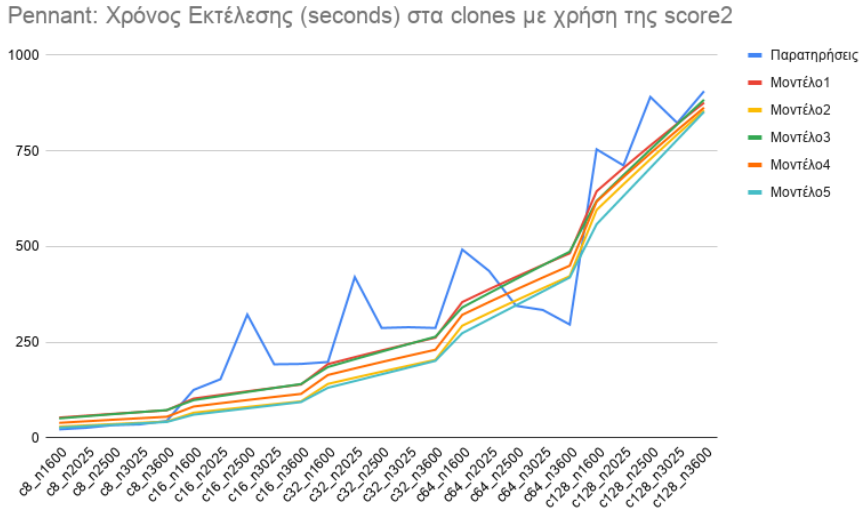
Σχετικά με τις μετρήσεις από το μηχάνημα 3 παρατηρούμε ότι σε αυτή την περίπτωση ο χρόνος εκτέλεσης φαίνεται να εξαρτάται περισσότερο από το μέγεθος του προβλήματος n σε σύγκριση με τα σχήματα που παρουσιάσθηκαν για τα μηχανήματα clones. Αυτό αποτυπώνεται και στα μοντέλα των πινάκων 5.7 και 5.8, όπου παρατηρούμε εξάρτηση του χρόνου εκτέλεσης από όλες τις μετρικές. Πιο συγκεκριμένα για τη συνάρτηση αξιολόγησης $score_1 = R^2$ παρατηρούμε ότι τα μοντέλα 3-5 εξαρτώνται από ένα άθροισμα όλων των μετρικών, με μικρές διαφορές ανάμεσά τους. Είναι ακόμη εμφανές ότι με αυτή τη μέθοδο αξιολόγησης λαμβάνουμε μοντέλα με υψηλούς εκθέτες, κάτι το οποίο θα μας οδηγούσε στο συμπέρασμα ότι η εφαρμογή δεν θα κλιμακώνει με βέλτιστο τρόπο σε ό,τι αφορά τον χρόνο εκτέλεσης, κάτι το οποίο δεν δικαιολογείται από τη σχηματική αναπαράσταση των μετρήσεων. Αντιθέτως, στον πίνακα 5.8 παρατηρούμε κι εδώ εξάρτηση από άθροισμα των μετρικών στα τέσσερα από τα πέντε μοντέλα, χωρίς ωστόσο να έχουμε υψηλές τιμές εκθετών. Για αυτό το λόγο αναμένουμε καλή κλιμάκωση του χρόνου εκτέλεσης της εφαρμογής σε μεγαλύτερα προβλήματα και ισχυρότερα μηχανήματα, μιας και το σχήμα 5.6β′ αναδεικνύει και την επιθυμητή προσαρμογή των προβλέψεων στις αντίστοιχες μετρήσεις.

### 5.2.3 Pennant

Στη συνέχεια θα παρουσιάσουμε τα αντίστοιχα αποτελέσματα μας για την εφαρμογή `Pennant` στα μηχανήματα της ουράς clones.




(α′) $score_1$




(β′) $score_2$

Σχήμα 5.7: *Τα 5 καλύτερα μοντέλα πρόβλεψης του χρόνου εκτέλεσης για την εφαρμογή* `Pennant` *στα μηχανήματα clones, με χρήση των 3.2 και 3.3*

| $score_1$ | Σχετικό Σφάλμα |
|---|---|
| $1.17 \cdot 10^{-2} \cdot sr^{0.5}$ | 0.1158 |
| $3.56 \cdot 10^{-4} \cdot sr^{0.5} \cdot log_2(sr)$ | 0.0538 |
| $1.09 \cdot 10^{-5} \cdot sr^{0.5} \cdot {log_2}^2(sr)$ | 0.1842 |
| $3.277 \cdot 10^{-3} \cdot sr^{0.25} \cdot {log_2}^2(sr)$ | 0.999 |
| $3.54 \cdot 10^{-3} \cdot ls^{0.25} \cdot {log_2}^2(ls)$ | 6.7868 |

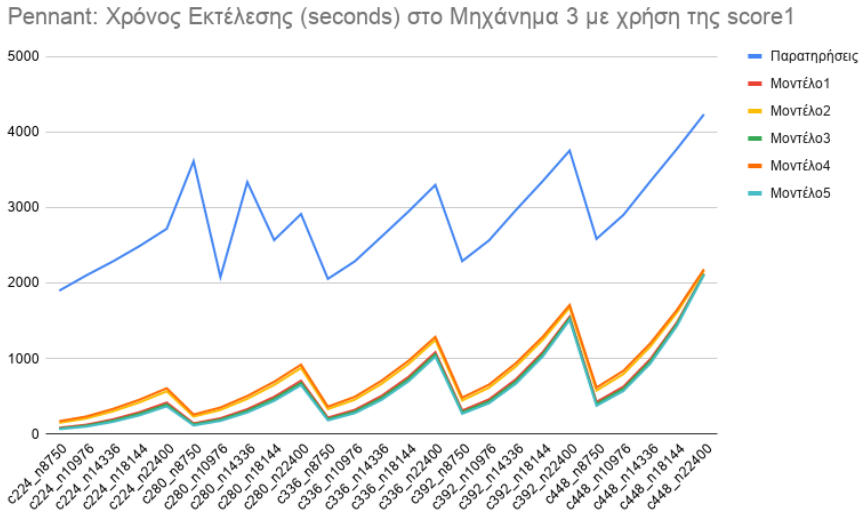
Πίνακας 5.9: *Τα 5 καλύτερα μοντέλα για χρόνο εκτέλεσης της εφαρμογής* `Pennant` *στα μηχανήματα clones με χρήση της* $score_1$

| $score_2$ | Σχετικό Σφάλμα |
|---|---|
| $1.17 \cdot 10^{-2} \cdot sr^{0.5}$ | 0.1158 |
| $1.09 \cdot 10^{-5} \cdot sr^{0.5} \cdot {log_2}^2(sr)$ | 0.1842 |
| $1.7 \cdot 10^{-2} \cdot b^{0.5}$ | 0.0999 |
| $3.56 \cdot 10^{-4} \cdot sr^{0.5} \cdot log_2(sr)$ | 0.0538 |
| $1.67 \cdot 10^{-5} \cdot b^{0.5} \cdot {log_2}^2(b)$ | 0.2174 |

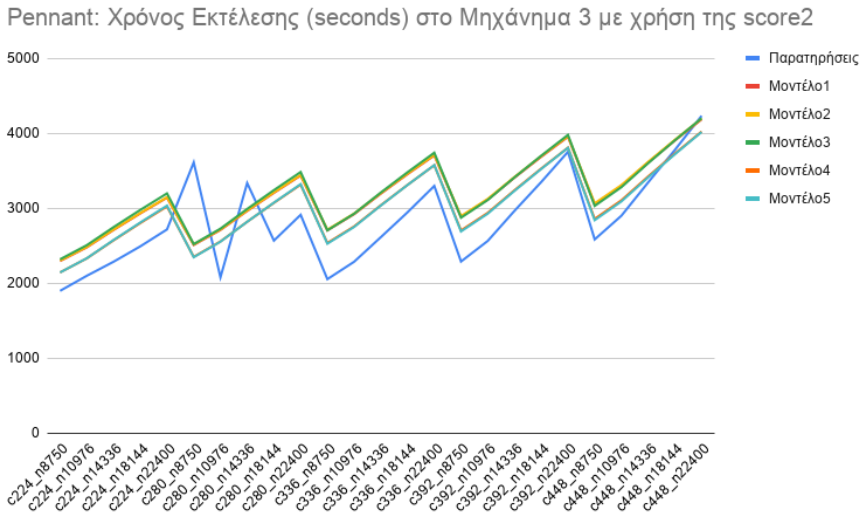
Πίνακας 5.10: *Τα 5 καλύτερα μοντέλα για χρόνο εκτέλεσης της εφαρμογής* `Pennant` *στα μηχανήματα clones με χρήση της* $score_2$

Παρατηρώντας τα σχήματα 5.7 γίνεται και σε αυτή την περίπτωση εμφανής η αδυναμία μοντελοποίησης του χρόνου εκτέλεσης στα clones, κάτι που μαρτυρούν οι ακανόνιστες "κορυφές" της καμπύλης των παρατηρήσεων, τις οποίες όλα τα μοντέλα μας αγνοούν. Παρατηρούμε επίσης ότι σχεδόν όλα τα μοντέλα συγκλίνουν με τις παρατηρήσεις για c128. Οι πίνακες 5.9 και 5.10 παρουσιάζουν αρκετά κοινά μοντέλα, ενώ σε όλες τις περιπτώσεις η μέθοδος πρόβλεψής μας δίνει προβλέψεις που εξαρτώνται από μοναδική μετρική, τις περισσότερες φορές αυτή του αριθμού Bytes επικοινωνίας.

Παρακάτω παρουσιάζουμε τα αποτελέσματά μας για την ίδια εφαρμογή στο μηχάνημα 3:




(α′) $score_1$




(β′) $score_2$

Σχήμα 5.8: *Τα 5 καλύτερα μοντέλα πρόβλεψης του χρόνου εκτέλεσης για την εφαρμογή* `Pennant` *στο μηχάνημα 3, με χρήση των 3.2 και 3.3*

| $score_1$ | Σχετικό Σφάλμα |
|---|---|
| $1.69 \cdot 10^{-9} \cdot b$ | 0.799 |
| $1.13 \cdot 10^{-9} \cdot b^{0.75} \cdot log_2{}^2(b)$ | 0.7504 |
| $4.2 \cdot 10^{-11} \cdot b \cdot log_2(b)$ | 0.807 |
| $4.58 \cdot 10^{-8} \cdot b^{0.75} \cdot log_2(b)$ | 0.7371 |
| $1.04 \cdot 10^{-12} \cdot b \cdot log_2{}^2(b)$ | 0.8151 |

Πίνακας 5.11: *Τα 5 καλύτερα μοντέλα για χρόνο εκτέλεσης της εφαρμογής* `Pennant` *στο μηχάνημα 3 με χρήση της* $score_1$

| $score_2$ | Σχετικό Σφάλμα |
|---|---|
| $2.74 \cdot 10^{-3} \cdot (sr^{0.5} + f^{0.5} + b^{0.25} + ls^{0.25} \cdot log_2{}^2(ls))$ | 0.1413 |
| $2.71 \cdot 10^{-3} \cdot (sr^{0}.5 + f^{0.5} + b^{0.25} \cdot log_2(b) + ls^{0.25} \cdot log_2{}^2(ls))$ | 0.1442 |
| $2.87 \cdot 10^{-3} \cdot (sr^{0.25} \cdot log_2(sr) + f^{0.5} + b^{0.5} + ls^{0.25} \cdot log_2{}^2(ls))$ | 0.1474 |
| $2.81 \cdot 10^{-4} \cdot (sr^{0.25} \cdot log_2{}^2(sr) + f^{0.5} \cdot log_2(f) + b^{0.25} \cdot log_2{}^2(b) + ls^{0.25} \cdot log_2{}^2(ls))$ | 0.084 |
| $2.96 \cdot 10^{-4} \cdot (sr^{0.25} \cdot log_2{}^2(sr) + f^{0.5} \cdot log_2(f) + b^{0.25} \cdot log_2(b) + ls^{0.25} \cdot log_2{}^2(ls))$ | 0.0827 |

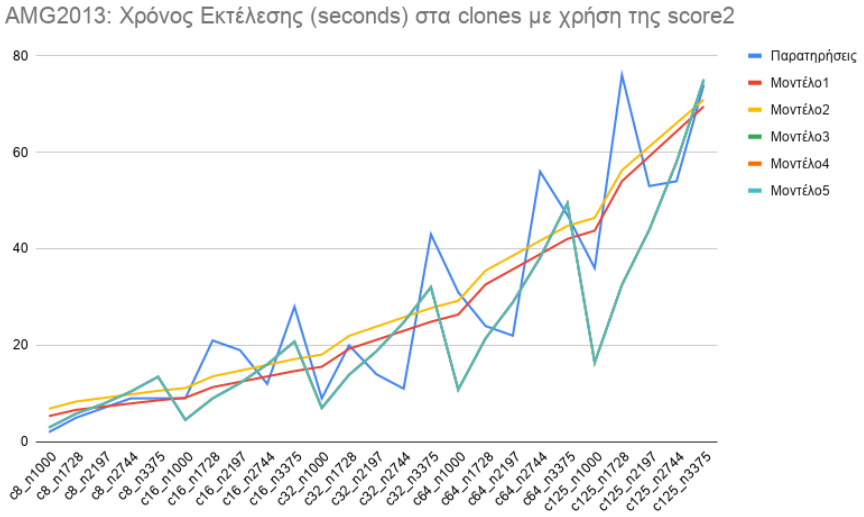
Πίνακας 5.12: *Τα 5 καλύτερα μοντέλα για χρόνο εκτέλεσης της εφαρμογής* `Pennant` *στο μηχάνημα 3 με χρήση της* $score_2$

Ξεκινώντας με τις παρατηρήσεις είναι εμφανές ότι στις μετρήσεις μας για c280 υπάρχουν κάποιες μη αναμενόμενες κορυφές, ωστόσο σε μεγαλύτερο αριθμό πυρήνων ($c \geq 336$) παρατηρούμε μια πιο ξεκάθαρη τάση στις μετρήσεις. Σχετικά με τα μοντέλα του πίνακα 5.11, τα οποία λαμβάνουμε με τη $score_1 = R^2$, έχουμε προβλέψεις που εξαρτώνται μονάχα από το πλήθος Bytes που καταλαμβάνονται στη μνήμη. Οι προβλέψεις αυτού του πίνακα δείχνουν καλή προσαρμογή στην τάση των δεδομένων, έχουν όμως πολύ υψηλές τιμές λάθους εκτίμησης, κάτι που γίνεται εύκολα αντιληπτό και γραφικά. Όσον αφορά τις προβλέψεις του πίνακα 5.12, εδώ έχουμε εξάρτηση από το άθροισμα όλων των μετρικών. Παρατηρούμε και σε αυτή την περίπτωση καλή προσαρμογή στην τάση των μετρήσεων, ειδικά για $c \geq 336$, αλλά έχουμε και μικρές τιμές λαθών (πιο συγκεκριμένα όλα τα μοντέλα δίνουν $pred_{<0.35} = 1$). Λόγω των χαμηλών εκθετών των μετρικών, αλλά και του ότι ο χρόνος εξαρτάται από άθροισμα αυτών, αναμένουμε η εφαρμογή `Pennant` να κλιμακώνει ικανοποιητικά όσον αφορά χρονικές απαιτήσεις σε συστήματα μεγαλύτερης κλίμακας.

### 5.2.4 AMG2013

Τέλος θα παρουσιάσουμε τα αντίστοιχα αποτελέσματα μας για την εφαρμογή `AMG2013` στα μηχανήματα της ουράς clones.


(α′) $score_1$




(β′) $score_2$

Σχήμα 5.9: *Τα 5 καλύτερα μοντέλα πρόβλεψης του χρόνου εκτέλεσης για την εφαρμογή* `AMG2013` *στα μηχανήματα clones, με χρήση των 3.2 και 3.3*

| $score_1$ | Σχετικό Σφάλμα |
|---|---|
| $8.25 \cdot 10^{-7} \cdot (sr^{0.5} \cdot log_2{}^2(sr) + f^{0.5} \cdot log_2{}^2(f) + b^{0.5} \cdot log_2{}^2(b) + ls^{0.75} \cdot log_2(ls))$ | 0.5772 |
| $8.26 \cdot 10^{-7} \cdot (sr^{0.5} \cdot log_2{}^2(sr) + f^{0.75} + b^{0.5} \cdot log_2{}^2(b) + ls^{0.75} \cdot log_2(ls))$ | 0.5776 |
| $8.26 \cdot 10^{-7} \cdot (sr^{0.5} \cdot log_2{}^2(sr) + f^{0.5} \cdot log_2(f) + b^{0.5} \cdot log_2{}^2(b) + ls^{0.75} \cdot log_2(ls))$ | 0.5776 |
| $8.26 \cdot 10^{-7} \cdot (sr^{0.5} \cdot log_2{}^2(sr) + f^{0.25} \cdot log_2{}^2(f) + b^{0.5} \cdot log_2{}^2(b) + ls^{0.75} \cdot log_2(ls))$ | 0.5776 |
| $8.26 \cdot 10^{-7} \cdot (sr^{0.5} \cdot log_2{}^2(sr) + f^{0.5} + b^{0.5} \cdot log_2{}^2(b) + ls^{0.75} \cdot log_2(ls)))$ | 0.5776 |

Πίνακας 5.13: *Τα 5 καλύτερα μοντέλα για χρόνο εκτέλεσης της εφαρμογής* `AMG2013` *στα μηχανήματα clones με χρήση της* $score_1$

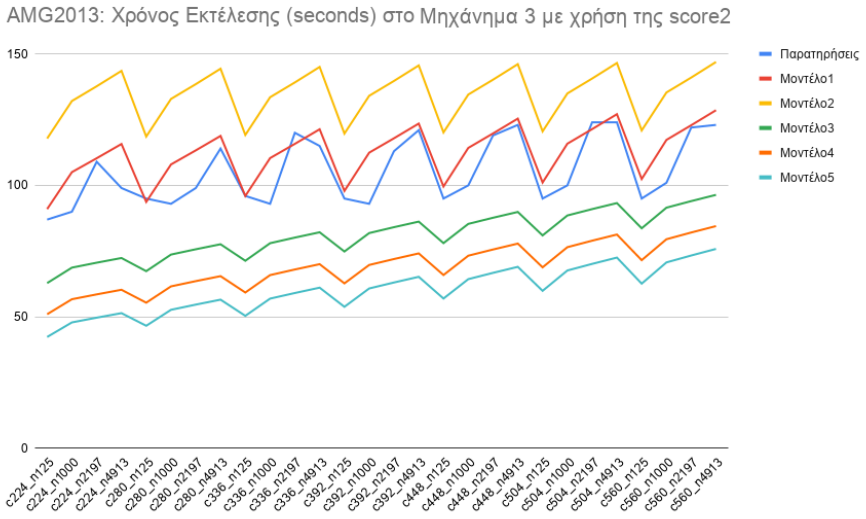
| $score_2$ | Σχετικό Σφάλμα |
|---|---|
| $3.296 \cdot 10^{-6} \cdot sr^{0.5} \cdot log_2{}^2(sr)$ | 0.3526 |
| $9.82 \cdot 10^{-5} \cdot sr^{0.5} \cdot log_2(sr)$ | 0.4507 |
| $1.076 \cdot 10^{-19} \cdot (sr^{0.75} \cdot log_2{}^2(sr) + f^{0.75} + b^{0.75} \cdot log_2(b) + ls^{2.5})$ | 0.3452 |
| $1.076 \cdot 10^{-19} \cdot (sr^{0.75} \cdot log_2{}^2(sr) + f^{0.75} \cdot log_2{}^2(f) + b^{0.75} \cdot log_2(b) + ls^{2.5})$ | 0.3452 |
| $1.076 \cdot 10^{-19} \cdot (sr^{0.75} \cdot log_2{}^2(sr) + f \cdot log_2(f) + b^{0.75} \cdot log_2(b) + ls^{2.5})$ | 0.3452 |

Πίνακας 5.14: *Τα 5 καλύτερα μοντέλα για χρόνο εκτέλεσης της εφαρμογής* `AMG2013` *στα μηχανήματα clones με χρήση της* $score_2$

Σχολιάζοντας τα σχήματα 5.9 είναι εμφανές ότι οι παρατηρήσεις που λαμβάνουμε στα clones είναι προβληματικές λόγω των πολλών και απρόσμενων "κορυφών" που εμφανίζονται. Για αυτό το λόγο παρατηρούμε πως τα μοντέλα του 5.13 φαίνεται να αγνοούν τις εν λόγω κορυφές κι αντ΄ αυτού να επιλέγουν να μοντελοποιήσουν τις μετρήσεις παρουσιάζοντας μια γραμμή τάσης αυτών. Αυτό έχει σαν αποτέλεσμα και υψηλότερες τιμές σφαλμάτων, κυρίως λόγω των σημείων όπου εμφανίζουν κορυφές οι παρατηρήσεις. Επιπλέον, ο πίνακας 5.13 παρουσιάζει μοντέλα τα οποία εξαρτώνται από το άθροισμα όλων των μετρικών. Σχετικά με τις προβλέψεις που λαμβάνουμε με χρήση της $score_2$, παρατηρούμε ότι τα δύο "καλύτερα" μοντέλα εξαρτώνται μόνο από την ανάγκη της εφαρμογής για επικοινωνία κι ακολουθούν παρόμοια οδό με αυτά της $score_1$, παρουσιάζοντας μια γραμμή τάσης των μετρήσεων. Αντιθέτως, τα μοντέλα 3,4,5, τα οποία κι εμφανίζουν εξάρτηση από όλες τις μετρικές, επιχειρούν να μοντελοποιήσουν ακόμη και τις "απρόβλεπτες" κορυφές που αναφέραμε προηγουμένως, κάτι που οδηγεί και σε μικρότερο σχετικό σφάλμα.

Παρακάτω παρουσιάζουμε τα αποτελέσματά μας για την ίδια εφαρμογή στο μηχάνημα 3:


(α′) $score_1$




(β′) $score_2$

Σχήμα 5.10: *Τα 5 καλύτερα μοντέλα πρόβλεψης του χρόνου εκτέλεσης για την εφαρμογή* ***AMG2013*** *στο μηχάνημα 3, με χρήση των 3.2 και 3.3*

| $score_1$ | Σχετικό Σφάλμα |
|---|---|
| $7.93 \cdot 10^{-7} \cdot (sr^{0.75} + log_2(f) + b^{0.75} + ls^{0.75})$ | 0.8379 |
| $7.93 \cdot 10^{-7} \cdot (sr^{0.75} + f^{0.25} + b^{0.75} + ls^{0.75})$ | 0.8379 |
| $7.93 \cdot 10^{-7} \cdot (sr^{0.75} + {log_2}^2(f) + b^{0.75} + ls^{0.75})$ | 0.8379 |
| $7.93 \cdot 10^{-7} \cdot (sr^{0.75} + f^{0.25} \cdot log_2(f) + b^{0.75} + ls^{0.75})$ | 0.8379 |
| $7.93 \cdot 10^{-7} \cdot (sr^{0.75} + f^{0.25} \cdot {log_2}^2(f) + b^{0.75} + ls^{0.75})$ | 0.8379 |

Πίνακας 5.15: *Τα 5 καλύτερα μοντέλα για χρόνο εκτέλεσης της εφαρμογής* ***AMG2013*** *στο μηχάνημα 3 με χρήση της* $score_1$

| $score_2$ | Σχετικό Σφάλμα |
|---|---|
| $1.24 \cdot 10^{-1} \cdot {log_2}^2(sr)$ | 0.0755 |
| $1.8 \cdot 10^{-1} \cdot {log_2}^2(ls)$ | 0.2853 |
| $3.07 \cdot 10^{-1} \cdot b^{0.25}$ | 0.2281 |
| $8.12 \cdot 10^{-3} \cdot b^{0.25} \cdot log_2(b)$ | 0.344 |
| $2.2 \cdot 10^{-4} \cdot b^{0.25} \cdot {log_2}^2(b)$ | 0.4292 |

Πίνακας 5.16: *Τα 5 καλύτερα μοντέλα για χρόνο εκτέλεσης της εφαρμογής* `AMG2013` *στο μηχάνημα 3 με χρήση της* $score_2$

Σχετικά με τις παρατηρήσεις στα 5.10 είναι εμφανές ότι οι μετρήσεις στο μηχάνημα 3 είναι σαφώς καλύτερες από αυτές που λαμβάνουμε στα clones. Όσον αφορά τη συνάρτηση αξιολόγησης $R^2$ παρατηρούμε ότι, αν και τα μοντέλα εμφανίζουν ικανοποιητική προσαρμογή, προβλέποντας ορθώς τις κορυφές και την τάση των μετρήσεων, έχουμε αρκετά υψηλό σφάλμα. Αυτό αποτυπώνεται και στον πίνακα 5.15 όπου και παρατηρούμε ότι και τα πέντε τα μοντέλα εξαρτώνται από το άθροισμα όλων των μετρικών, με μικρές αποκλίσεις στους εκάστοτε εκθέτες. Αντίθετα, στον πίνακα 5.16 παρατηρούμε ότι κανένα από τα πέντε καλύτερα μοντέλα δεν εξαρτάται από όλες τις μετρικές, παρά μόνο από μία τη φορά. Ωστόσο, διαφαίνονται πολύ καλή προσαρμογή στις μετρήσεις κι αισθητά μικρότερα σφάλματα σε σύγκριση με τα μοντέλα της $score_1$, κάτι φυσικά αναμενόμενο αφού το σφάλμα είναι κάτι που λαμβάνει υπόψιν η $score_2$ για την επιλογή των προβλέψεων. Τέλος αξίζει να σχολιάσουμε ότι τόσο η εξάρτηση από άθροισμα όλων των μετρικών που μας δείχνει ο πίνακας 5.15 όσο και η εξάρτηση από μία μετρική τη φορά που παρατηρούμε στον 5.16 μαρτυρούν ότι ο χρόνος εκτέλεσης της εφαρμογής δεν αναμένεται να δημιουργήσει κώλυμα στην εφαρμογή σε μεγαλύτερα συστήματα και προβλήματα, γεγονός που μοιάζει να επιβεβαιώνεται και γραφικά στα σχήματα 5.10.

Κεφάλαιο 6

# Επιλογή μοντέλων και "συν-σχεδίαση"

Αφού παρουσιάσαμε τα εμπειρικά μας μοντέλα τόσο για τις ανάγκες σε πόρους των τεσσάρων εφαρμογών μας όσο και για το χρόνο εκτέλεσης αυτών, θα εξερευνήσουμε, σε αυτό το κεφάλαιο, ποια είναι τα κριτήρια που θα μπορούσαν να βοηθήσουν στην επιλογή ενός από τα πέντε καλύτερα μοντέλα κάθε μετρικής αλλά και το με ποιο τρόπο θα μπορούσαμε να χρησιμοποιήσουμε αυτά τα μοντέλα σε μία άσκηση συν-σχεδίασης.

## 6.1 Επιλογή μοντέλων

Όπως προαναφέραμε στα προηγούμενα κεφάλαια παρουσιάσαμε εμπειρικά μοντέλα για κάθε μετρική, καταταγμένα σε πεντάδες με δύο διαφορετικές συναρτήσεις αξιολόγησης. Η επιλογή να συμπεριλάβουμε τα πέντε καλύτερα μοντέλα που λαμβάναμε με τη μέθοδο μας κι όχι απλώς το καλύτερο έγινε καθώς, μερικές φορές, η κατάταξη των μοντέλων δεν καταλήγει με το επιθυμητό αποτέλεσμα. Αυτό είναι κάτι που γίνεται αντιληπτό και γραφικά από τα σχήματα των προηγουμένων ενοτήτων. Επομένως, είναι ιδιαίτερης σημασίας να παρουσιάσουμε, σε αυτό το σημείο, τρόπους με τους οποίους θα μπορούσαμε να καταλήξουμε στην επιλογή ενός από τα μοντέλα που έχουμε κάθε φορά διαθέσιμα.

Αρχικά, η πιο άμεση μέθοδος επιλογής θα ήταν να επιλέξουμε το πρώτο μοντέλο σύμφωνα με την κατάταξη που προέκυψε από τις συναρτήσεις αξιολόγησης. Με αυτό τον τρόπο η επιλογή γίνεται με βάση τις μετρικές που λαμβάνει υπόψιν της η συνάρτηση και τα ειδικά χαρακτηριστικά καθεμίας εξ αυτών, όπως περιγράφηκαν στην ενότητα 3.3.2. Ωστόσο, όπως παρατηρήσαμε και στα αποτελέσματα που παρουσιάστηκαν, μερικές φορές η κατάταξη των μοντέλων που λαμβάνουμε δεν καταλήγει με το καλύτερο στην κορυφή.

Μια μέθοδος εκλογής του επιθυμητού μοντέλου από τα αποτελέσματά μας θα μπορούσε να είναι μέσω της γραφικής αναπαράστασης. Παρατηρώντας δηλαδή τις μετρήσεις και τις προβλέψεις που παράγονται για αυτές, μπορούμε, αφού έχει γίνει μια διαλογή ενός συνόλου μοντέλων μέσω των συναρτήσεων κατάταξη, να αποφανθούμε ως προς το ποιο μοντέλο διαθέτει καλύτερη προσαρμογή στις παρατηρήσεις γραφικά. Ακόμη, μια τέτοια μέθοδος, μπορεί να αποδειχθεί ευεργετική στην προβολή των προβλέψεων σε μεγαλύτερα προβλήματα ή συστήματα, λαμβάνοντας υπόψιν την ικανοποιητική, ή όχι, προσομοίωση των παρατηρήσεων στις μεγαλύτερες παρατηρήσεις.

Άλλη μέθοδος εκλογής καλύτερου μοντέλου είναι η επιλογή με βάση το μικρότερο σχετικό σφάλμα. Θυμίζουμε εδώ ότι η συνάρτηση αξιολόγησης $score_2$ 3.3 λαμβάνει υπόψιν το σχετικό

σφάλμα, αλλά επηρεάζεται από το ποσοστό των παρατηρήσεων που βρίσκονται σε απόσταση ±35% από τις πραγματικές παρατηρήσεις, ενώ ακόμη δεν εξαρτάται μόνο από αυτή τη μετρική. Για αυτό το λόγο συχνά το πρώτο μοντέλο με βάση αυτή τη μετρική δεν διαθέτει απαραίτητα το μικρότερο σχετικό σφάλμα συνολικά. Επιπλέον η πρώτη συνάρτηση αξιολόγησης $score_1$ 3.2 δεν λαμβάνει καθόλου υπόψιν το σφάλμα των προβλέψεων, κι άρα μια επιλογή των μοντέλων της με βάση αυτή τη μετρική θα ήταν θεμιτή.

Τέλος, σε μερικές κατατάξεις μοντέλων που παρουσιάσαμε δύναται να συνυπάρχουν προβλέψεις που εξαρτώνται από ένα μέρος του συνόλου παραμέτρων κι όχι από όλες, π.χ. μόνο από το πλήθος διαδικασιών p. Είναι πιθανό να γνωρίζουμε, λόγω διαίσθησης ή επίγνωσης της λειτουργίας του κώδικα, ότι η μετρική θα έπρεπε να εξαρτάται από όλες τις παραμέτρους ή από διαφορετικό μέρος αυτών. Σε αυτή την περίπτωση είναι θεμιτό να αλλάξουμε την επιλογή μας για το καλύτερο μοντέλο, λαμβάνοντας φυσικά υπόψιν και τα υπόλοιπα κριτήρια που παρατάθηκαν ανωτέρω.

## 6.2 Άσκηση συν-σχεδίασης

Στο σημείο αυτό θα προχωρήσουμε σε μια μελέτη συν-σχεδίασης [5] με τις τέσσερις εφαρμογές που παρουσιάστηκαν στις προηγούμενες ενότητες. Σε αυτή τη μελέτη θα συγκρίνουμε τα πλεονεκτήματα τριών διαφορετικών αναβαθμίσεων ενός υποθετικού συστήματος για τις ανάγκες της κάθε εφαρμογής, αν αυτή χρησιμοποιεί όλους τους διαθέσιμους πόρους. Στόχος είναι χρησιμοποιώντας τα μοντέλα που έχουμε εξάγει για τις ανάγκες των εφαρμογών, να υπολογίσουμε τα αντίστοιχα μοντέλα μετά την εκάστοτε αναβάθμιση και να καταλήξουμε σε έναν λόγο μεταξύ των δύο μοντέλων, συγκρίνοντας τις ανάγκες στα δύο συστήματα. Οι τρεις πιθανές αναβαθμίσεις που θα λάβουμε υπόψιν είναι:

1. διπλασιασμός όλου του συστήματος
2. διπλασιασμός του αριθμού των επεξεργαστών ανά κόμβο, αφήνοντας τις υπόλοιπες παραμέτρους αναλλοίωτες
3. διπλασιασμός της μνήμης, αφήνοντας τις υπόλοιπες παραμέτρους αναλλοίωτες.

Συγκεντρωτικά μπορούμε να δούμε στον παρακάτω πίνακα την πρακτική μετάφραση καθεμίας εκ των ανωτέρω αναβαθμίσεων:

| Αναβάθμιση Συστήματος | Πλήθος Διαδικασιών | Μνήμη ανά διαδικασία |
|---|---|---|
| Α: Διπλασιασμός Συστήματος | $p' = 2 \cdot p$ | $m' = m$ |
| Β: Διπλασιασμός Αριθμού Επεξεργαστών | $p' = 2 \cdot p$ | $m' = 0.5 \cdot m$ |
| Γ: Διπλασιασμός Μνήμης | $p' = p$ | $m' = 2 \cdot m$ |

Πίνακας 6.1: *Αναβαθμίσεις συστήματος κι αντίκτυπος αυτών σε διαδικασίες και μνήμη*

Στη συνέχεια θα εξηγήσουμε τη μεθοδολογία που ακολουθείται προκειμένου να καταλήξουμε στους λόγους που παρουσιάζονται στον πίνακα 6.3 κι εκφράζουν την αλλαγή στις ανάγκες των εφαρμογών ανά διαφορετική αναβάθμιση.

Αρχικά χρειάζεται να γίνει μια επιλογή ενός μοντέλου από τα καλύτερα πέντε που παρουσιάσαμε για κάθε μετρική. Η επιλογή αυτή γίνεται με βάση τα κριτήρια που αναφέρθηκαν

στην προηγούμενη ενότητα. Ως επί το πλείστον κινηθήκαμε με βάση τη γραφική απεικόνιση προσαρμογής στις παρατηρήσεις, εκτός από περιπτώσεις όπου όλα τα μοντέλα επεδείκνυαν καλή προσαρμογή, οπότε και χρησιμοποιήθηκαν και τα άλλα κριτήρια. Αποφασίζουμε ακόμη να παρουσιάσουμε τη μελέτη μας μόνο για τα μοντέλα που προέκυψαν με χρήση της συνδυαστικής συνάρτησης αξιολόγησης, $score_2$, αφού παρατηρήσαμε ότι αυτή επιδεικνύει πιο καλά προσαρμοσμένα αποτελέσματα.

Παρακάτω μπορούμε να δούμε συγκεντρωτικά και σε ευκρινή μορφή τα βήματα που ακολουθούνται για την άσκηση:

1. Δημιουργούμε τα εμπειρικά μοντέλα για τις ανάγκες των εφαρμογών σε υπολογισιμότητα, επικοινωνία, αποτύπωμα μνήμης, πρόσβαση στη μνήμη και χρόνο εκτέλεσης. Επιλογή ενός εκ των πέντε κορυφαίων με βάση τα κριτήρια της ενότητας 6.1.

2. Προσδιορισμός του νέου μέγιστου αριθμού διαδικασιών και της νέας μνήμης ανά διαδικασία που υποστηρίζονται από το αναβαθμισμένο σύστημα, σύμφωνα με τον πίνακα 6.1.

3. Εύρεση του νέου αποτυπώματος στη μνήμη ανά διαδικασία για την εφαρμογή ενδιαφέροντος, αν χρησιμοποιηθούν όλοι οι επεξεργαστές του νέου συστήματος.

4. Υπολογισμός του νέου μεγέθους προβλήματος ανά διαδικασία και του νέου συνολικού μεγέθους προβλήματος. Το νέο μέγεθος προβλήματος ανά διαδικασία $n'$, υπολογίζεται με βάση την παραδοχή ότι το αποτύπωμα μνήμης ισούται με τη μνήμη που υπάρχει διαθέσιμη ανά διαδικασία. Αντίστοιχα, το συνολικό μέγεθος προβλήματος είναι το μέγεθος προβλήματος ανά διαδικασία επί το πλήθος διαδικασιών.

5. Προσδιορισμός των νέων αναγκών για υπολογισιμότητα, επικοινωνία και πρόσβαση στη μνήμη, με τη βοήθεια των $p', n'$ που υπολογίσθηκαν.

6. Εύρεση του νέου χρόνου εκτέλεσης με τη βοήθεια των νέων αναγκών που υπολογίσθηκαν στα βήματα 3,5, αντικαθιστώντας τις ανάγκες με τα μοντέλα αυτών σε συνάρτηση των $(n, p)$.

Για την καλύτερη κατανόηση των παραπάνω θα παρουσιάσουμε βήμα προς βήμα την άσκηση για την εφαρμογή Pennant, επιλέγοντας την αναβάθμιση Α: διπλασιασμός συστήματος.

1. Επιλέγουμε τα παρακάτω μοντέλα για κάθε μετρική:

| Μετρική | Μοντέλο |
|---|---|
| FLOPS | $p^{0.75} \cdot n^{0.5} \cdot log_2{}^2 n$ |
| Bytes στη μνήμη | $p^{1.5} \cdot log_2 p \cdot n^{0.75} \cdot log_2 n$ |
| Bytes που απεστάλησαν και παραλήφθησαν | $n^{0.75} \cdot p^{1.75}$ |
| Αριθμός Load και Store εντολών | $p^{0.25} \cdot log_2 p \cdot n^{0.75} \cdot log_2{}^2 n$ |
| Χρόνος εκτέλεσης | $sr^{0.5} + f^{0.5} + b^{0.25} + ls^{0.25} \cdot log_2 ls$ |

2. Σύμφωνα με αυτή την αναβάθμιση έχουμε τις εξής μεταβολές:

| Πλήθος διαδικασιών | |
|---|---|
| Παλιά Τιμή | $p$ |
| Νέα Τιμή | $p' = 2p$ |

| Μνήμη | |
|---|---|
| Παλιά Τιμή | $m$ |
| Νέα Τιμή | $m' = m$ |

3. Υπολογίζουμε το νέο αποτύπωμα μνήμης:

| Αποτύπωμα μνήμης | |
|---|---|
| Παλιά Τιμή | $p^{1.5} \cdot log_2 p \cdot n^{0.75} \cdot log_2 n$ |
| Νέα Τιμή | $p'^{1.5} \cdot log_2 p' \cdot n'^{0.75} \cdot log_2 n'$ |
| Λόγος | 1 |

4. Υπολογίζουμε το $n'$ και το νέο συνολικό μέγεθος προβλήματος:

| Μέγεθος προβλήματος ανά διαδικασία | |
|---|---|
| Παλιά Τιμή | $p^{1.5} \cdot log_2 p \cdot n^{0.75} \cdot log_2 n = m$ |
| Νέα Τιμή | $p'^{1.5} \cdot log_2 p' \cdot n'^{0.75} \cdot log_2 n' = m'$ |
| Λόγος | 0.25 |

| Συνολικό μέγεθος προβλήματος | |
|---|---|
| Παλιά Τιμή | $p \cdot n$ |
| Νέα Τιμή | $p' \cdot n'$ |
| Λόγος | 0.5 |

5. Προσδιορίζουμε τις νέες ανάγκες για υπολογισμό, επικοινωνία και πρόσβαση στη μνήμη

| FLOPS | |
|---|---|
| Παλιά Τιμή | $p^{0.75} \cdot n^{0.5} \cdot {log_2}^2 n$ |
| Νέα Τιμή | $(2p)^{0.75} \cdot (0.25n)^{0.5} \cdot {log_2}^2 0.25n$ |
| Λόγος | 0.845 |

| Bytes επικοινωνίας | |
|---|---|
| Παλιά Τιμή | $n^{0.75} \cdot p^{1.75}$ |
| Νέα Τιμή | $(0.25)n^{0.75} \cdot (2p)^{1.75}$ |
| Λόγος | 1.176 |

| Πλήθος εντολών πρόσβασης στη μνήμη | |
|---|---|
| Παλιά Τιμή | $p^{0.25} \cdot log_2 p \cdot n^{0.75} \cdot log_2{}^2 n$ |
| Νέα Τιμή | $(2p)^{0.25} \cdot log_2 2p \cdot (0.25n)^{0.75} \cdot log_2{}^2 0.25n$ |
| Λόγος | 0.42 |

6. Υπολογίζουμε τη νέα προσδοκία για το χρόνο εκτέλεσης

| Χρόνος Εκτέλεσης | |
|---|---|
| Παλιά Τιμή | $sr^{0.5} + f^{0.5} + b^{0.25} + ls^{0.25} \cdot log_2 ls$ |
| Νέα Τιμή | $sr'^{0.5} + f'^{0.5} + b^{0.25} + ls'^{0.25} \cdot log_2 ls'$ |
| Λόγος | 1.1 |

Πίνακας 6.2: *Μεθοδολογία για τον προσδιορισμό των αναγκών του Pennant αν αναβαθμίσουμε το σύστημα διπλασιάζοντάς το. Οι συντελεστές των μοντέλων που επιλέγονται μπορούν να παραλειφθούν αφού η αναβάθμιση είναι σχετική*

Ακολουθώντας την ίδια διαδικασία για όλες τις εφαρμογές και για τις τρεις πιθανές αναβαθμίσεις συστήματος καταλήγουμε στον παρακάτω πίνακα όπου μπορούμε να δούμε συγκεντρωτικά τους λόγους μεταξύ νέων και παλιών προσδοκιών για κάθε μέγεθος ενδιαφέροντος:

| | Kripke | LULESH | AMG2013 | Pennant | Baseline |
|---|---|---|---|---|---|
| **Αναβάθμιση Συστήματος Α:** Διπλασιασμός Συστήματος | | | | | |
| Μέγεθος προβλήματος ανά διαδικασία | 1 | 1 | 0.03 | 0.25 | **1** |
| Συνολικό μέγεθος προβλήματος | 2 | 2 | 0.06 | 0.5 | **2** |
| Υπολογισιμότητα | 1 | 1.7 | 0.007 | 0.85 | **1** |
| Επικοινωνία | 1 | 1.2 | 0.2 | 1.2 | **1** |
| Πρόσβαση στη μνήμη | 1.2 | 1.7 | 0.2 | 0.4 | **1** |
| Χρόνος Εκτέλεσης | 1.1 | 1.3 | 1 | 1.1 | **1** |
| **Αναβάθμιση Συστήματος Β:** Διπλασιασμός Αριθμού Επεξεργαστών | | | | | |
| Μέγεθος προβλήματος ανά διαδικασία | 0.5 | 0.4 | 0.002 | 0.1 | **0.5** |
| Συνολικό μέγεθος προβλήματος | 1 | 1 | 0.004 | 0.2 | **1** |
| Υπολογισιμότητα | 0.5 | 0.65 | 0.0001 | 0.5 | **0.5** |
| Επικοινωνία | 0.5 | 0.5 | 0.05 | 0.6 | **0.5** |
| Πρόσβαση στη μνήμη | 0.6 | 2 | 0.05 | 0.2 | **0.5** |
| Χρόνος Εκτέλεσης | 0.8 | 0.8 | 0.8 | 0.8 | **0.5** |
| **Αναβάθμιση Συστήματος Γ:** Διπλασιασμός Μνήμης | | | | | |
| Μέγεθος προβλήματος ανά διαδικασία | 2 | 2.5 | 16 | 1.5 | **2** |
| Συνολικό μέγεθος προβλήματος | 2 | 2.5 | 16 | 1.5 | **2** |
| Υπολογισιμότητα | 2 | 2.5 | 64 | 1.2 | **2** |
| Επικοινωνία | 2 | 2.5 | 4 | 1.35 | **2** |
| Πρόσβαση στη μνήμη | 2 | 3 | 4 | 1.35 | **2** |
| Χρόνος Εκτέλεσης | 1.4 | 1.6 | 1.2 | 1.2 | **2** |

Πίνακας 6.3: *Σύγκριση μεταξύ των τριών διαφορετικών αναβαθμίσεων*

Στην τελευταία στήλη, με τίτλο "Baseline", μπορούμε να δούμε τις αναμενόμενες μεταβολές αν υποθέσουμε μια αισιόδοξη γραμμική συσχέτιση μεταξύ μεγέθους προβλήματος ανά διαδικασία και αναγκών. Για παράδειγμα, αν διπλασιάσουμε όλο το σύστημα αναμένουμε ότι και το συνολικό μέγεθος προβλήματος που θα μπορούμε να λύσουμε θα διπλασιαστεί, χωρίς όμως να μεταβληθούν οι ανάγκες της εφαρμογής. Φυσικά αυτή η υπόθεση δεν θα αληθεύει γενικά αλλά προσφέρει μια διαίσθηση για την επιθυμητή συμπεριφορά και λειτουργεί ως μέσο σύγκρισης [2].

Αναλύοντας λοιπόν τον πίνακα 6.3 είναι εμφανές ότι το μέγεθος προβλήματος ανά διαδικασία αποτελεί τη σημαντικότερη παράμετρο καθώς, όπως αναδείξαμε και στη μεθοδολογία της άσκησης, καθορίζει όλες τις άλλες παραμέτρους. Παρατηρούμε επίσης ότι, ιδανικά, θα θέλαμε οι ανάγκες της εφαρμογής να ακολουθούν τους λόγους αυτής της παραμέτρου. Έχοντας αυτό υπόψιν παρατηρούμε ότι η μόνη εφαρμογή που πλησιάζει σε αυτή την ιδανική συσχέτιση με ικανοποιητική συχνότητα είναι το Kripke.

Θα επιχειρήσουμε τέλος να συνοψίσουμε πώς θα μπορούσε να επωφεληθεί (ή όχι) η κάθε εφαρμογή από κάθε πιθανή αναβάθμιση συστήματος. Το Kripke μοιάζει να έχει το μεγαλύτερο κέρδος από τις αναβαθμίσεις Β και Γ, καθώς εκεί παρατηρούμε την πιο κοντινή συμπεριφορά προς την ιδανική γραμμική, ενώ ακόμη παρατηρούμε και καλύτερη κλιμάκωση, όσον αφορά το χρόνο εκτέλεσης, για μεγαλύτερα συνολικά μεγέθη προβλήματος. Το LULESH μοιάζει να έχει το μεγαλύτερο όφελος από τον διπλασιασμό ολόκληρου του συστήματος τόσο για τη μεταβολή στις ανάγκες όσο και για την κλιμάκωση για τον χρόνο εκτέλεσης. Το AMG2013 μπορεί να επωφεληθεί περισσότερο από τον διπλασιασμό μνήμης, ειδικά για τον χρόνο εκτέλεσης. Τέλος, το Pennant θα μπορούσε να έχει τα περισσότερα οφέλη από τον διπλασιασμό μνήμης. Παρατηρούμε λοιπόν ότι δεν υπάρχει μια συγκεκριμένη αναβάθμιση που θα ήταν ιδανική για όλες τις εφαρμογές, ωστόσο και οι τρεις μοιάζουν να επωφελούνται ικανοποιητικά από τον διπλασιασμό μνήμης του συστήματος, ειδικά αν λάβουμε υπόψιν και την κλιμάκωση του χρόνου εκτέλεσης.

# Μέρος V

# Μοντελοποίηση Αναγκών σε πιο Σύνθετα Προβλήματα

Κεφάλαιο 7

# Μοντελοποίηση Αναγκών σε πιο Σύνθετα Προβλήματα

Στα κεφάλαια 3 και 4 παρουσιάσαμε μια μέθοδο που έχει αναπτυχθεί [2] για τη μοντελοποίηση των αναγκών μιας εφαρμογής σε δεδομένους πόρους, καθώς και τα πειραματικά μας αποτελέσματα για τέσσερις εφαρμογές. Οι εφαρμογές που μελετήσαμε προηγουμένως αποτελούν μικρούς κώδικες που έχουν εξαχθεί από άλλους αρκετά μεγαλύτερους, με σκοπό να βοηθήσουν σε συγκεκριμένες δοκιμές και προσομοιώσεις. Ενώ τα αποτελέσματα που λαμβάνουμε για αυτές τις εφαρμογές δύναται να αναδείξουν πιθανά προβλήματα στην εκτέλεση εργασιών που περιέχουν τους υπολογισμούς που αυτές μοντελοποιούν, είναι όμως περιοριστικά, καθώς οι εφαρμογές αυτές μοντελοποιούν με βάση δεδομένα και σαφώς ορισμένα μεγέθη εισόδου. Επομένως, σε αυτό το κεφάλαιο, θα παρουσιάσουμε τις ίδιες διαδικασίες μοντελοποίησης για μια πιο σύνθετη εφαρμογή, η οποία δέχεται ως είσοδο έναν γράφο, κι όχι κάποιο απλό μέγεθος διάστασης προβλήματος προς επίλυση.

## 7.1 Η εφαρμογή FLEE

Η εφαρμογή την οποία θα μελετήσουμε είναι η εφαρμογή FLEE [24]. Πρόκειται για έναν κώδικα που βασίζεται σε συστήματα πρακτόρων(Agent Based Modeling) [25], ο οποίος αναπτύχθηκε με σκοπό να μοντελοποιήσει τις κινήσεις προσφύγων και ατόμων που αναγκάστηκαν σε εσωτερική μετακίνηση λόγω ανωτέρων συνθηκών. Η εφαρμογή δέχεται ως είσοδο έναν γράφο, που περιγράφει την τοπολογία μέσα στην οποία ενεργούν οι πράκτορες, καθώς και τον αρχικό αριθμό των πρακτόρων (προσφύγων). Με βάση αυτά, δημιουργείται μια προσομοίωση για την κίνηση των ατόμων στον γράφο, η παράλληλη εκτέλεση της οποίας χρησιμοποιεί το πρωτόκολλο MPI [26]. Προκειμένου να λάβουμε τις μετρήσεις που χρειαζόμαστε χρησιμοποιούμε τα ίδια εργαλεία που παρουσιάσθηκαν και στο κεφάλαιο 3.2, δηλαδή το `PAPI` και το `Score-P`, με τη διαφορά ότι, αφού το FLEE είναι γραμμένο σε Python, χρειαζόμαστε τις αντίστοιχες υλοποιήσεις αυτών των εργαλείων για τη γλώσσα αυτή [27].

Η διαφορά του FLEE με τις εφαρμογές που μελετήσαμε στα προηγούμενα κεφάλαια είναι ότι λαμβάνει είσοδο πιο σύνθετη από ένα απλό μέγεθος εισόδου. Αυτή προσδιορίζεται τόσο από το γράφο και τα μεγέθη που τον χαρακτηρίζουν, δηλαδή ακμές και κορυφές, όσο και από τον αριθμό των πρακτόρων που συμμετέχουν στην προσομοίωση. Μάλιστα το FLEE δίνει μια επιπλέον δυνατότητα προσθήκης πρακτόρων ανά βήμα της εκτέλεσης, την οποία όμως

δεν χρησιμοποιήσαμε στις εκτελέσεις μας. Πειραματιστήκαμε με τέσσερις γράφους και πέντε διαφορετικά πλήθη πρακτόρων για τον καθένα τους. Όλοι οι γράφοι που χρησιμοποιήσαμε είναι πλεγματικού τύπου και παρουσιάζονται στον κατώθι πίνακα:

| Γράφος | Πλήθος Κορυφών | Πλήθος Ακμών |
|---|---|---|
| 10-10-4 | 100 | 200 (4 ακμές/κορυφή) |
| 10-10-8 | 100 | 400 (8 ακμές/κορυφή) |
| 50-50-4 | 2500 | 5000 (4 ακμές/κορυφή) |
| 50-50-8 | 2500 | 10000 (8 ακμές/κορυφή) |

Πίνακας 7.1: *Οι 4 γράφοι που δόθηκαν ως είσοδος στα πειράματα με το FLEE*

Για την μοντελοποίηση των αναγκών εξετάσαμε δύο διαφορετικές αναπαραστάσεις του γράφου ως παράμετρο εισόδου, το μέσο όρο ακμών ανά κορυφή και το συνολικό πλήθος ακμών. Με αυτό τον τρόπο τα μοντέλα τα οποία εξάγουμε δεν είναι τύπου $r_i(n, p)$, σαν αυτά που παρουσιάστηκαν στο κεφάλαιο 4, αλλά είναι της μορφής $r_i(p, n, a)$ ή $r_i(p, n, e)$ αντίστοιχα, όπου $a$ = μέσος όρος ακμών ανά κορυφή, $e$ = συνολικό πλήθος ακμών και $n$ = πλήθος αρχικών πρακτόρων. Εκτός από αυτό η υπόλοιπη διαδικασία για την εξαγωγή των μοντέλων είναι πανομοιότυπη με αυτή που περιγράφηκε στο κεφάλαιο 3.3.1 και για αυτό δεν θα επαναληφθεί η επεξήγησή της σε αυτό το σημείο.

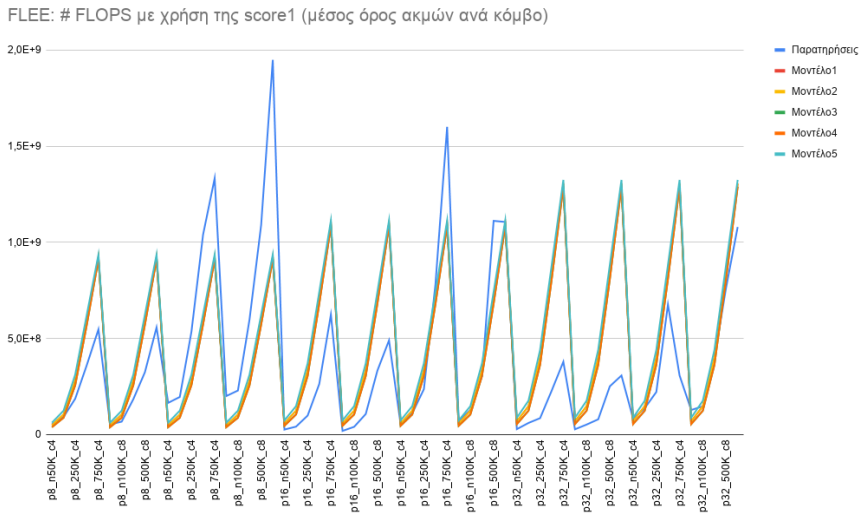
## 7.2 Μοντελοποίηση Αναγκών του FLEE στο μηχάνημα Sandman

Στην ενότητα αυτή θα παρουσιάσουμε τις μετρήσεις και τα μοντέλα τα οποία εξάγαμε για την πρόβλεψη της επίδοσης της εφαρμογής FLEE μέσα από εκτελέσεις στο μηχάνημα Sandman.

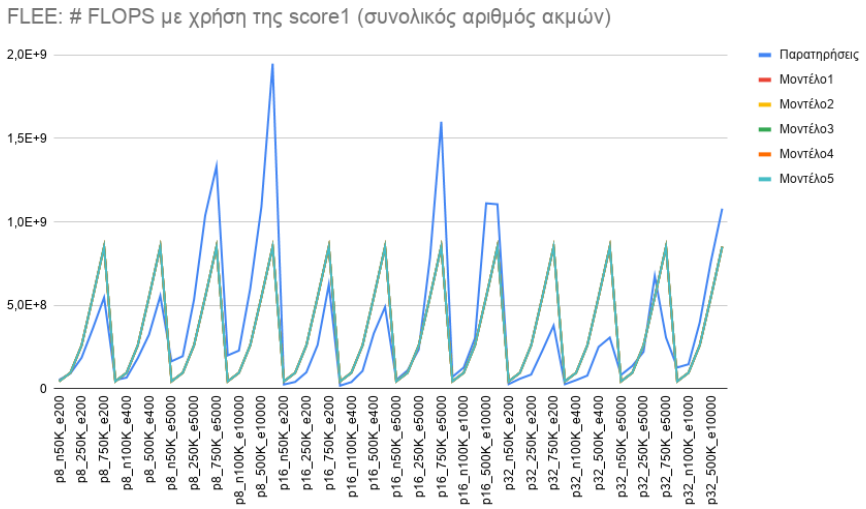
*(Στην ανάλυση που ακολουθεί ως $p$ θεωρούμε το πλήθος των διαδικασιών, ως $n$ το πλήθος των πρακτόρων, ως $a$ το μέσο όρο ακμών/κορυφή και ως $e$ το συνολικό πλήθος ακμών ενός γράφου)*

### 7.2.1 FLOPS

Αρχίζουμε την ανάλυση μας με τη μετρική των πράξεων κινητής υποδιαστολής ανά δευτερόλεπτο (FLOPS). Στο σχήμα 7.1 και στους πίνακες 7.2 και 7.3 βλέπουμε τα αποτελέσματα με χρήση της συνάρτησης αξιολόγησης $score_1$ (εξίσωση 3.2), τόσο για τον μέσο όρο ακμών ανά κορυφή όσο και για το συνολικό πλήθος ακμών.




(α′) *Μέσος Όρος Ακμών/Κορυφή*




(β′) *Σύνολο Ακμών*

Σχήμα 7.1: *Τα 5 καλύτερα μοντέλα πρόβλεψης του πλήθους FLOPS για την εφαρμογή FLEE με χρήση της* $score_1$

| Μοντέλο | Σχετικό Σφάλμα |
|---|---|
| $37.5 \cdot p^{0.25} \cdot n \cdot log_2 n$ | 0.734 |
| $37.5 \cdot (p^{0.25} \cdot n \cdot log_2 n + a^2 \cdot log_2 a)$ | 0.7341 |
| $1.9 \cdot p^{0.25} \cdot n \cdot log_2 n^2$ | 0.57 |
| $1.9 \cdot (p^{0.25} \cdot n \cdot log_2 n^2 + a^2 \cdot log_2 a)$ | 0.57 |
| $742.7 \cdot p^{0.25} \cdot n$ | 0.9316 |

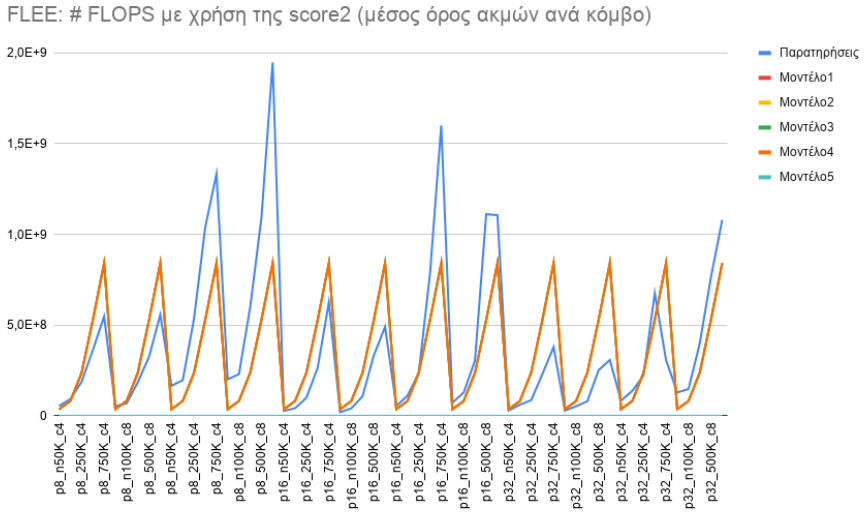
Πίνακας 7.2: *Τα 5 καλύτερα μοντέλα για το πλήθος των FLOPS της εφαρμογής FLEE με είσοδο τον μέσο όρο ακμών/κορυφή και χρήση της* $score_1$

| Μοντέλο | Σχετικό Σφάλμα |
|---|---|
| $58.21 \cdot n \cdot log_2 n$ | 0.2933 |
| $58.21 \cdot (n \cdot log_2 n + e^{0.25})$ | 0.2933 |
| $58.21 \cdot (p^{0.25} + n \cdot log_2 n)$ | 0.2933 |
| $58.21 \cdot (p^{0.25} + n \cdot log_2 n + e^{0.25})$ | 0.2933 |
| $58.21 \cdot (log_2 p + n \cdot log_2 n)$ | 0.2933 |

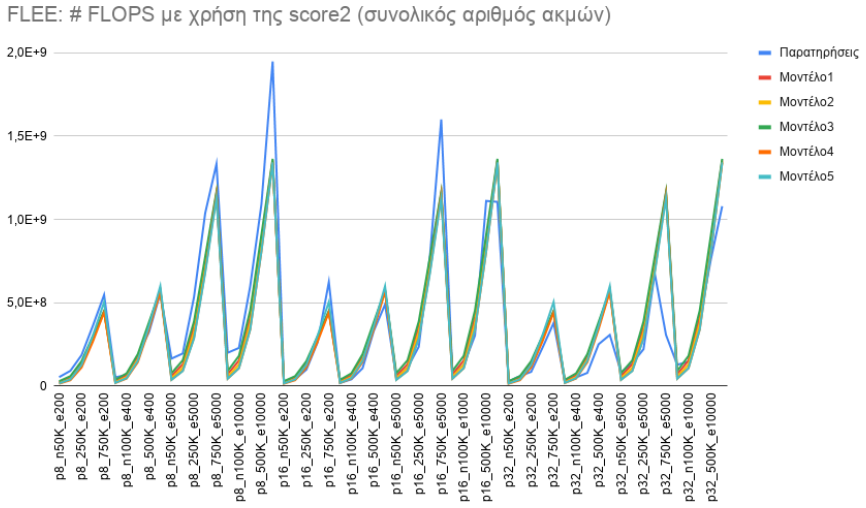
Πίνακας 7.3: *Τα 5 καλύτερα μοντέλα για το πλήθος των FLOPS της εφαρμογής FLEE με είσοδο το συνολικό πλήθος ακμών και χρήση της* $score_1$

Παρατηρώντας τα σχήματα 7.1 είναι εμφανές ότι οι δύο μοντελοποιήσεις με τον μέσο όρο ακμών ανά κορυφή αλλά και με το σύνολο ακμών πετυχαίνουν ικανοποιητικές προβλέψεις για τις υπολογιστικές ανάγκες της εφαρμογής. Συγκεκριμένα παρατηρούμε μικρότερα σφάλματα στις προβλέψεις που εκφράζονται ως προς τον συνολικό αριθμό των ακμών, κάτι που λογικά οφείλεται στις υψηλές τιμές των προβλέψεων του σχήματος 7.1α′ στις 32 διαδικασίες. Επίσης στο ίδιο σχήμα παρατηρούμε ότι το μοντέλο 5 μας δίνει μεγάλο σφάλμα καθώς "υπό-εκτιμά" τις πράξεις κινητής υποδιαστολής που εκτελεί η εφαρμογή μας. Όσον αφορά τις παρατηρήσεις καθαυτές είναι εμφανές ότι εξαρτώνται σε μεγάλο βαθμό από τον αριθμό των πρακτόρων ($n$), μιας και σε κάθε γράφο έχουμε σταδιακή αύξηση της μετρικής σε συνάρτηση με αυτόν. Αυτό είναι κάτι το οποίο καταφέρνουν να προβλέψουν τα μοντέλα που εξάγουμε με χρήση της συνάρτησης αξιολόγησης $score_1 = R^2$, αφού όλα εμφανίζουν εξάρτηση από το $n$.

Στη συνέχεια θα παρουσιάσουμε τα πειραματικά μας αποτελέσματα για την πρόβλεψη των FLOPS με χρήση της συνάρτησης αξιολόγησης $score_2 = R2 \cdot RCC \cdot pred_{<0.35}$.




(α′) *Μέσος Όρος Ακμών/Κορυφή*




(β′) *Σύνολο Ακμών*

Σχήμα 7.2: *Τα 5 καλύτερα μοντέλα πρόβλεψης του πλήθους FLOPS για την εφαρμογή FLEE με χρήση της* $score_2$

| Μοντέλο | Σχετικό Σφάλμα |
|---|---|
| $2.945 \cdot (p^2 \cdot log_2 p^2 \cdot a^{0.25} \cdot log_2 a + n \cdot log_2 n^2)$ | 0.1707 |
| $2.945 \cdot (p^2 \cdot log_2 p^2 \cdot log_2 a + n \cdot log_2 n^2)$ | 0.1702 |
| $2.945 \cdot (p^2 \cdot log_2 p^2 \cdot a^{0.5} + n \cdot log_2 n^2)$ | 0.1702 |
| $2.945 \cdot (p^2 \cdot log_2 p^2 \cdot a^{0.25} + n \cdot log_2 n^2)$ | 0.17 |
| $2.945 \cdot (p^2 \cdot log_2 p^2 + log_2 n^2 + a^{0.25} \cdot log_2 a)$ | 0.9997 |

Πίνακας 7.4: *Τα 5 καλύτερα μοντέλα για το πλήθος των FLOPS της εφαρμογής FLEE με είσοδο τον μέσο όρο ακμών/κορυφή και χρήση της* $score_2$

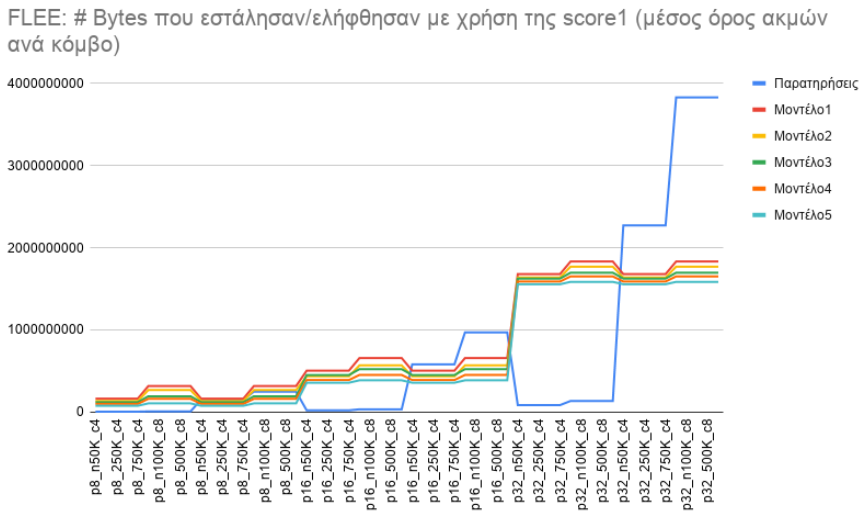
| Μοντέλο | Σχετικό Σφάλμα |
|---|---|
| $0.5245 \cdot n \cdot log_2 n \cdot log_2 e^2$ | 0.0797 |
| $0.0267 \cdot n \cdot log_2 n^2 \cdot log_2 e^2$ | 0.0152 |
| $10.304 \cdot n \cdot log_2 e^2$ | 0.1919 |
| $0.344 \cdot n^{1.25} \cdot log_2 e^2$ | 0.1054 |
| $6.0856 \cdot n^{1.25} \cdot e^{0.25}$ | 0.0655 |

Πίνακας 7.5: *Τα 5 καλύτερα μοντέλα για το πλήθος των FLOPS της εφαρμογής FLEE με είσοδο το συνολικό πλήθος ακμών και χρήση της* $score_2$

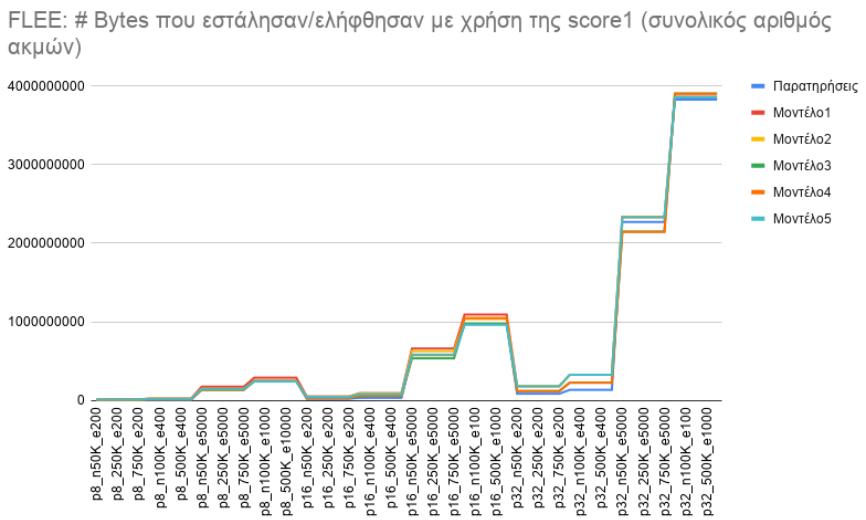
Σε αυτή την περίπτωση παρατηρούμε καλύτερη πρόβλεψη του αριθμού των FLOPS, ιδίως από τα μοντέλα του πίνακα 7.5. Σε αυτά παρατηρούμε αρκετά μικρό σχετικό σφάλμα, κάτι το οποίο διαπιστώνουμε εύκολα και από τη γραφική απεικόνιση. Πιο συγκεκριμένα, οι προβλέψεις αυτές μας υποδηλώνουν ότι οι υπολογιστικές ανάγκες της εφαρμογής εξαρτώνται αποκλειστικά από το πρόβλημα εισόδου, δηλαδή τον αριθμό των πρακτόρων αλλά και το πλήθος των ακμών του γράφου, και δεν υπάρχει συσχέτιση με το πλήθος των διαδικασιών στις οποίες μοιράζεται η προσομοίωση, γεγονός που μάλλον αποδεικνύεται κι από την απεικόνιση των παρατηρήσεων.

### 7.2.2 Αριθμός Bytes που ελήφθησαν και απεστάλησαν ανά διαδικασία

Στη συνέχεια θα παρουσιάσουμε τα αποτελέσματά μας για τις ανάγκες της προσομοίωσης σε επικοινωνία μεταξύ των διαδικασιών, με την πρόβλεψη του πλήθους των Bytes που επικοινωνήθηκαν μεταξύ αυτών.




(α′) *Μέσος Όρος Ακμών/Κορυφή*




(β′) *Σύνολο Ακμών*

Σχήμα 7.3: *Τα 5 καλύτερα μοντέλα πρόβλεψης του πλήθους Bytes που ελήφθησαν και απεστάλησαν για την εφαρμογή FLEE με χρήση της* $score_1$

| Μοντέλο | Σχετικό Σφάλμα |
|---|---|
| $1.81 \cdot 10^6 \cdot (p^{1.5} \cdot log_2 p + a^{2.25})$ | 11.06 |
| $1.56 \cdot 10^6 \cdot (p^2 + a^{2.25})$ | 9.57 |
| $8.44 \cdot 10^5 \cdot (p^{1.25} \cdot log_2 p^2 + a^{2.25})$ | 8.51 |
| $7.305 \cdot 10^5 \cdot (p^{1.75} \cdot log_2 p + a^{2.25})$ | 7.41 |
| $3.421 \cdot 10^5 \cdot (p^{1.5} \cdot log_2 p^2 + a^{2.25})$ | 6.16 |

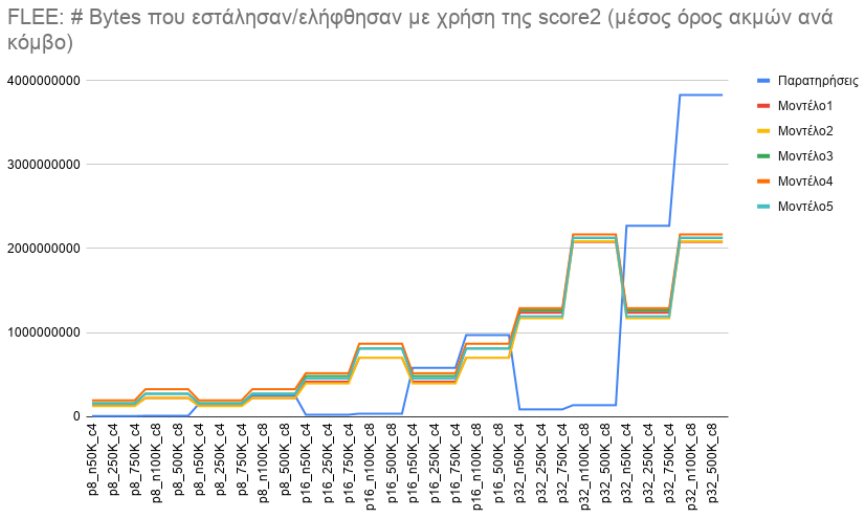
Πίνακας 7.6: *Τα 5 καλύτερα μοντέλα για το πλήθος των Bytes που ελήφθησαν και απεστάλησαν της εφαρμογής FLEE με είσοδο τον μέσο όρο ακμών/κορυφή και χρήση της* $score_1$

| Μοντέλο | Σχετικό Σφάλμα |
|---|---|
| $2.42 \cdot 10^2 \cdot p^{1.5} \cdot log_2 p \cdot e^{0.5} \cdot log_2 e^2$ | 0.7748 |
| $1.15 \cdot 10^5 \cdot p^{1.25} \cdot log_2 p^2 \cdot e^{0.5} \cdot log_2 e^2$ | 0.6476 |
| $2.87 \cdot 10^2 \cdot p^2 \cdot e^{0.75} \cdot log_2 e$ | 0.2291 |
| $1.54 \cdot 10^2 \cdot p^{1.25} \cdot log_2 p^2 \cdot e^{0.75} \cdot log_2 e$ | 0.2669 |
| $2.13 \cdot 10^2 \cdot p^2 \cdot e^{0.5} \cdot log_2 e^2$ | 0.5974 |

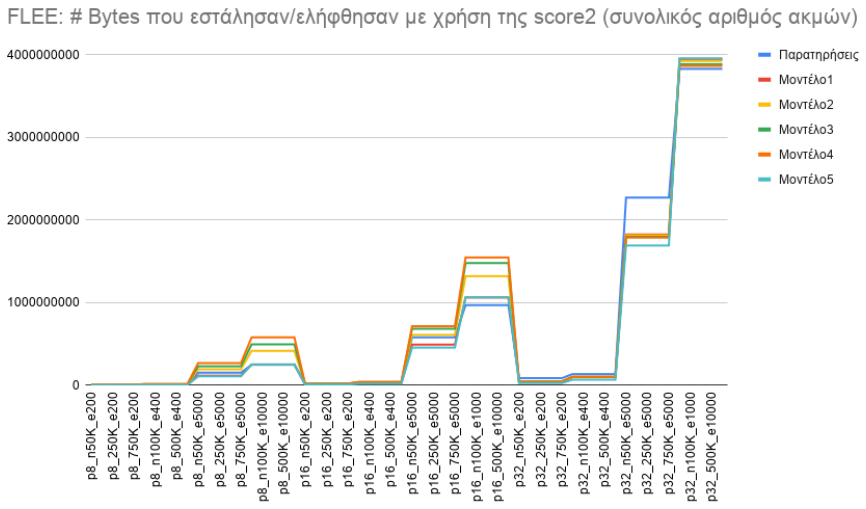
Πίνακας 7.7: *Τα 5 καλύτερα μοντέλα για το πλήθος των Bytes που ελήφθησαν και απεστάλησαν της εφαρμογής FLEE με είσοδο το συνολικό πλήθος ακμών και χρήση της* $score_1$

Παρατηρώντας τα σχήματα 7.3 είναι εμφανές ότι οι παρατηρήσεις όσον αφορά τα Bytes που αποστέλλονται και λαμβάνονται από τις διαδικασίες κατά τη διάρκεια εκτέλεσης του FLEE είναι ανεξάρτητα του πλήθους πρακτόρων $n$. Αυτό αποτυπώνεται στη μορφή των παρατηρήσεων, η οποία μοιάζει ως "βηματική", διατηρώντας την ίδια τιμή όσο τα $p, a/e$ παραμένουν αμετάβλητα. Αυτή η ανεξαρτησία αποτυπώνεται και στα μοντέλα πρόβλεψης που κατασκευάζει η μέθοδός μας, από τα οποία απουσιάζει ο παράγοντας $n$. Όσον αφορά τη σύγκριση των δύο διαφορετικών παραμετροποιήσεων της εισόδου (μέσος όρος ακμών ανά κορυφή και συνολικός αριθμός ακμών), παρατηρούμε ότι οι προβλέψεις του πίνακα 7.7 επιτυγχάνουν σαφώς μικρότερο σφάλμα σε σύγκριση με αυτά που βασίζονται στο μέσο όρο ακμών/κορυφή. Πέρα από τη διαφορετική παράμετρο εισόδου, αυτά μοιάζουν να πετυχαίνουν καλύτερα τον σκοπό τους αφού παρουσιάζουν εξάρτηση από το γινόμενο των $p, e$, σε αντίθεση με αυτά του πίνακα 7.6 τα οποία εξαρτώνται από το άθροισμα των $p, a$. Τέλος, η εξάρτηση της μετρικής αποκλειστικά από το πλήθος ακμών και διαδικασιών είναι μάλλον αναμενόμενη, καθώς ο κώδικας υλοποιεί την επικοινωνία με χρήση "συγκεντρωτικών" ("collectives") συναρτήσεων του MPI, όπως `MPI_Allreduce`, `MPI_Gather`, οι οποίες κι εκτελούνται ανά κόμβο του γράφου.

Στη συνέχεια παρουσιάζουμε τα ευρήματά μας με χρήση της συνάρτησης αξιολόγησης $score_2$.




(α′) *Μέσος Όρος Ακμών/Κορυφή*




(β′) *Σύνολο Ακμών*

Σχήμα 7.4: *Τα 5 καλύτερα μοντέλα πρόβλεψης του πλήθους Bytes που ελήφθησαν και απεστάλησαν για την εφαρμογή FLEE με χρήση της* $score_2$

| Μοντέλο | Σχετικό Σφάλμα |
|---|---|
| $1.15 \cdot 10^6 \cdot p^{1.25} \cdot log_2 p \cdot a^{0.75}$ | 9.091 |
| $1.087 \cdot 10^6 \cdot p^{1.25} \cdot log_2 p \cdot a^{0.25} \cdot log_2 a$ | 8.848 |
| $1.33 \cdot 10^6 \cdot p^{0.75} \cdot log_2 p^2 \cdot a^{0.75}$ | 10.616 |
| $2.85 \cdot 10^6 \cdot p \cdot log_2 p \cdot a^{0.75}$ | 11.97 |
| $1.255 \cdot 10^6 \cdot p^{0.75} \cdot log_2 p^2 \cdot a^{0.25} \cdot log_2 a$ | 10.312 |

Πίνακας 7.8: *Τα 5 καλύτερα μοντέλα για το πλήθος των Bytes που ελήφθησαν και απεστάλησαν της εφαρμογής FLEE με είσοδο τον μέσο όρο ακμών/κορυφή και χρήση της* $score_2$

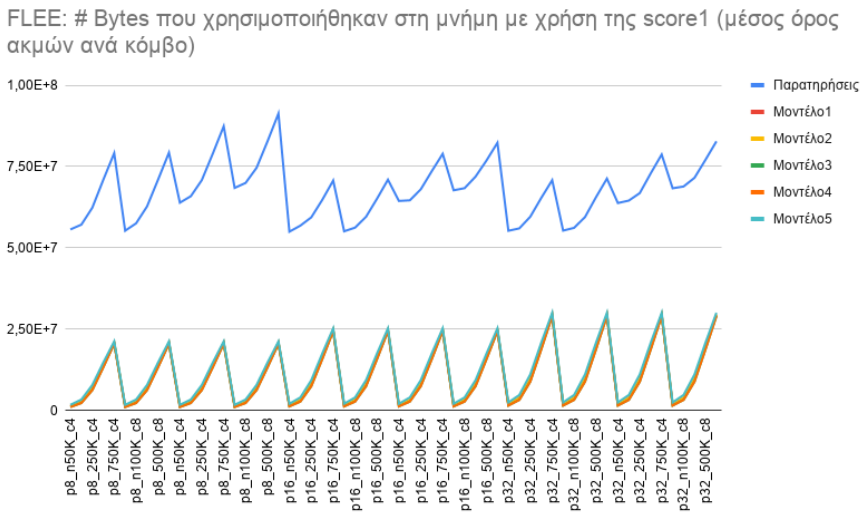
| Μοντέλο | Σχετικό Σφάλμα |
|---|---|
| $15.616 \cdot p^{1.25} \cdot log_2 p^2 \cdot e \cdot log_2 e$ | 0.2132 |
| $77.635 \cdot p^{1.25} \cdot log_2 p \cdot e \cdot log_2 e$ | 0.0174 |
| $86.94 \cdot p^{0.75} \cdot log_2 p^2 \cdot e \cdot log_2 e$ | 0.134 |
| $181.81 \cdot p \cdot log_2 p \cdot e \cdot log_2 e$ | 0.2354 |
| $1.177 \cdot p^{1.25} \cdot log_2 p^2 \cdot e \cdot log_2 e^2$ | 0.3502 |

Πίνακας 7.9: *Τα 5 καλύτερα μοντέλα για το πλήθος των Bytes που ελήφθησαν και απεστάλησαν της εφαρμογής FLEE με είσοδο το συνολικό πλήθος ακμών και χρήση της* $score_2$

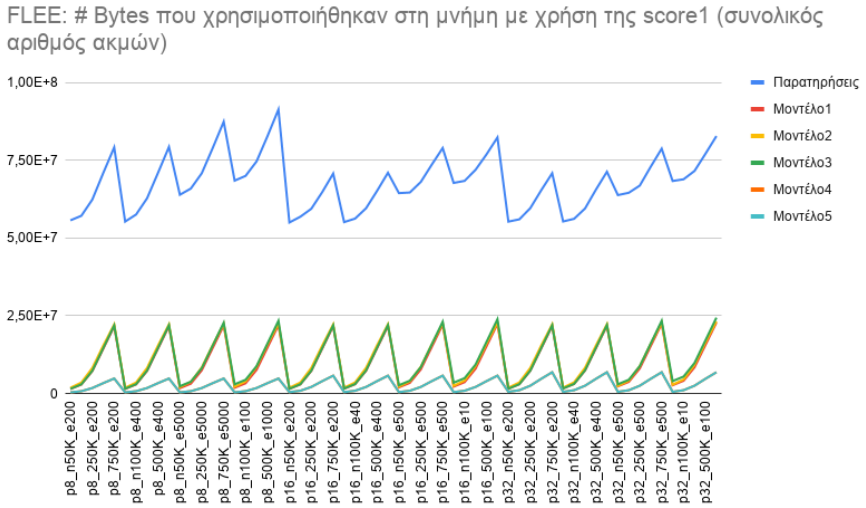
Είναι εμφανές ότι χρησιμοποιώντας και τη δεύτερη μέθοδο αξιολόγησης τα μοντέλα που βασίζονται στο συνολικό πλήθος ακμών επιτυγχάνουν καλύτερες προβλέψεις συγκριτικά με αυτά του πίνακα 7.8. Αυτό είναι μάλλον αναμενόμενο αφού οι τιμές του $a$ είναι ίδιες για τους γράφους "10-10-4", "50-50-4" και "10-10-8", "50-50-8" αντιστοίχως, ενώ αντίθετα το $e$ λαμβάνει μοναδική τιμή για κάθε γράφο. Όπως διαπιστώθηκε και στην ανάλυση των απλούστερων εφαρμογών, η δεύτερη μέθοδος αξιολόγησης δίνει βέλτιστα αποτελέσματα, αφού λαμβάνει υπόψιν διαφορετικές μετρικές αξιολόγησης. Τέλος, όπως γίνεται εύκολα αντιληπτό από την εξάρτηση της επικοινωνίας από μέγεθος γράφου και πλήθος διαδικασιών, ο αριθμός Bytes που απαιτούνται για επικοινωνία θα πληθαίνει σε απαγορευτικό βαθμό, όσο τα $p, e$ (ή $p, a$) αυξάνουν.

### 7.2.3 Αριθμός Bytes που χρησιμοποιήθηκαν στη μνήμη

Στη συνέχεια παρουσιάζουμε τις προβλέψεις για τις ανάγκες της εφαρμογής σε Bytes μνήμης, ξεκινώντας από τα μοντέλα με τη μέθοδο αξιολόγησης $score_1$.




(α′) *Μέσος Όρος Ακμών/Κορυφή*




(β′) *Σύνολο Ακμών*

Σχήμα 7.5: *Τα 5 καλύτερα μοντέλα πρόβλεψης του πλήθους Bytes που χρησιμοποιήθηκαν στη μνήμη για την εφαρμογή FLEE με χρήση της* $score_1$

| Μοντέλο | Σχετικό Σφάλμα |
|---|---|
| $16.54 \cdot (p^{0.25} \cdot n + a^2 \cdot log_2 a^2)$ | 0.8463 |
| $16.55 \cdot p^{0.25} \cdot n$ | 0.8464 |
| $0.835 \cdot (p^{0.25} \cdot n \cdot log_2 n + a^2 \cdot log_2 a^2)$ | 0.8548 |
| $0.835 \cdot p^{0.25} \cdot n \cdot log_2 n$ | 0.8548 |
| $1.3 \cdot (p^{0.25} \cdot n^{0.75} \cdot log_2 n^2 + a^2 \cdot log_2 a^2)$ | 0.8349 |

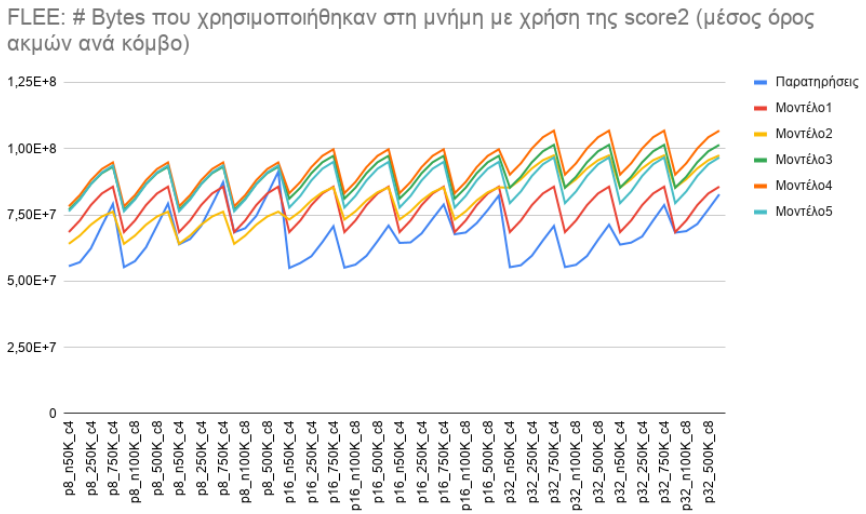
Πίνακας 7.10: *Τα 5 καλύτερα μοντέλα για το πλήθος των Bytes που χρησιμοποιήθηκαν στη μνήμη για την εφαρμογή FLEE με είσοδο τον μέσο όρο ακμών/κορυφή και χρήση της* $score_1$

| Μοντέλο | Σχετικό Σφάλμα |
|---|---|
| $28.777 \cdot (log_2 p^2 \cdot e^{0.25} \cdot log_2 e^2 + n)$ | 0.8622 |
| $2.29 \cdot (log_2 p^2 \cdot e^{0.5} \cdot log_2 e^2 + n^{0.75} \cdot log_2 n^2)$ | 0.8529 |
| $29.13 \cdot (log_2 p \cdot e^{0.5} \cdot log_2 e^2 + n)$ | 0.8542 |
| $0.297 \cdot (p^{0.25} \cdot n^{0.75} \cdot log_2 n^2 + e^{0.5} \cdot log_2 e^2)$ | 0.9623 |
| $0.297 \cdot (p^{0.25} \cdot n^{0.75} \cdot log_2 n^2 + e^{0.25} \cdot log_2 e^2)$ | 0.9623 |

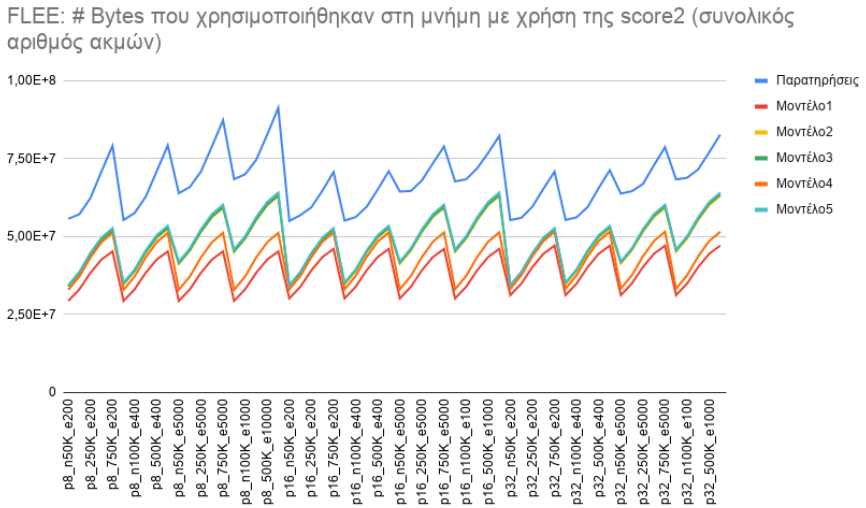
Πίνακας 7.11: *Τα 5 καλύτερα μοντέλα για το πλήθος των Bytes που χρησιμοποιήθηκαν στη μνήμη για την εφαρμογή FLEE με είσοδο το συνολικό πλήθος ακμών και χρήση της* $score_1$

Στα σχήματα 7.5 μπορούμε να δούμε τις προβλέψεις των μοντέλων που εξάγαμε με τη $score_1$ για τα Bytes που χρησιμοποιήθηκαν στη μνήμη από την εφαρμογή FLEE. Είναι εμφανές ότι οι προβλέψεις, τόσο με χρήση του μέσου όρου ακμών ανά κορυφή όσο και με το σύνολο ακμών, απέχουν από τις πραγματικές τιμές που σημειώνονται με μπλε χρώμα. Αυτό αποτυπώνεται και στους αντίστοιχους πίνακες, όπου η χαμηλότερη τιμή σχετικού σφάλματος που παρατηρούμε είναι περίπου 83.5%.

Στη συνέχεια θα παρουσιάσουμε τα αντίστοιχα αποτελέσματά μας με χρήση της $score_2$.




(α′) *Μέσος Όρος Ακμών/Κορυφή*




(β′) *Σύνολο Ακμών*

Σχήμα 7.6: *Τα 5 καλύτερα μοντέλα πρόβλεψης του πλήθους Bytes που χρησιμοποιήθηκαν στη μνήμη για την εφαρμογή FLEE με χρήση της* $score_2$

| Μοντέλο | Σχετικό Σφάλμα |
|---|---|
| $4.39 \cdot 10^6 \cdot log_2 n$ | 0.1583 |
| $3.1 \cdot 10^6 \cdot (p^{0.25} \cdot log_2 p + log_2 n)$ | 0.2083 |
| $4.14 \cdot 10^6 \cdot (log_2 p + log_2 n)$ | 0.3412 |
| $4.24 \cdot 10^6 \cdot (p^{0.5} + log_2 n)$ | 0.3859 |
| $4.416 \cdot 10^6 \cdot (p^{0.25} + log_2 n)$ | 0.2997 |

Πίνακας 7.12: *Τα 5 καλύτερα μοντέλα για το πλήθος των Bytes που χρησιμοποιήθηκαν στη μνήμη για την εφαρμογή FLEE με είσοδο τον μέσο όρο ακμών/κορυφή και χρήση της* $score_2$

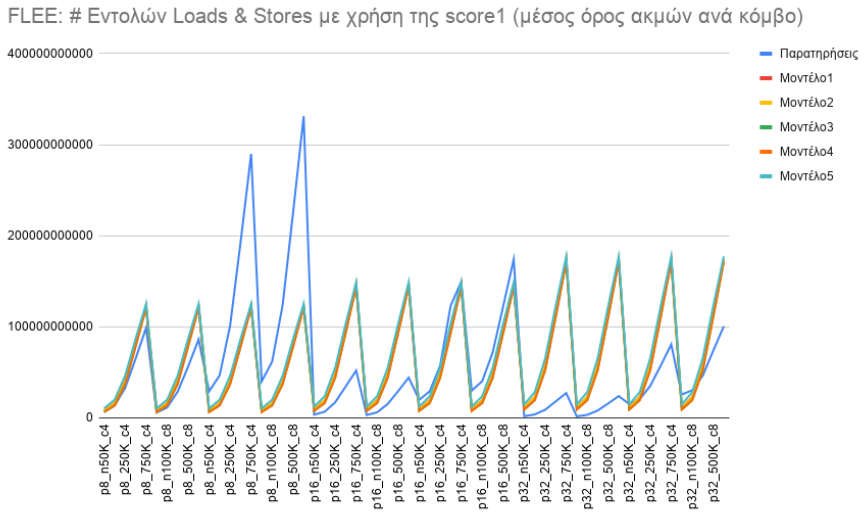
| Μοντέλο | Σχετικό Σφάλμα |
|---|---|
| $1.16 \cdot 10^5 \cdot (log_2 p^2 + log_2 n^2)$ | 0.4289 |
| $1.3 \cdot 10^5 \cdot (p^{0.25} + log_2 n^2 + e^{0.5})$ | 0.2921 |
| $1.3 \cdot 10^5 \cdot (log_2 p + log_2 n^2 + e^{0.5})$ | 0.2866 |
| $1.33 \cdot 10^5 \cdot (p^{0.5} + log_2 n^2)$ | 0.3686 |
| $1.33 \cdot 10^5 \cdot (log_2 n^2 + e^{0.5})$ | 0.2801 |

Πίνακας 7.13: *Τα 5 καλύτερα μοντέλα για το πλήθος των Bytes που χρησιμοποιήθηκαν στη μνήμη για την εφαρμογή FLEE με είσοδο το συνολικό πλήθος ακμών και χρήση της* $score_2$

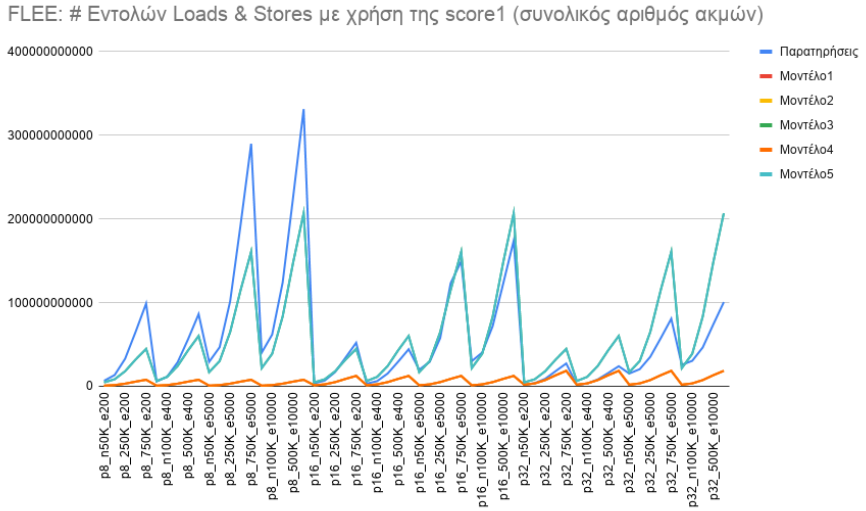
Παρατηρώντας τα σχήματα 7.6 γίνεται εύκολα αντιληπτό ότι τα μοντέλα της $score_2$ επιτυγχάνουν πολύ καλύτερη απόδοση σε σύγκριση με αυτά της $score_1$. Αυτό είναι κάτι που παρατηρούμε και στα σχετικά σφάλματα των δύο πινάκων, όπου έχουμε αρκετά χαμηλές τιμές. Όσον αφορά τα μοντέλα του πίνακα 7.12, κανένα από αυτά δεν έχει εξάρτηση από το μέσο όρο ακμών ανά κορυφή. Αντίστοιχα, στον πίνακα 7.13, παρατηρούμε ότι ο παράγοντας $e$ είτε απουσιάζει, είτε βρίσκεται υψωμένος σε αρκετά μικρό εκθέτη. Επιπλέον, παρατηρώντας το σχήμα 7.6β′, αντιλαμβανόμαστε ότι τα μοντέλα 3,4,5 του πίνακα 7.13 επιτυγχάνουν αρκετά καλή προσαρμογή στο "σχήμα" των παρατηρήσεων. Τέλος, τόσο λόγω της γραφικής αναπαράστασης όσο και λόγω της μορφής των προβλέψεων, αναμένουμε καλή κλιμάκωση ως προς τη δεδομένη μετρική σε μεγαλύτερα προβλήματα και συστήματα. Εξάλλου, αφού πρόκειται για κώδικα Python, στη διαχείριση μνήμης εμπλέκονται και οι βιβλιοθήκες της γλώσσας, όπως π.χ. η numpy, οι οποίες υλοποιούν δικά τους μοντέλα διαχείρισης μνήμης.

### 7.2.4 Πλήθος εντολών πρόσβασης στη μνήμη (Loads/Stores)

Τέλος παρουσιάζουμε τις μετρήσεις και τις προβλέψεις μας για το πλήθος των εντολών πρόσβασης στη μνήμη που εκτελεί το FLEE.




(α′) *Μέσος Όρος Ακμών/Κορυφή*




(β′) *Σύνολο Ακμών*

Σχήμα 7.7: *Τα 5 καλύτερα μοντέλα πρόβλεψης του πλήθους των Loads & Stores εντολών για την εφαρμογή FLEE με χρήση της* $score_1$

| Μοντέλο | Σχετικό Σφάλμα |
|---|---|
| $9.764 \cdot 10^4 \cdot p^{0.25} \cdot n$ | 1.2252 |
| $9.763 \cdot 10^4 \cdot (p^{0.25} \cdot n + a^{2.25} \cdot log_2 a)$ | 1.2265 |
| $4.937 \cdot 10^3 \cdot p^{0.25} \cdot n \cdot log_2 n$ | 1.0022 |
| $4.927 \cdot 10^3 \cdot (p^{0.25} \cdot n \cdot log_2 n + a^{2.25} \cdot log_2 a)$ | 1.0023 |
| $7.684 \cdot 10^3 \cdot p^{0.25} \cdot n^{0.75} \cdot log_2 n^2$ | 1.527 |

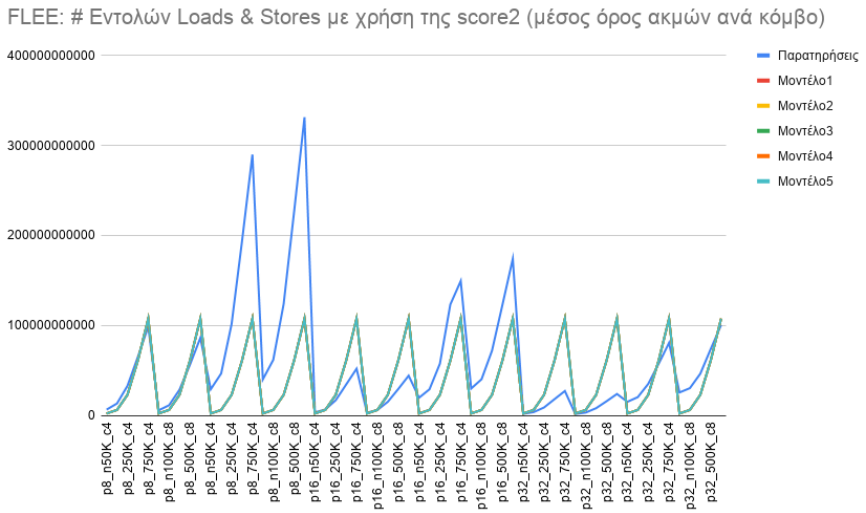
Πίνακας 7.14: *Τα 5 καλύτερα μοντέλα για το πλήθος των Loads & Stores εντολών για την εφαρμογή FLEE με είσοδο τον μέσο όρο ακμών/κορυφή και χρήση της* $score_1$

| Μοντέλο | Σχετικό Σφάλμα |
|---|---|
| $3.13 \cdot 10^3 \cdot (p^{0.25} \cdot log_2 p \cdot n^{0.75} \cdot log_2 n + e^{0.25} \cdot log_2 e^2)$ | 0.7384 |
| $3.13 \cdot 10^3 \cdot (p^{0.25} \cdot log_2 p \cdot n^{0.75} \cdot log_2 n + log_2 e^2)$ | 0.7384 |
| $3.13 \cdot 10^3 \cdot (n^{0.75} \cdot log_2 n \cdot e^{0.25} \cdot log_2 e + log_2 p^2)$ | 0.3106 |
| $3.13 \cdot 10^3 \cdot (p^{0.25} \cdot log_2 p \cdot n^{0.75} \cdot log_2 n + e^{0.25} \cdot log_2 e)$ | 0.7384 |
| $3.13 \cdot 10^3 \cdot (n^{0.75} \cdot log_2 n \cdot e^{0.25} \cdot log_2 e + p^{0.25} \cdot log_2 p)$ | 0.3106 |

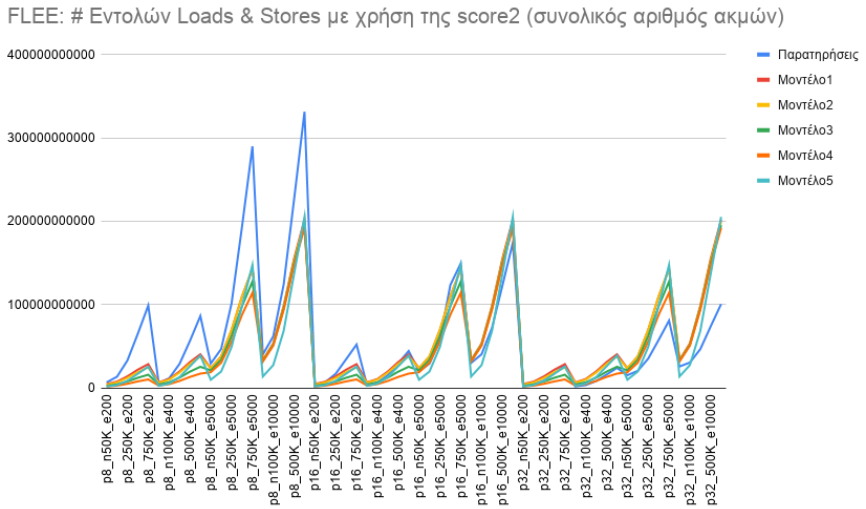
Πίνακας 7.15: *Τα 5 καλύτερα μοντέλα για το πλήθος των Loads & Stores εντολών για την εφαρμογή FLEE με είσοδο το συνολικό πλήθος ακμών και χρήση της* $score_1$

Στα σχήματα 7.7 παρουσιάζουμε τις παρατηρήσεις μας για το πλήθος εντολών πρόσβασης στη μνήμη καθώς και τις προβλέψεις που λαμβάνουμε για αυτές με χρήση της $score_1$. Παρατηρούμε ότι τα μοντέλα του πίνακα 7.14 διαθέτουν υψηλά σφάλματα κάτι που μπορούμε να παρατηρήσουμε και γραφικά, αφού φαίνεται να υποεκτιμούν τις παρατηρήσεις για χαμηλό αριθμό διαδικασιών, ενώ τις υπερεκτιμούν για υψηλό αριθμό διαδικασιών. Όσον αφορά τα μοντέλα του πίνακα 7.15 παρατηρούμε ότι τα μοντέλα 3,5 επιτυγχάνουν μια αρκετά καλή εκτίμηση των παρατηρήσεων, σε αντίθεση με τα υπόλοιπα τρία τα οποία κι υποεκτιμούν τις παρατηρήσεις.

Στη συνέχεια παρουσιάζουμε τα αποτελέσματά μας για τη συνάρτηση αξιολόγησης $score_2$.




(α′) *Μέσος Όρος Ακμών/Κορυφή*




(β′) *Σύνολο Ακμών*

Σχήμα 7.8: *Τα 5 καλύτερα μοντέλα πρόβλεψης του πλήθους των Loads & Stores εντολών για την εφαρμογή FLEE με χρήση της* $score_2$

| Μοντέλο | Σχετικό Σφάλμα |
|---|---|
| $12.81 \cdot (p^2 \cdot log_2 p^2 \cdot a^{1.25} + n^{1.25} \cdot log_2 n^2)$ | 0.0167 |
| $12.81 \cdot (p^2 \cdot log_2 p^2 \cdot log_2 a + n^{1.25} \cdot log_2 n^2)$ | 0.0166 |
| $12.81 \cdot (p^2 \cdot log_2 p^2 \cdot a^{0.5} + n^{1.25} \cdot log_2 n^2)$ | 0.0166 |
| $12.81 \cdot (p^2 \cdot log_2 p^2 \cdot a^{0.25} + n^{1.25} \cdot log_2 n^2)$ | 0.0166 |
| $12.81 \cdot (p^2 \cdot log_2 p^2 + n^{1.25} \cdot log_2 n^2 + a^{1.25})$ | 0.0166 |

Πίνακας 7.16: *Τα 5 καλύτερα μοντέλα για το πλήθος των Loads & Stores εντολών για την εφαρμογή FLEE με είσοδο τον μέσο όρο ακμών/κορυφή και χρήση της* $score_2$

| Μοντέλο | Σχετικό Σφάλμα |
|---|---|
| $6.12 \cdot 10^3 \cdot n^{0.5} \cdot log_2 n^2 \cdot e^{0.5}$ | 0.2652 |
| $3.4166 \cdot 10^2 \cdot n^{0.5} \cdot log_2 n^2 \cdot e^{0.25} \cdot log_2 e^2$ | 0.2016 |
| $4.4532 \cdot 10^2 \cdot n^{0.5} \cdot log_2 n^2 \cdot e^{0.5} \cdot log_2 e$ | 0.0346 |
| $5.82 \cdot 10^2 \cdot n^{0.5} \cdot log_2 n^2 \cdot e^{0.75}$ | 0.1981 |
| $1.55 \cdot 10^2 \cdot n \cdot e^{0.25} \cdot log_2 e^2$ | 0.1841 |

Πίνακας 7.17: *Τα 5 καλύτερα μοντέλα για το πλήθος των Loads & Stores εντολών για την εφαρμογή FLEE με είσοδο το συνολικό πλήθος ακμών και χρήση της* $score_2$

Παρατηρώντας τα σχήματα 7.8 και τους αντίστοιχους πίνακες μπορούμε να εξάγουμε συμπεράσματα για τις προβλέψεις των μοντέλων μας για τις εντολές πρόσβασης στη μνήμη που εκτελεί το FLEE. Συγκεκριμένα, παρατηρούμε ότι τα μοντέλα του πίνακα 7.16 διαθέτουν πολύ μικρά σφάλματα, χωρίς όμως να επιδεικνύουν τη βέλτιστη προσαρμογή, όπως μαρτυρά το σχήμα 7.8α′. Αυτό συμβαίνει διότι τα μοντέλα αυτά επιλέγουν μια "μέση προσαρμογή" στα δεδομένα, η οποία μοιάζει να ταιριάζει περισσότερο στις παρατηρήσεις για $p = 16$. Με αυτό τον τρόπο οι προβλέψεις επιδεικνύουν χαμηλό σφάλμα, χωρίς όμως να προσαρμόζονται σχηματικά στις παρατηρήσεις, ενώ μάλιστα αυτός είναι και ο λόγος που επιλέγονται από τη $score_2$, η οποία "επιβραβεύει" τις προβλέψεις με χαμηλά σφάλματα. Αντιθέτως, τα μοντέλα που παραμετροποιούνται με βάση το συνολικό πλήθος ακμών επιδεικνύουν πολύ καλύτερη προσαρμογή στα δεδομένα μας, χωρίς να έχουν απαραίτητα τόσο χαμηλές τιμές σφαλμάτων. Μάλιστα παρατηρούμε ότι τα μοντέλα του πίνακα 7.17 εξαρτώνται μόνο από το πλήθος πρακτόρων και ακμών κι όχι από τον αριθμό των διαδικασιών οι οποίες τρέχουν τον κώδικα. Τέλος, οι χαμηλοί εκθέτες αυτών των παραγόντων υποδηλώνουν ότι η εφαρμογή αναμένεται να "κλιμακώνει" χωρίς ιδιαίτερα προβλήματα σε συστήματα μεγαλύτερου μεγέθους και για μεγαλύτερες εισόδους προβλήματος.

# Μέρος VI

# Τελικές παρατηρήσεις

Κεφάλαιο 8

# Συμπεράσματα

Ολοκληρώνοντας την εργασία, είναι σημαντικό σε αυτό το κεφάλαιο να πραγματοποιήσουμε μια σύνοψη των όσων παρουσιάστηκαν αλλά και τα συμπεράσματα που μπορούν να προκύψουν από αυτά.

## 8.1 Σύνοψη-Συμπεράσματα της εργασίας

Στην εργασία αυτή παρουσιάσαμε, στο μέρος 3, μια εμπειρική μέθοδο που έχει αναπτυχθεί στα [16] και [2] με σκοπό τη μοντελοποίηση των αναγκών ορισμένων εφαρμογών συγκριτικά με συγκεκριμένους πόρους του συστήματος. Η μέθοδος αυτή αναπτύχθηκε με σκοπό να βοηθήσει τόσο τους κατασκευαστές των συστημάτων της επόμενης γενιάς στη σχεδίαση τους όσο και τους ανθρώπους που γράφουν τις εφαρμογές στην παραγωγή καλύτερου κώδικα. Στην εργασία μας, πέρα από την παρουσίαση της μεθόδου, προχωρούμε και στη δοκιμή της σε δύο νέες εφαρμογές οι οποίες είναι παρόμοιες με αυτές που παρουσιάζονται στο [2], μιας κι αποτελούν κι αυτές κομμάτια υπολογισμών που απαντώνται συχνά σε παράλληλες εφαρμογές, προσφέροντας έτσι επιπλέον αποτελέσματα. Επιπλέον, προτείνουμε και μια νέα συνάρτηση αξιολόγησης των παραγόμενων προβλέψεων, συγκρίνοντάς τη με αυτή που χρησιμοποιείται στη μέθοδο του [2].

Στο μέρος 4 επεκτείνουμε τη μέθοδο που παρουσιάστηκε προηγουμένως επιχειρώντας να μοντελοποιήσουμε, εμπειρικά, το χρόνο εκτέλεσης των ίδιων τεσσάρων εφαρμογών. Επιχειρήσαμε να προβλέψουμε τον χρόνο εκτέλεσης σε δύο διαφορετικά μηχανήματα ως συνάρτηση των αναγκών της εφαρμογής στους πόρους που παρουσιάστηκαν στο μέρος 3. Μια τέτοια ανάλυση έχει ιδιαίτερο ενδιαφέρον, καθώς εκφράζει τις ανάγκες σε πόρους μιας εφαρμογής ως παραμέτρους που επηρεάζουν το χρόνο εκτέλεσής της. Τα αποτελέσματα μας για τα δύο μηχανήματα αναδεικνύουν και τη διαφορά ανάμεσα σε παλιότερα και πιο σύγχρονα μηχανήματα, κάτι που συνεπάγεται και δυσκολία μοντελοποίησης σε παλιότερες συστοιχίες. Επιπρόσθετα κλείνουμε αυτό το μέρος με ορισμένες παρατηρήσεις σχετικά με την επιλογή κατάλληλων μοντέλων καθώς και με μια άσκηση συν-σχεδίασης, επιχειρώντας να εξερευνήσουμε τα οφέλη που θα είχαν οι τέσσερις εφαρμογές από τρεις πιθανές αναβαθμίσεις συστήματος στο μέλλον.

Τέλος, στο μέρος 5, παρουσιάζεται η ίδια ανάλυση με αυτή που έγινε στο μέρος 2 αλλά για μια πιο περίπλοκη εφαρμογή προσομοίωσης σε γράφο. Μια τέτοια ανάλυση αποτελεί πρόκληση καθώς οι αλγόριθμοι που δρουν σε γράφους εξαρτώνται σε μέγιστο βαθμό από τη φύση της εισόδου (γράφου) αλλά κι από τον τρόπο καταμερισμού της εργασίας στους κόμβους

που συμμετέχουν στην παράλληλη επίλυση του προβλήματος. Προτείνουμε δυο διαφορετικούς τρόπους παραμετροποίησης του γράφου ως προς τα χαρακτηριστικά στοιχεία του, είτε λαμβάνοντας υπόψιν το συνολικό πλήθος των ακμών, είτε μελετώντας το πλήθος των ακμών ανά κορυφή, καταλήγοντας σε ένα διαισθητικό συμπέρασμα ότι η πρώτη μέθοδος υπερτερεί. Σε κάθε περίπτωση, μια τέτοια ανάλυση σε προβλήματα με γράφους θα πρέπει να καταλήξει στο ποιοι παράμετροι αυτού περιγράφουν καλύτερα την επίδοση του αλγορίθμου που εκτελεί η εφαρμογή, εξετάζοντας αν συγκεκριμένες ενέργειες που εκτελούνται συχνά σε γράφους, όπως αναζήτηση, χρωματισμός, κ.ά., έχουν κοινές παραμέτρους που επηρεάζουν την απόδοση.

# Βιβλιογραφία


[1] *Top500: The List*. https://www.top500.org/. Ιστοσελίδα. Ημερομηνία πρόσβασης: 13-05-2021.

[2] A. Calotoiu, A. Graf, T. Hoefler κ.ά. . *Lightweight Requirements Engineering for Exascale Co-design. 2018 IEEE International Conference on Cluster Computing (CLUSTER)*, σελίδες 201–211, 2018.

[3] John L. Gustafson. *Amdahl's Law*, σελίδες 53–60. Springer US, Boston, MA, 2011.

[4] Torsten Hoefler, William Gropp, William Kramer και Marc Snir. *Performance modeling for systematic performance tuning. State of the Practice Reports, SC'11*, 2011.

[5] R.F. Barrett, S. Borkar, S.S. Dosanjh κ.ά. . *On the role of co-design in high performance computing. Advances in Parallel Computing*, 24:141–155, 2013.

[6] Dieter Mey, Scott Biersdorf, Christian Bischof κ.ά. . *Score-P: A Unified Performance Measurement System for Petascale Applications*, σελίδες 85–97. 2012.

[7] Markus Geimer, Felix Wolf, Brian J. N. Wylie κ.ά. . *The Scalasca Performance Toolset Architecture. Concurr. Comput.: Pract. Exper.*, 22(6):702–719, 2010.

[8] Henri Casanova, Frédéric Desprez, George S. Markomanolis και Frédéric Suter. *Simulation of MPI Applications with Time-Independent Traces. Concurr. Comput.: Pract. Exper.*, 27(5):1145–1168, 2015.

[9] Sameer S. Shende και Allen D. Malony. *The Tau Parallel Performance System. Int. J. High Perform. Comput. Appl.*, 20(2):287–311, 2006.

[10] Te Phhh, Shirley Moore, Jack Dongarra κ.ά. . *A Portable Programming Interface for Performance Evaluation on Modern Processors. International Journal of High Performance Computing Applications*, 14, 2000.

[11] Laura Carrington, Allan Snavely και Nicole Wolter. *A performance prediction framework for scientific applications. Future Generation Computer Systems*, 22(3):336–346, 2006.

[12] Gabriel Marin και John Mellor-Crummey. *Cross-architecture performance predictions for scientific applications using parameterized models.* τόμος 32, σελίδες 2–13, 2004.

[13] LAURA CARRINGTON, MICHAEL LAURENZANO και ANANTA TIWARI. *CHARACTERIZING LARGE-SCALE HPC APPLICATIONS THROUGH TRACE EXTRAPOLATION. Parallel Processing Letters*, 23(04):1340008, 2013.

[14] Leo Yang, Xiaosong Ma και Frank Mueller. *Cross-Platform Performance Prediction of Parallel Applications Using Partial Execution.* τόμος 2005, σελίδα 40, 2005.

[15] Sergei Shudler, Alexandru Calotoiu, Torsten Hoefler κ.ά. . *Exascaling Your Library: Will Your Implementation Meet Your Expectations?* 2015.

[16] A. Calotoiu, D. Beckinsale, C. W. Earl κ.ά. . *Fast Multi-parameter Performance Modeling. 2016 IEEE International Conference on Cluster Computing (CLUSTER)*, σελίδες 172–181, 2016.

[17] A. Calotoiu, T. Hoefler, M. Poke και F. Wolf. *Using automated performance modeling to find scalability bugs in complex codes. SC '13: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, σελίδες 1–12, 2013.

[18] Nikela Papadopoulou, Georgios Goumas και Nectarios Koziris. *Predictive communication modeling for HPC applications. Cluster Computing*, 20, 2017.

[19] Ian Karlin, Jeff Keasler και Rob Neely. *LULESH 2.0 Updates and Changes.* Τεχνική Αναφορά με αριθμό LLNL-TR-641973, 2013.

[20] A. Kunen, T. Bailey και P. Brown. *KRIPKE - A MASSIVELY PARALLEL TRANSPORT MINI-APP.* 2015.

[21] Charles Ferenbaugh. *PENNANT: An unstructured mesh mini-app for advanced architecture research. Concurrency and Computation: Practice and Experience*, 27, 2014.

[22] Van Emden Henson και Ulrike Meier Yang. *BoomerAMG: A parallel algebraic multigrid solver and preconditioner. Applied Numerical Mathematics*, 41(1):155–177, 2002. Developments and Trends in Iterative Methods for Large Systems of Equations - in memorium Rudiger Weiss.

[23] *Hydrodynamics Challenge Problem, Lawrence Livermore National Laboratory.* Τεχνική Αναφορά με αριθμό LLNL-TR-490254.

[24] Diana Suleimenova, David Bell και Derek Groen. *A generalized simulation development approach for predicting refugee destinations. Scientific Reports*, 7(1):13377, 2017.

[25] Eric Bonabeau. *Agent-based modeling: methods and techniques for simulating human systems. Proceedings of the National Academy of Sciences of the United States of America*, 99 Συππλ 3(Συππλ 3):7280–7287, 2002. 12011407[πμιδ].

[26] Lisandro D. Dalcin, Rodrigo R. Paz, Pablo A. Kler και Alejandro Cosimo. *Parallel distributed computing using Python. Advances in Water Resources*, 34(9):1124–1139, 2011. New Computational Methods and Software Tools.

[27] Andreas Gocht, Robert Schöne και Jan Frenzel. *Advanced Python Performance Monitoring with Score-P*, 2020.