ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

Σχολή Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών

Διπλωματική Εργασία

Acceleration of data compression for the Phylogenetic Likelihood Function using FPGA technology

Συγγραφέας:
Κουτρουλάκης Ευστράτιος

Επιτροπή Διπλωματικής Εργασίας:
Καθηγητής Δόλλας Απόστολος
Καθηγητής Ζερβάκης Μιχαήλ
Επικ.Καθ. Αλαχιώτης Νικόλαος(U Twente)

21 Απριλίου 2021

Περίληψη

Η ανάγκη του ανθρώπου να απαντήσει σε ερωτήματα για το πως κληρονομούνται τα διάφορα χαρακτηριστικά στους οργανισμούς, με ποιο τρόπο δημιουργείται η ποικιλομορφία που υπάρχει στη φύση, να ταξινομήσει τα διάφορα είδη με βάση κάποια κοινά χαρακτηριστικά κ.α. οδήγησαν στην ανάπτυξη του κλάδου της βιολογίας που ασχολείται με την εξέλιξη και τους μηχανισμούς της. Η μελέτη των μοριακών δεδομένων οδήγησε στην πιθανοτική εξέταση της εξέλιξης τους στο χρόνο όπως η μετάλλαξη κάποιας αλληλουχίας DNA σε κάποια άλλη. Με αυτό το τρόπο παράγουμε υποθέσεις για τη πορεία της εξέλιξης των διαφορετικών οργανισμών με συγκεκριμένη πιθανότητα.Τα τελευταία χρόνια η εξέλιξη βιολογικών εργαστηριακών μεθόδων έχει οδηγήσει σε ραγδαία αύξηση των δεδομένων από μοριακές ακολουθίες.Το γεγονός αυτό αυξάνει τα διαθέσιμα δεδομένα που μπορούμε να επεξεργαστούμε για να βγάλουμε χρήσιμα συμπεράσματα.Ταυτόχρονα με τις δυνατότητες που ανοίγει η όλο και μεγαλύτερη συσσώρευση μοριακών δεδομένων δημιουργεί και μεγαλύτερες απαιτήσεις μνήμης για την επεξεργασία τους.

Η εργασία εστιάζει στο πρόβλημα του μεγαλύτερου ρυθμού συσσώρευσης μοριακών δεδομένων προς επεξεργασία σε σχέση με το ρυθμό αύξησης της διαθέσιμης ποσότητας μνήμης.Μελετά τον υπολογισμό της συνάρτησης φυλογενετικής πιθανοφάνειας(Phylogenetic Likelihood Function-PLF), η οποία βασίζεται στο κριτήριο της μέγιστης πιθανοφάνειας.Με σκοπό τη μείωση των απαιτήσεων μνήμης σχεδιάζει κατάλληλο λογισμικό το οποίο διαχειρίζεται τους επανυπολογισμούς δεδομένων που απαιτούνται λόγω της αποθήκευσης ενός μόνο μέρους των δεδομένων στη μνήμη.Ακόμα σχεδιάζεται κατάλληλος επιταχυντής για την εκτέλεση των υπολογισμών μιας κλήσης της συνάρτησης PLF. Για την υλοποίηση του επιταχυντή χρησιμοποιείται σαν αλγόριθμος αναφοράς ο αλγόριθμος που υλοποιεί το πρόγραμμα φυλογενετικών αναλύσεων RAxML ενώ υλοποιείται και ένας επιπλέον επιταχυντής με δενδρική δομή βάθους μεγαλύτερη του ενός με σκοπό τη συμπίεση των δεδομένων που αποθηκεύονται στη μνήμη. Οι επιταχυντές είναι υλοποιημένοι σε αναδιατασσόμενη λογική (Field Programmable Gate Array-FPGA).

Περιεχόμενα

Κεφάλαιο 1

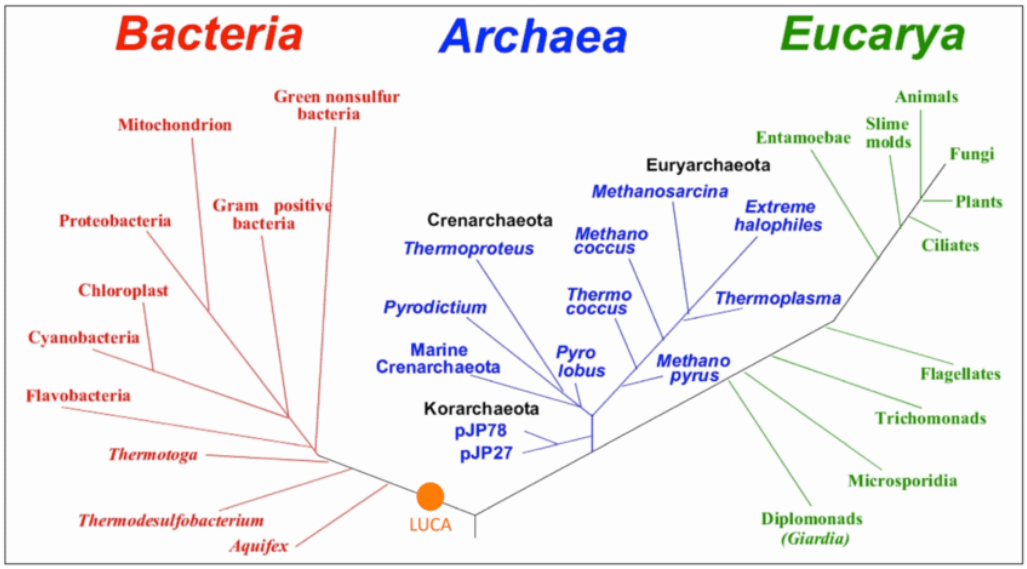
Εισαγωγή

1.1 Εισαγωγή στη μελέτη της εξελικτικής διαδικασίας

Η διαδικασία της εξέλιξης είναι αναπόσπαστο κομμάτι της φύσης.Η εξέλιξη με τους διάφορους μηχανισμούς είναι υπεύθυνη για τις αξιοπρόσεκτες ομοιότητες αλλά και για τη βιοποικιλότητα που παρατηρούμε στη φύση.Μέσα από το σπουδαίο έργο του "Η καταγωγή των ειδών" ο Κάρολος Δαρβίνος απέδειξε με πλήθος στοιχείων και συμπερασμάτων ότι όλα τα υπάρχοντα είδη στο περιβάλλον προέρχονται από κοινούς προγόνους και οι νέες μορφές ζωής προκύπτουν με τη δράση διεργασιών τροποποίησης πάνω σε προϋπάρχουσες.Η επικράτηση συγκεκριμένων ειδών και χαρακτηριστικών σε κάθε είδος είναι αποτέλεσμα του μηχανισμού της φυσικής επιλογής που δρα πάνω σε όλους τους οργανισμούς. Στη πορεία της εξέλιξης δημιουργούνται ποικίλες αλλαγές στις ιδιότητες του πληθυσμού σε κάθε είδος. Το γεγονός αυτό οδηγεί στην επικράτηση με τη πάροδο του χρόνου των συγκεκριμένων χαρακτηριστικών που συμβάλουν στη προσαρμογή στο περιβάλλον.Η συσσώρευση όλων αυτών των αλλαγών με το πέρασμα των γενεών έχει σαν αποτέλεσμα τη δημιουργία νέων ειδών στη φύση και εξαφάνιση των ειδών με τα χαρακτηριστικά εκείνα που δεν τους δίνουν τη δυνατότητα προσαρμογής στο περιβάλλον.Συμπερασματικά η εξέλιξη είναι η διαδικασία με την οποία προκύπτουν νέες μορφές ζωής από τροποποιημένες προϋπάρχουσες με τη δράση συγκεκριμένων διεργασιών.Στην εικόνα 1.1 απεικονίζεται το καθολικό φυλογενετικό δέντρο της ζωής με τους τρεις βασικούς τύπους οργανισμών(Βακτήρια,Αρχαιοβακτηρίδια,Ευκαριωτικά) [36].

Οι επιστήμονες ακόμα και πριν από το καιρό του Δαρβίνου προσπαθούσαν να κατασκευάσουν εξελικτικά δέντρα που να αποδίδουν τις σχέσεις εξέλιξης ανάμεσα στους διαφορετικούς οργανισμούς(taxa) [31], έμβιους και μη, και για το σκοπό αυτό χρησιμοποιούσαν τα φαινοτυπικά χαρακτηριστικά, δηλαδή τα χαρακτηριστικά ενός οργανισμού που μπορούν να παρατηρηθούν όπως είναι η μορφή, ο τρόπος που αναπαράγεται κ.α. Με την εξέλιξη της επιστήμης της βιολογίας, την ανάπτυξη της μοριακής βιολογίας, την ανακάλυψη του DNA και των αμινοξέων έγινε πλέον δυνατό να μελετήσουμε τις

εξελικτικές σχέσεις ανάμεσα στους οργανισμούς με βάση μοριακά δεδομένα.Ένας από τους παράγοντες που δημιουργούν τη ποικιλότητα στα άτομα ενός είδους είναι οι μεταλλάξεις.Οι μεταλλάξεις είναι αλλαγές στην αλληλουχία του DNA και αποτελούν ένα από τους σημαντικότερους μηχανισμούς της εξέλιξης καθώς συμβάλλουν στη δημιουργία της γενετικής ποικιλότητας.Οι αλλαγές στο DNA περνάνε μέσω των μηχανισμών της κληρονομικότητας και στους απογόνους ενός οργανισμού.Οι αλλαγές μπορεί να μην έχουν κάποιο αποτέλεσμα, μπορεί να έχουν επιβλαβές αποτέλεσμα για έναν οργανισμό αλλά μπορεί να οδηγήσουν και στη δημιουργία νέων χαρακτηριστικών τα οποία θα παραχθούν από την έκφραση του DNA. Συνεπώς, η μελέτη των μοριακών δεδομένων(αλληλουχιών DNA,πρωτεϊνών) ενός οργανισμού μπορεί να βοηθήσει σημαντικά στη μελέτη των εξελικτικών σχέσεων [36].

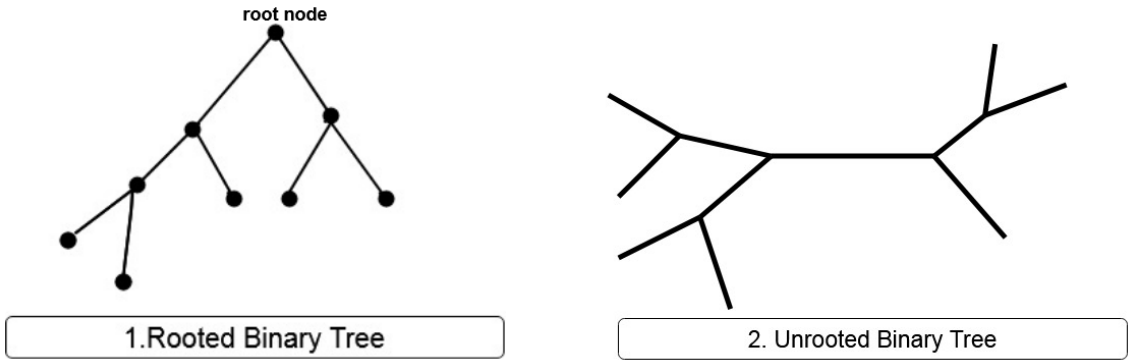


Σχήμα 1.1: Phylogenetic Tree of Life [36]

Με τη μελέτη της εξελικτικής διαδικασίας και τη ταξινόμηση των διαφορετικών ειδών μπορούμε να βγάλουμε χρήσιμα συμπεράσματα που βοηθούν σε διάφορους επιστημονικούς τομείς αλλά και σε τεχνολογικές εφαρμογές [19].Οι αρχές της εξέλιξης βρίσκονται πίσω από τις βελτιώσεις στις καλλιέργειες, την κτηνοτροφία και τις νεότερες μεθόδους γεωργίας.Αξιοποιώντας τις γνώσεις πάνω στη διαδικασία της φυσικής επιλογής αλλά και συνολικά της εξελικτικής βιολογίας κατανοείται γιατί κάποια παράσιτα και ασθένειες προσαρμόζονται στα ήδη υπάρχοντα φυτοφάρμακα και αναπτύσσονται νέα που προστατεύουν τις καλλιέργειες. Ακόμα υπάρχουν αρκετές εφαρμογές για τη προστασία του περιβάλλοντος όπως η χρήση φυτών και βακτηρίων που μπορούν να προσαρμοστούν

σε μολυσμένα περιβάλλοντα με σκοπό την αναπλήρωση της χαμένης βλάστησης και το καθαρισμό του τοξικού περιβάλλοντος.Είδη οργανισμών, από μικρόβια έως θηλαστικά προσαρμόζονται στην κλιματική αλλαγή. Η μελέτη του μηχανισμού προσαρμογής αλλά και του ρυθμού των μεταβολών στη διαδικασία της προσαρμογής μπορεί να βοηθήσει τους ειδικούς επιστήμονες σε θέματα διατήρησης ειδών, να διατυπώσουν τα κατάλληλα μέτρα για την προστασία των ειδών που αντιμετωπίζουν το κίνδυνο της εξαφάνισης.Η γνώση της εξελικτικής πορείας είναι κεντρικής σημασίας ζήτημα και για τη πρόοδο της ιατρικής. Το πεδίο της εξελικτικής ιατρικής είναι αφιερωμένο στη χρήση των αρχών της εξέλιξης με σκοπό τη κατανόηση και θεραπεία ανθρωπίνων νόσων και ασθενειών. Η γνώση των εξελικτικών σχέσεων μεταξύ των ειδών επιτρέπει στους επιστήμονες να επιλέγουν κατάλληλους οργανισμούς για τη μελέτη ασθενειών, όπως ο ιός HIV.

Η φυλογενετική είναι η επιστήμη που μελετά τις εξελικτικές σχέσεις μεταξύ των ειδών. Για να μπορεί να κάνει προβλέψεις σχετικά με αυτές τις σχέσεις κατασκευάζονται φυλογενετικά δέντρα τα οποία συνδέουν τα διάφορα είδη.Κάθε οργανισμός ακολουθεί μια ιστορική πορεία ανάπτυξης.Στις εξελικτικές σχέσεις για μια ομάδα γενετικά συσχετιζόμενων οργανισμών,όπως αυτή προκύπτει από την ανάπτυξή τους, δίνεται ο όρος φυλογένεια.Τα φυλογενετικά δέντρα είναι δυαδικά δέντρα με κόμβους που αναπαριστούν τους διαφορετικούς οργανισμούς και κλαδιά που το μήκος τους αναπαριστά το χρόνο εξέλιξης από το ένα είδος στο εξελικτικά επόμενο.Οι διακλαδώσεις απεικονίζουν τις συνδέσεις των βιολογικών οντοτήτων προς μελέτη.Ο τρόπος με τον οποίο ορίζεται η τοπολογία του δέντρου δείχνει πως έχουν εξελιχθεί τα είδη από τους κοινούς τους προγόνους ως αποτέλεσμα της λειτουργίας των μηχανισμών της εξέλιξης. Τα φυλογενετικά δέντρα αποτελούν υποθέσεις για τις εξελικτικές σχέσεις με συγκεκριμένη πιθανότητα και όχι οριστικά γεγονότα.Στις παρακάτω εικόνες παρουσιάζονται δύο φυλογενετικά δέντρα. Η διαφορά τους βρίσκεται στην ύπαρξη ρίζας.Η ρίζα αντανακλά την ύπαρξη ενός βασικού προγόνου από τον οποίο προέρχονται όλοι οι υπόλοιποι οργανισμοί. Η απουσία της δείχνει ότι η ανάλυση δε κάνει κάποια υπόθεση για κάποιο ξεκάθαρο πρόγονο από τον οποίο προέρχονται τα άλλα είδη.



Σχήμα 1.2: Binary Trees

1.2 Εισαγωγή στο υπολογιστικό πρόβλημα

Τα μοριακά δεδομένα από υπάρχοντες οργανισμούς τα οποία έχουν στη διάθεσή τους οι μελετητές αποτελούν μια καλή πηγή για την εκτίμηση της πιθανοφάνειας της τοπολογίας κάθε φυλογενετικού δέντρου.Τα δεδομένα αυτά συνήθως είναι πολλαπλές ευθυγραμμισμένες ακολουθίες νουκλεοτιδίων του DNA ή αμινοξέων των πρωτεϊνών. Στο DNA περιέχονται οι πληροφορίες που καθορίζουν όλα τα διαφορετικά χαρακτηριστικά ενός οργανισμού.Ακόμη τα μόρια DNA ενός οργανισμού με τη βοήθεια του μηχανισμού της αντιγραφής μεταφέρουν τη γενετική πληροφορία στους απογόνους του οργανισμού.Οι λειτουργίες αυτές του DNA το καθιστούν μια πολύ καλή πηγή πληροφορίας για τη μελέτη της εξέλιξης. Ο βαθμός ομοιότητας της αλυσίδας DNA μεταξύ δύο οργανισμών δείχνει και το βαθμό συγγένειας τους. Ακόμα με τη μελέτη της αλληλουχίας του μπορούμε να μελετήσουμε όλες τις πιθανές μεταλλάξεις οι οποίες αποτελούν τη βάση της γενετικής ποικιλομορφίας και επομένως παίζουν καθοριστικό ρόλο στη διαδικασία της εξέλιξης.Μπορούμε να μελετήσουμε οργανισμούς που δεν έχουν εμφανή φυλογενή χαρακτηριστικά όπως τα βακτήρια ή τα ευκαριωτικά.Όλοι οι παραπάνω παράγοντες συμβάλουν ώστε τεράστιο κομμάτι των φυλογενετικών αναλύσεων να πραγματοποιείται με δεδομένα εισόδου αλυσίδες DNA.

Οι μέθοδοι κατασκευής φυλογενετικών δέντρων κατατάσσονται με ευρεία αποδοχή σε δύο μεγάλες κατηγορίες. Η πρώτη βασίζεται στη κατασκευή πινάκων απόστασης(distance based) και η δεύτερη στη παρουσία χαρακτήρων(character based). Οι πιο κοινές μέθοδοι που ανήκουν στη πρώτη κατηγορία είναι η μη σταθμισμένη μέθοδος ομάδας ζευγών που χρησιμοποιεί αριθμητικούς μέσους όρους(UPGMA) [35] και η Neighbor Joining [40]. Από την άλλη πλευρά έχουμε τη μέθοδο της μέγιστης φειδωλότητας(maximum parsimony) [15] και μέγιστης πιθανοφάνειας(maximum likelihood) [14] οι οποίες εφαρμόζουν πιθανοτική προσέγγιση στη κατασκευή του φυλογενε-

τικού δέντρου. Από τη στιγμή που το πρόβλημα της κατασκευής εξελικτικών δέντρων έχει αναχθεί σε ένα καθαρά μαθηματικό υπολογιστικό πρόβλημα έχουν αναπτυχθεί μια σειρά αλγόριθμοι για τη κατασκευή φυλογενετικών δέντρων.Στη παρούσα μελέτη επικεντρώνουμε στη μέθοδο της Μέγιστης Πιθανοφάνειας. Κριτήριο για τη κατασκευή ενός φυλογενετικού δέντρου είναι η μεγιστοποίηση της πιθανοφάνειας του.Η ανάλυση πραγματοποιείται με τη κατασκευή ενός εκτιμητή ο οποίος παράγει διάφορες υποθέσεις για την εξελικτική πορεία οι οποίες εκφράζονται με τη παραγωγή διαφορετικών τοπολογιών για το φυλογενετικό δέντρο και με βάση κάποιο μοντέλο αξιολογεί αυτές τις υποθέσεις. Η υπόθεση με τη καλύτερη τιμή με βάση το κριτήριο αξιολόγησης προτιμάται.

Ένας μεγάλος αριθμός από μελέτες και πειράματα δείχνουν ότι οι εκτιμητές που χρησιμοποιούν τη μέθοδο της μέγιστης πιθανοφάνειας για τη κατασκευή και αξιολόγηση φυλογενετικών δέντρων μπορούν να κατασκευάσουν δέντρα χρησιμοποιώντας κάποια σύνολα δεδομένων που ανταποκρίνονται στη πραγματικότητα με μεγαλύτερη συχνότητα απ΄ ότι μπορούν άλλες μέθοδοι.Μειονέκτημα της μεθόδου αποτελεί ο υπολογιστικός φόρτος που απαιτεί τόσο σε χρόνο όσο και σε μνήμη.Χαρακτηριστικά σε όλα τα ευρέως χρησιμοποιούμενα προγράμματα φυλογενετικών αναλύσεων που χρησιμοποιούν πιθανοτικές προσεγγίσεις στους υπολογισμούς (είτε με το κριτήριο Μέγιστης Πιθανοφάνειας είτε με Μπαγιεσιανές στατιστικές μεθόδους) η συνάρτηση φυλογενετικής πιθανοφάνειας(Phylogenetic Likelihood Function(PLF)), τυπικά καταναλώνει το 85%-95% τόσο στο συνολικό χρόνο εκτέλεσης όσο και στους πόρους μνήμης [23].

Για τη κατασκευή των φυλογενετικών δέντρων, τα οποία θα ανταποκρίνονται στη πραγματικότητα με μεγάλη ακρίβεια απαιτείται συσσώρευση και χρησιμοποίηση μεγάλης ποσότητας μοριακών δεδομένων.Τα μοριακά δεδομένα για την εξέταση του προβλήματος της κατασκευής φυλογενετικών δέντρων προέρχονται από βιολογικές βάσεις δεδομένων οι οποίες περιέχουν μοριακά δεδομένα όπως αλληλουχίες DNA (GENBANK [7]) ή πρωτεϊνών(SWISSPROT [6]).Η ραγδαία εξέλιξη της μοριακής βιολογίας έχει συμβάλει τις τελευταίες δεκαετίες στη συστηματική μελέτη του γονιδιώματος και των αμινοξέων(βασικό συστατικό των πρωτεϊνών) των διαφορετικών οργανισμών και στη συγκέντρωση μεγάλης ποσότητας μοριακών δεδομένων στις βιολογικές βάσεις οι οποίες ανήκουν σε διεθνείς οργανισμούς και ενημερώνονται διαρκώς.

Η αύξηση των μοριακών δεδομένων που μπορούμε να αξιοποιήσουμε για κάθε οργανισμό ξεχωριστά αλλά και η προσθήκη νέων οργανισμών στη μελέτη της εξελικτικής διαδικασίας μεγαλώνουν με πολύ μεγαλύτερο ρυθμό από ότι το μέγεθος των μνημών RAM καθώς και από τη ταχύτητα πρόσβασης στα δεδομένα τους.Για μεγάλης κλίμακας φυλογενετική ανάλυση που ανταποκρίνεται σε είσοδο χιλιάδων οργανισμών και αλυσίδων DNA που αντιστοιχίζονται σε πραγματικά γονίδια με μήκος εκατομμύρια νουκλεοτίδια ο κύριος περιοριστικός παράγοντας στο χρόνο εκτέλεσης είναι η ανεπάρκεια μνήμης.Ο τύπος που δίνει τις απαιτήσεις μνήμης για μια ανάλυση με n taxa στην είσοδο, καθένα από τα οποία έχει μήκος ακολουθίας s sites είναι $R = (n-2) * 8 * 16 * s$.Η εξίσωση αυτή θα αναλύεται διεξοδικά στην ενότητα 2.6.Αν θέσουμε σε ένα πείραμα n

= 1481 και s = 20000000 απαιτείται περίπου 1TB μνήμης και 648 πυρήνες επεξεργασίας για τον υπολογισμό της ανάλυσης σε 48 ώρες [23].

1.3 Πρόγραμμα Φυλογενετικών Αναλύσεων **RAxML**

Το πρόγραμμα φυλογενετικών αναλύσεων RAxML(randomized axelerated maximum likelihood for high performance computing) είναι ένα ευρέως χρησιμοποιούμενο πρόγραμμα για φυλογενετικές αναλύσεις οι οποίες στηρίζονται στο κριτήριο της μέγιστης πιθανοφάνειας.Δέχεται ως είσοδο ένα σύνολο από αλληλουχίες είτε DNA είτε πρωτεϊνών που προέρχονται από υπαρκτούς οργανισμούς και παράγει ένα σύνολο φυλογενετικών δέντρων χρησιμοποιώντας ευριστικούς αλγόριθμους αναζήτησης μέσα στο σύνολο των πιθανών δέντρων που μπορούν να παραχθούν.Οι ευριστικοί αλγόριθμοι του RAxML επιλέγουν δέντρα τα οποία βελτιώνουν τη παράμετρο της συνολικής πιθανοφάνειας από κάποιο αρχικό δέντρο που περιλαμβάνει το σύνολο των διαφορετικών ακολουθιών και το οποίο κατασκευάζεται με το κανόνα των ελάχιστων μεταβολών(parsimony).Το RAxML επιδεικνύει εξαιρετική απόδοση τόσο στην ακρίβεια των υπολογισμών όσο και στη ταχύτητα και θεωρείται από πολλούς ως τεχνολογία αιχμής για γρήγορες φυλογενετικές αναλύσεις μέγιστης πιθανοφάνειας με μεγάλα σύνολα δεδομένων στην είσοδο και αποτελέσματα μεγάλης ακρίβειας [55].

Το RAxML παρέχει μεγάλο επίπεδο λεπτομέρειας(fine-grain στη συνάρτηση πιθανοφάνειας για συστήματα με πολλούς πυρήνες μέσω της έκδοσης που βασίζεται σε PThreads αλλά και μεγάλο επίπεδο αφαίρεσης(coarse-grain) μέσω του MPI(Message Passing Interface) για τις διαφορετικές ανεξάρτητες αναζητήσεις πάνω σε κάποιο δέντρο. Ακόμα προσφέρει συνδυασμό των παραπάνω υλοποιήσεων παραλληλισμού με υβριδική έκδοση MPI/PThreads [37].Την ίδια στιγμή το RAxML υποστηρίζει τις επεκτάσεις SSE3, AVX και AVX2 της x86 αρχιτεκτονικής εντολών [43]. Με τις επεκτάσεις αυτές μπορεί να υποστηρίξει σύνθετες εντολές που συγχωνεύουν πράξεις πρόσθεσης και πολλαπλασιασμού επιταχύνοντας τις πράξεις για τη κατασκευή φυλογενετικού δέντρου ελάχιστων μεταβολών και τον υπολογισμό της πιθανοφάνειας του.

1.4 Συνεισφορά Μελέτης

Η παρούσα διπλωματική έχει σκοπό να αντιμετωπίσει το πρόβλημα των μεγάλων απαιτήσεων σε πόρους μνήμης για τη πραγματοποίηση των απαραίτητων υπολογισμών που χρειάζεται μια φυλογενετική ανάλυση.Σχεδιάζει και υλοποιεί σύστημα διαχείρισης των PLF κλήσεων που παράγονται από το RAxML, με τη βοήθεια του οποίου υλοποιούνται οι φυλογενετικές αναλύσεις παρότι μειώνεται η ποσότητα των δεδομένων που αποθηκεύονται στη μνήμη.Η αρχιτεκτονική του συστήματος που παρουσιάζει η εργασία στηρίζεται στην αποθήκευση ενός μέρους των δεδομένων και τον υπολογισμό του

υπόλοιπου μέρους κάθε φορά που απαιτείται από κάποιο αίτημα.Για κάθε κλήση της φυλογενετικής συνάρτησης,εκτελεί ένα νέο αριθμό κλήσεων της PLF που απαιτούνται για τον υπολογισμό δεδομένων που δε βρίσκονται στη μνήμη και χρειάζονται για τον υπολογισμό της κλήσης που δέχθηκε στην είσοδο το σύστημα.

Οι υπολογισμοί των κλήσεων της PLF του συστήματος πραγματοποιούνται από επιταχυντή που υλοποιήθηκε σε αναδιατασσόμενη λογική(FPGA) με τη βοήθεια των εργαλείων της Xillinx.Με αυτό το τρόπο ανταλλάσσεται ποσότητα μνήμης με μια άλλη ποσότητα από υπολογισμούς.Κριτήριο ισσοροπίας σε αυτή την αναταλλαγή είναι η επιβάρυνση στο χρόνο εκτέλεσης σε σχέση με το συνολικό χρόνο εκτέλεσης της φυλογενετικής ανάλυσης.Η ποσότητα των δεδομένων που είναι δυνατόν να μείνει εκτός μνήμης αυξάνεται με την υλοποίηση επιταχυντή ο οποίος έχει δενδρική δομή με βάθος μεγαλύτερο του ενός και ενσωματώνει εσωτερικά υπολογισμούς για δεδομένα που σε διαφορετική περίπτωση θα έπρεπε είτε να βρίσκονται στη μνήμη είτε να υπολογιστούν από ξεχωριστές κλήσεις της PLF.Στη συνεισφορά της μελέτης συμπεριλαμβάνεται και η περιγραφή της υλοποίησης των επιταχυντών μέσα από τα εργαλεία της Xillinx και πιο συγκεκριμένα του High-Level-Synthesis Edition 2018.3 από τη σειρά εργαλείων Vivado Design Suite και το περιβάλλον ανάπτυξης SDSoC^{TM} από τη σειρά εργαλείων SDx IDE 2018.3.

1.5 Δομή Διπλωματικής Εργασίας

Στο κεφάλαιο 2 η εργασία μελετά τη κατασκευή και αξιολόγηση φυλογενετικών δέντρων με τη μέθοδο της μέγιστης πιθανοφάνειας.Ακόμα παρουσιάζεται η υλοποίηση των υπολογισμών της φυλογενετικής συνάρτησης πιθανοφάνειας από το πρόγραμμα φυλογενετικών αναλύσεων RAxML. Στο 3^o κεφάλαιο παρουσιάζονται μια σειρά από ερευνητικές εργασίες οι οποίες αφορούν είτε τη μείωση της κατανάλωσης πόρων μνήμης είτε την επιτάχυνση της εκτέλεσης της φυλογενετικής συνάρτησης πιθανοφάνειας.Στη συνέχεια στο 4^o κεφάλαιο αναλύεται η λειτουργία του συστήματος monitor για τη διαχείριση των κλήσεων της PLF με σκοπό την εξοικονόμηση μνήμης ενώ στο 5^o κεφάλαιο παρουσιάζεται η προτεινόμενη σχεδίαση για τον επιταχυντή του συστήματος μνήμης. Το 6^o κεφάλαιο αφιερώνεται στην υλοποίηση της προτεινόμενης αρχιτεκτονικής με χρήση των εργαλείων της Xillinx.Τέλος το κεφάλαιο 7 παρουσιάζει τα αποτελέσματα των πειραμάτων που πραγματοποιήθηκαν για την αξιολόγηση της προτεινόμενης λύσης, ενώ στον επίλογο παρουσιάζονται συνοπτικά προτάσεις για μελλοντικές επεκτάσεις.

Κεφάλαιο 2

Κατασκευή και Αξιολόγηση Φυλογενετικών Σχέσεων

Στο κεφάλαιο αυτό αναλύουμε τη πορεία κατασκευής ενός φυλογενετικού δέντρου ,μέσα από την υλοποίηση φυλογενετικής ανάλυσης, βασιζόμενοι στην ευθυγράμμιση πολλαπλών αλληλουχιών(**Multiple Sequence Alignment-MSA**). Παρουσιάζουμε τα διαφορετικά μοντέλα εξέλιξης των μοριακών δεδομένων μέσω της νουκλεοτιδικής υποκατάστασης και το κριτήριο της μέγιστης πιθανοφάνειας το οποίο βελτιστοποιεί τη πιθανοφάνεια των μοριακών δεδομένων που έχουν παρατηρηθεί δεδομένης μιας δενδρικής τοπολογίας και ενός μοντέλου νουκλεοτιδικής εξέλιξης. Τέλος παρουσιάζουμε τις απαιτήσεις σε μνήμη που χρειάζονται οι υπολογισμοί του κριτηρίου μέγιστης πιθανοφάνειας.

2.1 Ευθυγράμμιση Αλληλουχιών

Στην ενότητα αυτή παρουσιάζουμε τη δομή των δεδομένων εισόδου μιας φυλογενετκής ανάλυσης.Αυτά αποτελούνται από n διαφορετικές ακολουθίες DNA μήκους m που αντιστοιχίζονται σε n διαφορετικούς οργανισμούς και οι οποίες σχηματίζουν μια δομή ευθυγραμμισμένων ακολουθιών.

Μια ακολουθία(***sequence***) DNA από έναν οργανισμό είναι μια συμβολοσειρά που αναπαριστά με απλό τρόπο τη μοριακή δομή του DNA του οργανισμού.Αυτή η αναπαράσταση χρησιμοποιείται ευρέως σε πολλές εφαρμογές καθώς αγνοεί τις φυσικές και χημικές ιδιότητες των μορίων του DNA.Το ίδιο ισχύει και για τις ακολουθίες του RNA ή των πρωτεϊνών.Για το DNA οι χαρακτήρες της συμβολοσειράς παίρνουν τιμές από το αλφάβητο (A,C,G,T) που αναπαριστά τις αντίστοιχες νουκλεοτιδικές βάσεις (αδενίνη,κυτοσίνη,γουανίνη,θυμίνη) από τις οποίες δομείται το μακρομόριο του DNA.

Μια φυλογενετική ανάλυση χρειάζεται να γνωρίζει για τις ακολουθίες εισόδου ποιοι

χαρακτήρες είναι ομόλογοι, δηλαδή ποιοι χαρακτήρες είναι δυνατόν να συνδέονται εξελικτικά μεταξύ τους με κάποιο κοινό πρόγονο.Η ευθυγράμμιση αλληλουχιών (sequence alignment) είναι ένας βολικός τρόπος για να παραθέσουμε αλληλουχίες DNA,RNA ή πρωτεϊνών με σκοπό να εντοπίσουμε περιοχές ομοιότητας μεταξύ των αλληλουχιών οι οποίες μπορεί να είναι συνέπεια εξελικτικών σχέσεων μεταξύ των ακολουθιών.Με αυτή τη μέθοδο αντιπαραβάλουμε τις αντίστοιχες θέσεις στις διαφορετικές αλληλουχίες.Αν δυο χαρακτήρες σε μια τέτοια στήλη είναι διαφορετικοί έχουμε αναντιστοιχία.

Η εξέλιξη των μορίων του DNA πραγματοποιείται μέσω των μεταλλάξεων που περιλαμβάνουν την αντικατάσταση μιας νουκλεοτιδικής βάσης από μια άλλη αλλά και από εισαγωγές και διαγραφές νουκλεοτιδικών βάσεων στην αλυσίδα.Οι εισαγωγές και οι διαγραφές δημιουργούν ακολουθίες DNA που μπορεί να προέρχονται από κοινό πρόγονο αλλά καταλήγουν να έχουν διαφορετικό μήκος.Η ευθυγράμμιση των ακολουθιών γίνεται με σκοπό όλες οι ακολουθίες να έχουν το ίδιο μήκος.Συνεπώς για να αναπαραστήσουμε στην αλληλουχία του DNA τις εισαγωγές και διαγραφές, προσθέτουμε το χαρακτήρα του κενού " - " .

Η ευθυγράμμιση παράγει ένα πίνακα mxn.Ο πίνακας αυτός αποτελείται από m γραμμές που αντιστοιχίζονται στους διαφορετικούς οργανισμούς και n στήλες που αντιστοιχίζονται στους χαρακτήρες των ακολουθιών.Όλοι οι ομόλογοι χαρακτήρες βρίσκονται στην ίδια στήλη.Αν $m > 2$ έχουμε ευθυγράμμιση πολλαπλών ακολουθιών(*Multiple Sequence Alignment*(*MSA*)).Οι στήλες της ευθυγράμμισης αναφέρονται ως sites.Η ευθυγράμμιση αλληλουχιών είναι η βάση της ανάλυσης των εξελικτικών σχέσεων μεταξύ των μοριακών αλληλουχιών καθώς ορίζει τις ομοιότητες και τις διαφορές στα παρατηρήσιμα δεδομένα. Στο σχήμα 2.1 μπορούμε να παρατηρήσουμε μια ευθυγράμμιση αλληλουχιών για 5 διαφορετικές αλληλουχίες.Μια φυλογενετική ανάλυση παραθέτει και συγκρίνει πολλαπλές αλληλουχίες.

Sequence1	**−TCAGGA−TGAAC−−−−**
Sequence2	**ATCACGA−TGAACC−−−**
Sequence3	**ATCAGGAATGAATCC−−**
Sequence4	**−TCACGATTGAATCGC−**
Sequence5	**−TCAGGAATGAATCGCM**

Σχήμα 2.1: Multiple Sequence Alignment.

Το σύνολο των αλληλουχιών εισόδου θεωρούμε ότι συνδέονται μεταξύ τους με εξελικτικές σχέσεις. Από το αποτέλεσμα μιας MSA, μπορεί να συναχθεί ομολογία αλληλουχιών δηλαδή δύο αλληλουχίες να συνδέονται με κοινό πρόγονο και να διεξαχθεί

φυλογενετική ανάλυση για να εκτιμηθεί η κοινή εξελικτική προέλευση των αλληλουχιών.Επομένως οι ευθυγραμμίσεις πολλαπλών αλληλουχιών μπορούν να χρησιμοποιηθούν ως είσοδος φυλογενετικής ανάλυσης για τη δημιουργία ενός φυλογενετικού δέντρου που αναπαριστά γραφικά τις εξελικτικές σχέσεις.Οι οπτικές απεικονίσεις της ευθυγράμμισης όπως στο σχήμα 2.1 δείχνουν γεγονότα μεταλλάξεων όπως είναι οι σημειακές μεταλλάξεις(αλλαγή ενός μόνο νουκλεοτιδίου) που εμφανίζονται ως διαφορετικοί χαρακτήρες σε μία στήλη μονής ευθυγράμμισης και μεταλλάξεις εισαγωγής ή εξάλειψης που εμφανίζονται ως παύλες σε μία ή περισσότερες από τις ακολουθίες στην ευθυγράμμιση.

Για την υλοποίηση της ευθυγράμμισης πολλαπλών αλληλουχιών έχουν αναπτυχθεί μια σειρά από εργαλεία [13].Ευρέως χρησιμοποιούμενες σειρές εργαλείων είναι οι CLUSTALW [47], MAFFT [28], MUSCLE [12], T-COFFEE [45], PROBCONS [34].

2.2 Υλοποίηση Φυλογενετικής Ανάλυσης

2.2.1 Κριτήριο Μέγιστης Πιθανοφάνειας

Όπως έχει περιγραφεί και στην εισαγωγή, η εργασία μελετά τους υπολογισμούς που πραγματοποιούνται με τη μέθοδο της μέγιστης πιθανοφάνειας για τη κατασκευή φυλογενετικών δέντρων.Η μέθοδος της μέγιστης πιθανοφάνειας προσφέρει μεγάλη ακρίβεια στην ανάλυση [20]. Αυτή η μέθοδος χρησιμοποιείται σήμερα από διαδεδομένα προγράμματα φυλογενετικών αναλύσεων όπως το RAxML [44].Ένα από τα βασικά του πλεονεκτήματα θεωρείται η στατιστική σύγκριση μεταξύ τοπολογιών και η δυνατότητα δοκιμής διαφορετικών υποθέσεων για τη τοπολογία του φυλογενετικού δέντρου.

Θεωρούμε ένα σύνολο από διαφορετικούς οργανισμούς οι οποίοι συνδέονται μεταξύ τους με κοινούς προγόνους.Η τοπολογία του φυλογενετικού δέντρου αποτελεί την άγνωστη παράμετρο που θέλουμε να υπολογίσουμε η οποία βελτιστοποιεί τη πιθανοφάνεια να παρατηρήσουμε τα υπάρχοντα δεδομένα που έχουμε στη διάθεσή μας.Κάθε οργανισμός αντιστοιχίζεται μέσα στο σύνολο δεδομένων σε κάποια γραμμική αλληλουχία DNA.Έστω $P(O|T)$ η πιθανότητα να παρατηρήσουμε τα μοριακά δεδομένα O που έχουμε στη διάθεση μας δεδομένης μια συγκεκριμένης τοπολογίας T του φυλογενετικού δέντρου.Η φυλογενετική ανάλυση που στηρίζεται στο κριτήριο της μέγιστης πιθανοφάνειας γνωρίζοντας τα παρατηρούμενα μοριακά δεδομένα παίρνει μια υπόθεση για την τοπολογία του φυλογενετικού δέντρου και υπολογίζει τη πιθανοφάνεια $L(T|O)$, δηλαδή τη πιθανοφάνεια οι σχέσεις εξέλιξης που εκφράζει η τοπολογία T να οδηγούν στα πραγματικά δεδομένα O.Το κριτήριο μέγιστης πιθανοφάνειας εκτιμάει ως προτιμότερη τη τοπολογία που δίνει μεγαλύτερη πιθανοφάνεια σε σχέση με τις υπόλοιπες που έχουν υπολογιστεί.Ποικίλα μοντέλα νουκλεοτιδικής υποκατάστασης μπορούν να εφαρμοστούν για να περιγράψουν την βασική διαδικασία των νουκλεοτιδικών αλλαγών στο DNA στην οποία στηρίζεται η ύπαρξη πολλαπλών παρόμοιων αλληλουχιών.Το μοντέλο και

οι παράμετροι του μπορούν να επιλεγούν από τον ερευνητή ή να εκτιμηθούν από το σύνολο δεδομένων.

Για τη πραγματοποίηση της φυλογενετικής ανάλυσης αρχικά απαιτείται να ορίσουμε το τρόπο με τον οποίο κατασκευάζονται διαφορετικές τοπολογίες.Επιπλέον χρειάζεται να προσδιορίσουμε τα διαφορετικά μοντέλα νουκλεοτιδικής υποκατάστασης.Οι νουκλεοτιδικές υποκαταστάσεις μαζί με τις εισαγωγές και τις διαγραφές παράγουν τις διαφορετικές αλληλουχίες των απογόνων από ένα πρόγονο.Τέλος ορίζουμε τη συνάρτηση φυλογενετικής πιθανοφάνειας που υπολογίζει τη πιθανοφάνεια κάθε φυλογενετικού δέντρου.

2.2.2 Τοπολογίες Φυλογενετικού Δέντρου

Η υποκατάσταση κάποιας βάσης σε μια αλυσίδα DNA είναι η αντικατάσταση της βάσης αυτής με κάποια άλλη μέσα σε κάποιο χρονικό διάστημα.Το μοντέλο υποκατάστασης έχει την ιδιότητα της χρονικής αντιστρεψιμότητας [48].Αυτό σημαίνει ότι η διαδικασία της υποκατάστασης βάσεων έχει την ίδια πιθανότητα είτε προχωρήσουμε μπροστά είτε πίσω στο χρόνο. Η ιδιότητα αυτή φαίνεται ξεκάθαρα και από τη ισχύουσα σχέση για τις υποκαταστάσεις:

$$\pi_i P_{ij}(t) = \pi_j P_{ji}(t) \tag{2.1}$$

Η ιδιότητα της χρονικής αντιστρεψιμότητας επιτρέπει να αμελήσουμε ποιος κόμβος είναι πρόγονος και ποιος απόγονος από τη στιγμή που άλλες παράμετροι όπως ο αριθμός των υποκαταστάσεων παραμένουν σταθεροί.Με τη χρονική αντιστρεψιμότητα το φυλογενετικό δέντρο μπορεί να ορίσει τυχαία ρίζα σε οποιοδήποτε σημείο αλλά και να επανακαθορίσει τη ρίζα του βασισμένο σε καινούρια γνώση δημιουργώντας καινούριες υποθέσεις τοπολογίας για αξιολόγηση [14].

Στην αρχή της ανάλυσης τίθεται μια υπόθεση για μια τυχαία τοπολογία διαλέγοντας αυθαίρετα κάποιο σημείο πάνω σε κάποιο τυχαίο κλαδί ως ρίζα του δέντρου. Τα $\mathbf{n}$ υπάρχοντα είδη για τα οποία έχουμε δείγμα DNA τοποθετούνται στα φύλλα του δέντρου, ενώ οι $n-2$ εσωτερικοί κόμβοι του δέντρου αντιστοιχούν σε είδη που έχουν εξαφανιστεί στο χρόνο με τη διαδικασία της φυσικής επιλογής και αποτελούν τους κοινούς προγόνους.Στη συνέχεια πραγματοποιούνται αναδιατάξεις με κριτήριο τη μεγιστοποίηση της πιθανοφάνειας της τοπολογίας του φυλογενετικού δέντρου.Τα φυλογενετικά δέντρα όπως έχει αναφερθεί είναι δυαδικά δέντρα. Αυτό συνεπάγεται ότι το βασικό μοτίβο που επαναλαμβάνεται στο δέντρο είναι ένας προγονικός κόμβος με δυο κόμβους παιδιά.Η πρόβλεψη που κάνουμε για την αλληλουχία του DNA σε κάθε κόμβο με τον υπολογισμό της πιθανοφάνειας εξαρτάται άμεσα από τους κόμβους που συνδέονται σε αυτόν ως απόγονοι, όπως θα φανεί και στη συνέχεια από το τρόπο που υπολογίζεται η πιθανοφάνεια.Συνεπώς ο αλγόριθμος με τον οποίο γίνεται η αναδιάταξη της τοπολογίας παίζει καθοριστικό ρόλο για τον υπολογισμό της πιθανοφάνειας.

2.2.3 Υπολογισμός Πιθανοφάνειας Φυλογενετικού Δέντρου

Υποθέτουμε ότι οι μεταλλάξεις στην ακολουθία του DNA συμβαίνουν με τυχαίο τρόπο, δηλαδή με τρόπο που δε μπορούμε μέχρι στιγμής να προβλέψουμε και να τις προσδιορίσουμε εξαρχής με βάση κάποιους νόμους. Χρησιμοποιούμε επομένως ένα πιθανοτικό μοντέλο για την υποκατάσταση των νουκλεοτιδίων και με βάση αυτό υπολογίζουμε τις πιθανοφάνειες των εξελικτικών δέντρων.

Για να υπολογίσουμε τη πιθανοφάνεια ενός δέντρου αρκεί να υπολογίσουμε τη πιθανοφάνεια για την εξέλιξη των αλληλουχιών DNA στο δοσμένο δέντρο. Συνεπώς η εκτίμηση της πιθανοφάνειας ενός δέντρου μπορεί να μετατραπεί σε πρόβλημα υπολογισμού των πιθανοτήτων μια αλληλουχία S_1 να εξελιχθεί σε μια άλλη S_2 σε κάποιο χρονικό διάστημα t. Οι αλληλουχίες αναπαριστώνται γραφικά από τους κόμβους του δέντρου και το χρονικό διάστημα από το μήκος στα κλαδιά που συνδέουν τους κόμβους.

Για τον υπολογισμό της πιθανοφάνειας μιας αλληλουχίας υποθέτουμε ότι οι αλλαγές στις διαφορετικές θέσεις της συγκεκριμένης αλληλουχίας είναι πιθανοτικά ενδεχόμενα ανεξάρτητα μεταξύ τους.Η υπόθεση της ανεξάρτητης εξέλιξης κάθε νουκλεοτιδίου συνεπάγεται ότι οι πιθανότητες ενός συνόλου αλληλουχιών σε ένα δέντρο μπορούν να υπολογιστούν ανα θέση ενώ το γινόμενο αυτών των πιθανοτήτων δίνει τη συνολική πιθανότητα της αλληλουχίας.

Η συνάρτηση $P_{ij}(t)$, i,j= 1,...,4, όπου i,j μεταβλητές που αναπαριστούν τις 4 διαφορετικές βάσεις A,T,C,G , επιστρέφει τη πιθανότητα η βάση i να υποκατασταθεί από τη βάση j μετά από χρόνο t που παρήλθε.

Για την υποκατάσταση των νουκλεοτιδίων στην αλυσίδα του DNA παίρνουμε τη παραδοχή ότι περιγράφονται από αλυσίδες **Markov** [18]. Μια αλυσίδα Markov περιγράφει μεταβολές μέσα σε ένα πεπερασμένο σύνολο καταστάσεων(στη περίπτωση μας το σύνολο των διαφορετικών νουκλεοτιδίων A,T,C,G) με τυχαίο τρόπο, ενώ δεν διατηρεί μνήμη για τις προηγούμενες μεταβολές. Δηλαδή κάθε μετάβαση εξαρτάται μόνο από τη παρούσα κατάσταση και καθόλου από τις μεταβάσεις που έχουν προηγηθεί. Στο πεδίο που μελετάμε η πιθανότητα υποκατάστασης μιας νουκλεοτιδικής βάσης i που υπάρχει σε κάποια θέση k της αλυσίδας σε κάποια χρονική στιγμή t_0 από μια άλλη j τη στιγμή t_1 εξαρτάται μόνο από το γεγονός ότι τη στιγμή t_0 έχουμε τη βάση i στη συγκεκριμένη θέση. Ακόμα παίρνουμε τον περιορισμό ότι αναφερόμαστε σε χρονικά ομογενείς αλυσίδες **Markov** όπου οι πίνακες των πιθανοτήτων μετάβασης παραμένουν σταθεροί με τη πάροδο του χρόνου, δηλαδή σε όλο το φυλογενετικό δέντρο, χωρίς αυτό φυσικά να σημαίνει ότι παραμένουν σταθερές και οι πιθανότητες να έχουμε τη χρονική στιγμή n σε κάποια θέση μια νουκλεοτιδική βάση i.

Τέλος υποθέτουμε ότι ισχύει η ιδιότητα της αντιστρεψιμότητας(reversibility) των ρυθμών υποκατάστασης [18], δηλαδή ο ρυθμός αλλαγής από μια βάση a σε μια βάση b είναι ίδιος σε ένα δεδομένο χρονικό διάστημα με το ρυθμό πραγματοποίησης της ανάστροφης διαδικασίας. Από την ιδιότητα της χρονικής αντιστρεψιμότητας προκύπτει

ότι η πιθανοφάνεια του φυλογενετικού δέντρου είναι εντελώς ανεξάρτητη από τη ρίζα του. Επομένως μπορούμε να υπολογίσουμε τη μέγιστη πιθανοφάνεια πάνω σε δέντρα χωρίς ρίζα με αυθαίρετη επιλογή ρίζας. Η υπόθεση της χρονικής αντιστρεψιμότητας συμβάλει σημαντικά στην απλοποίηση των υπολογισμών.

Η $\vec{L}_{S_k}^{(P)}(i)$ υπολογίζει τη πιθανοφάνεια στο κόμβο P στη θέση i να παρατηρήσουμε το νουκλεοτίδιο S_k. Για κάθε θέση λοιπόν μιας ακολουθίας υπολογίζουμε 4 τιμές που αντιστοιχούν στις πιθανοφάνειες να υπήρχε A,T,C,G.Αν ο κόμβος είναι στα φύλλα του δέντρου, τότε η $\vec{L}_{S_k}^{(P)}(i)$ θα είναι 0 για κάθε κατάσταση εκτός από αυτή που έχουμε παρατηρήσει η οποία θα είναι ίση με 1.Στη περίπτωση που ο κόμβος δε βρίσκεται στα φύλλα και έχει απογόνους τους κόμβους L,R με αντίστοιχα μήκη σε αριστερό και δεξί κλαδί b_l,b_r, πίνακες πιθανοτήτων υποκατάστασης $P(b_l)$ και $P(b_r)$ και διανύσματα πιθανότητας των απογόνων $\vec{L}^{(l)}$, $\vec{L}^{(r)}$ η σχέση που υπολογίζει τη πιθανοφάνεια είναι εξής:

$$\vec{L}_{S_p}^{(P)}(i) = (\sum_{S_i=A}^{T} P_{S_p S_i}(b_l) L_{S_i}^{(L)}(i))(\sum_{S_i=A}^{T} P_{S_p S_i}(b_r) L_{S_i}^{(R)}(i)) \tag{2.2}$$

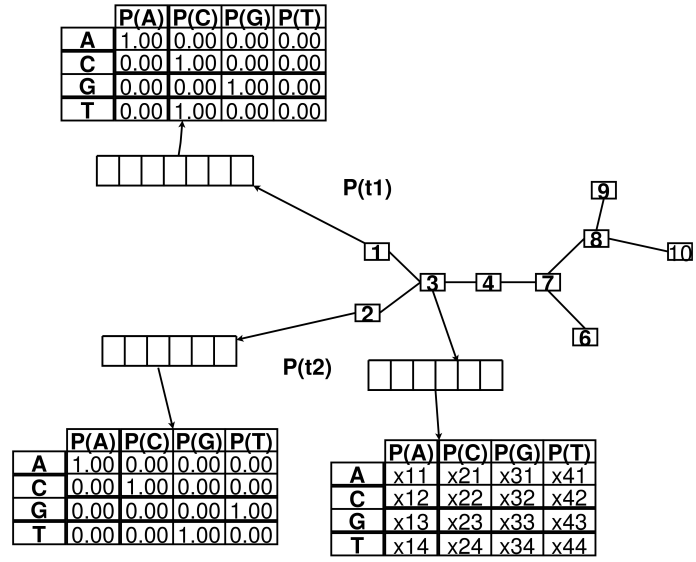
Γνωρίζοντας τη σχέση που υπολογίζει το βασικό πρότυπο στο φυλογενετικό δέντρο μπορούμε να προσδιορίσουμε τη μέθοδο για τον υπολογισμό της συνολικής πιθανοφάνειας ενός φυλογενετικού δέντρου.Η πιθανοφάνεια υπολογίζεται με βάση τη τοπολογία του δέντρου και το μήκος των κλαδιών του. Συνεπώς στον υπολογισμό υποθέτουμε ως δεδομένα εισόδου τα παραπάνω στοιχεία και με αυτά αξιολογούμε το δέντρο. Αρχικά προσδιορίζουμε αυθαίρετα ένα κόμβο που δε βρίσκεται στα φύλλα του δέντρου και αποτελεί τη ρίζα του δέντρου.Στη συνέχεια εκτελούμε post-order διάσχιση του δέντρου ξεκινώντας από τη ρίζα του. Με το τρόπο αυτό τα διανύσματα πιθανότητας των κόμβων υπολογίζονται από κάτω προς τα πάνω, δηλαδή από τα φύλλα προς τη ρίζα. Με την ολοκλήρωση των παραπάνω υπολογισμών, βρίσκουμε τη συνολική πιθανοφάνεια για κάθε θέση στην αλληλουχία στο κόμβο της ρίζας με βάση τη σχέση:

$$L(i) = \sum_{S_k=A}^{T} p_{S_k} L_{S_k}^{(k)}(i) \tag{2.3}$$

Η συνολική πιθανοφάνεια του δέντρου υπολογίζεται από τη σχέση:

$$L = \prod_i L(i) \tag{2.4}$$

Αυτό που μένει να προσδιορίσουμε είναι το μοντέλο με το οποίο υπολογίζουμε τις πιθανότητες με βάση τις οποίες μια νουκλεοτιδική βάση i μεταλλάσσεται σε κάποια j.



Σχήμα 2.2: Phylogenetic Likelihood Function Computation

2.2.4 Στατιστικά Μοντέλα Εξέλιξης

Ο υπολογισμός των πιθανοτήτων νουκλεοτιδικής υποκατάστασης απαιτεί να ορίσουμε το μοντέλο που υπολογίζει τη πιθανότητα μια νουκλεοτιδική βάση i να μεταλλαχθεί σε οποιαδήποτε άλλη j μέσα σε χρονικό διάστημα t που ορίζει ένα τυχαίο κλαδί. Παρατηρούμε ότι η τυχαία νουκλεοτιδική βάση j μπορεί να ταυτίζεται με την i. Ο πίνακας με τις πιθανότητες υποκατάστασης είναι ένας 4x4 πίνακας οποίος είναι συνάρτηση του χρόνου και είναι της μορφής:

$$\mathrm{P(t)} = \begin{bmatrix} P_{AA}(t) & P_{AG}(t) & P_{AC}(t) & P_{AT}(t) \\ P_{GA}(t) & P_{GG}(t) & P_{GC}(t) & P_{GT}(t) \\ P_{CA}(t) & P_{CG}(t) & P_{CC}(t) & P_{CT}(t) \\ P_{TA}(t) & P_{TG}(t) & P_{TC}(t) & P_{TT}(t) \end{bmatrix}$$

Θεωρούμε ότι για την νουκλεοτιδική υποκατάσταση αναφερόμαστε σε χρονικά ομογενείς αλυσίδες Markov και επομένως ο πίνακας υπολογίζεται από τη σχέση:

$$P(t) = e^{Qt}$$

Το Q στη παραπάνω σχέση αντιστοιχίζεται στον πίνακα του οποίου τα στοιχεία είναι οι ρυθμοί υποκατάστασης των νουκλεοτιδικών βάσεων.Ο πίνακας Q παραμένει σταθερός σε σχέση με το χρόνο. Αρχικά λοιπόν χρειάζεται να υπολογίσουμε τα στοιχεία του πίνακα Q. Κάθε στοιχείο Q_{ij} όπου $i \neq j$ ισούται με το ρυθμό με τον οποίο κάποια βάση i υποκαθίσταται από μια διαφορετική βάση j. Τα στοιχεία της διαγωνίου, δηλαδή για $i = j$, παίρνουν τιμές τέτοιες ώστε το συνολικό άθροισμα στη κάθε γραμμή του πίνακα να είναι ίση με το 0.

$$Q_{ii} = -\sum_{j|j\neq i} Q_{ij}$$

Θεωρούμε επιπλέον τις παραμέτρους $\pi_A, \pi_C, \pi_G, \pi_T$ οι οποίες είναι οι συχνότητες των βάσεων A,C,G,T αντίστοιχα και τον παράγοντα μ που ισούται με τη μέση στιγμιαία τιμή των ρυθμών υποκατάστασης.

Το πρώτο και ιδιαίτερα απλοϊκό μοντέλο νουκλεοτιδικής υποκατάστασης προτάθηκε το 1969 από τους Jukes και Cantor (JC) [26] . Το μοντέλο αυτό υποθέτει τον ίδιο ρυθμό αλλαγών μεταξύ όλων των νουκλεοτιδικών βάσεων και ίσες συχνότητες βάσεων $\pi_A = \pi_T = \pi_C = \pi_G = 1/4$. Ο Felsestein επέκτεινε το JC μοντέλο ώστε να περιέχει διαφορετική συχνότητα για κάθε βάση(F81) [14].Ωστόσο οι μεταλλάξεις μεταξύ βάσεων που έχουν όμοια χημική δομή (A,G:πουρίνες, C,T:πυριμιδίνες) που ονομάζονται μεταπτώσεις έχουν μεγαλύτερη συχνότητα από τις μεταστροφές(μετάλλαξη από πουρίνη σε πυριδιμίνη ή αντίστροφα) επειδή η αντικατάσταση παρόμοιων χημικών δομών είναι πιθανότερη από τη σκοπιά της μοριακής ενέργειας. Στηριζόμενος στο γεγονός ότι για μεγαλύτερο αριθμό μεταπτώσεων σε σχέση με τις μεταστροφές δεν αλλάζει η κωδικοποίηση του DNA σε μια συγκεκριμένη πρωτεΐνη με αντικατάσταση αμινοξέων ο Kimura πρότεινε ένα μοντέλο υποκατάστασης με 2 παραμέτρους(K2P) [29] στο οποίο οι ρυθμοί των αλλαγών διαφέρουν μεταξύ μεταπτώσεων και μεταστροφών ενώ διατηρούνται ίσες οι συχνότητες για τις βάσεις, $\pi_A = \pi_T = \pi_C = \pi_G = 0.25$. Η γενίκευση του μοντέλου K2P για διαφορετικές συχνότητες μεταξύ των βάσεων οδηγεί στο μοντέλο που πρότειναν οι Hasegawa,Kishino,Yano(HKY85) [17].Το μοντέλο HKY85 είναι ιδιαίτερα αποτελεσματικό στην εφαρμογή του καθώς παρουσιάζει ένα καλό συμβιβασμό μεταξύ ακρίβειας και υπολογιστικής ταχύτητας και για αυτό το λόγο χρησιμοποιείται από αρκετά σύγχρονα προγράμματα που κάνουν φυλογενετικές αναλύσεις όπως το RAxML [43] και το PhyML [16].

Θεωρώντας το γενικό χρονικά αντιστρεπτό μοντέλο (**GTR**) ο πίνακας υποκατάστασης Q είναι:

$$Q=\begin{bmatrix} \mu(a\pi_C+b\pi_G+c\pi_T) & \mu a\pi_C & \mu b\pi_G & \mu c\pi_T \\ \mu g\pi_A & \mu(g\pi_A+d\pi_G+e\pi_T) & \mu d\pi_G & \mu e\pi_T \\ \mu h\pi_A & \mu j\pi_C & \mu(h\pi_C+j\pi_C+f\pi_T) & \mu f\pi_T \\ \mu i\pi_A & \mu k\pi_C & \mu l\pi_G & \mu(i\pi_A+k\pi_C+l\pi_G) \end{bmatrix}$$

Το GTR μοντέλο παρουσιάστηκε για πρώτη φορά από τον Simon Tavaré [46] και αποτελεί το όσο το δυνατόν πιο γενικευμένο, ανεξάρτητο, χρονικά αντιστρεπτό μοντέλο.Οι τιμές $a, b, ..., l$ είναι παράμετροι που αντιστοιχούν σε όλους τους πιθανούς συνδυασμούς υποκατάστασης μιας βάσης από μια άλλη.Η παράμετρος μ αντιστοιχεί στο συνολικό μέσο ρυθμό νουκλεοτιδικής υποκατάστασης.Ακόμα με το κ εκφράζεται η αναλογία μεταπτώσεων-μεταστροφών.

Εφαρμόζοντας στον παραπάνω πίνακα το **JC** μοντέλο έχουμε $\pi_A = \pi_T = \pi_C = \pi_G = 1/4$ και επειδή όλες οι υποκαταστάσεις θεωρούνται ισοδύναμες $a = b = ... = l = 1$.Συνεπώς ο πίνακας Q διαμορφώνεται ως εξής:

$$Q = \begin{bmatrix} \frac{3}{4}\mu & \frac{1}{4}\mu & \frac{1}{4}\mu & \frac{1}{4}\mu \\ \frac{1}{4}\mu & \frac{3}{4}\mu & \frac{1}{4}\mu & \frac{1}{4}\mu \\ \frac{1}{4}\mu & \frac{1}{4}\mu & \frac{3}{4}\mu & \frac{1}{4}\mu \\ \frac{1}{4}\mu & \frac{1}{4}\mu & \frac{1}{4}\mu & \frac{3}{4}\mu \end{bmatrix}.$$

Για το μοντέλο **F81**, το οποίο αποτελεί γενίκευση του JC για διαφορετικές δύο διαφορετικές συχνότητες βάσεων ανάλογα αν υπάρχει μετάπτωση ή μεταστροφή θεωρούμε για τις συχνότητες υποκατάστασης $\pi_R = \pi_A + \pi_G$ και $\pi_Y = \pi_C + \pi_T$ και για το πίνακα υποκατάστασης έχουμε:

$$Q = \begin{bmatrix} -\mu(\pi_G+\pi_Y) & \mu\pi_C & \mu\pi_G & \mu\pi_T \\ \mu\pi_A & -\mu(\pi_T+\pi_R) & \mu\pi_G & \mu\pi_T \\ \mu\pi_A & \mu\pi_C & -\mu(\pi_A+\pi_Y) & \mu\pi_T \\ \mu\pi_A & \mu\pi_C & \mu\pi_G & -\mu(\pi_C+\pi_R) \end{bmatrix}.$$

Στο μοντέλο **K2P** έχουμε 2 ρυθμούς υποκατάστασης, έναν για τις μεταπτώσεις και έναν για τις μεταστροφές.Θεωρούμε ότι $a = b = c = f = 1$ και $b = e = k$. Αν

ο ρυθμός μεταπτώσεων είναι $\mu\kappa/4$ και μεταστροφών $\mu/4$ ο πίνακας υποκατάστασης γίνεται:

$$\mathrm{Q} = \begin{bmatrix} -\frac{1}{4}\mu(\kappa+2) & \frac{1}{4}\mu & \frac{1}{4}1/4\mu\kappa & \frac{1}{4}\mu \\ \frac{1}{4}\mu & -\frac{1}{4}\mu(\kappa+2) & \frac{1}{4}\mu & \frac{1}{4}\mu\kappa \\ \frac{1}{4}\mu\kappa & \frac{1}{4}\mu & -\frac{1}{4}\mu(\kappa+2) & \frac{1}{4}\mu \\ \frac{1}{4}\mu & \frac{1}{4}\mu\kappa & \frac{1}{4}\mu & -\frac{1}{4}\mu(\kappa+2) \end{bmatrix}$$

.

Γενικεύοντας το μοντέλο K2P για διαφορετικές συχνότητες για κάθε βάση έχουμε το μοντέλο **HKY85** και το πίνακα Q να διαμορφώνεται ως εξής:

$$\mathrm{Q} = \begin{bmatrix} -\mu(\kappa\pi_G + \pi_Y) & \mu\pi_C & \mu\kappa\pi_G & \mu\pi_T \\ \mu\pi_A & -\mu(\kappa\pi_T + \pi_R) & \mu\pi_G & \mu\kappa\pi_T \\ \mu\kappa\pi_A & \mu\pi_C & -\mu(\kappa\pi_A + \pi_Y) & \mu\pi_T \\ \mu\pi_A & \mu\kappa\pi_C & \mu\pi_G & -\mu(\kappa\pi_C + \pi_R) \end{bmatrix}.$$

Έχοντας ορίσει τον υπολογισμό για το πίνακα με τους ρυθμούς υποκατάστασης μπορούμε να υπολογίσουμε τους πίνακες πιθανοτήτων. Η σχέση $P(t) = e^{Qt}$ μπορεί να υπολογιστεί με τη βοήθεια των ιδιοτιμών και των ιδιοδιανυσμάτων του πίνακα Q.Για τα μοντέλα υποκατάστασης που έχουμε αναλύσει προκύπτουν οι παρακάτω σχέσεις.

Για το μοντέλο K2P έχουμε τις πιθανότητες μετάπτωσης,μεταστροφής και υποκατάστασης μιας βάσης με την ίδια βάση.

$$P_{ij}^{(K2P)}(t) = \begin{cases} \frac{1}{4} + \frac{1}{4}e^{-\mu t} + \frac{1}{2}e^{-\mu t\frac{k+1}{2}} & i = j \\ \frac{1}{4} + \frac{1}{4}e^{-\mu t} - \frac{1}{2}e^{-\mu t\frac{k+1}{2}} & i \neq j \\ \frac{1}{4} - \frac{1}{4}e^{-\mu t} & i \neq j \end{cases} \tag{2.5}$$

Στη πρώτη περίπτωση όπου $i \neq j$ έχουμε μετάπτωση και στη άλλη μεταστροφή.

Οι σχέσεις που υπολογίζουν τα στοιχεία για τους πίνακες πιθανοτήτων για τα μο-

ντέλα που έχουμε περιγράψει δίνονται παρακάτω.

$$P_{ij}^{(JC)}(t) = \begin{cases} \frac{1}{4} + \frac{3}{4}e^{-\mu t} & i = j \\ \frac{1}{4} - \frac{1}{4}e^{-\mu t} & i \neq j \end{cases} \tag{2.6}$$

$$P_{ij}^{(F81)}(t) = \begin{cases} \pi_j + (1 - \pi_j)e^{-\mu t} & i = j \\ \pi_j(1 - e^{-\mu t}) & i \neq j \end{cases} \tag{2.7}$$

$$P_{ij}^{(HKY85)}(t) = \begin{cases} \pi_j + \pi_j(\frac{1}{\pi_{sum(j)}} - 1)e^{-\mu t} + (\frac{\pi_{sum(j)} - \pi_j}{\pi_{sum(j)}})e^{-\mu t A} & i = j \\ \pi_j + \pi_j(\frac{1}{\pi_{sum(j)}} - 1)e^{-\mu t} - (\frac{\pi_{sum(j)} - \pi_j}{\pi_{sum(j)}})e^{-\mu t A} & i \neq j \\ \pi_j(\frac{1}{4}e^{-\mu t} & i \neq j \end{cases} \tag{2.8}$$

Για το μοντέλο HKY85 η παράμετρος $\pi_{sum(j)}$ είναι ίση με $\pi_A + \pi_G$ αν η νουκλεοτιδική βάση j είναι πουρίνη(A,G) ή με $\pi_C + \pi_T$ αν είναι πυριδιμίνη(C,T). Η παράμετρος A είναι ίση με $A = 1 + \pi_{sum(j)}(k - 1)$.

Είναι γενικώς αποδεκτό ότι οι στατιστικές μέθοδοι όπως το κριτήριο μέγιστης πιθανοφάνειας(Felsenstein 1981) [14] παράγουν περισσότερο αξιόπιστα αποτελέσματα από τις μεθόδους των ελάχιστων μεταβολών ή των ελάχιστων αποστάσεων(Yang and Rannala 2012; Whelan and Morrison 2017) [54], [50]. Ωστόσο οι φυλογενετικές αναλύσεις που βασίζονται στο κριτήριο μέγιστης πιθανοφάνειας απαιτούν περισσότερους υπολογιστικούς πόρους καθώς απαιτούν τη χρήση ευριστικών αλγορίθμων αναζήτησης μέσα στο τεράστιο χώρο των πιθανών φυλογενετικών δέντρων που μπορούν να σχηματιστούν(Chor and Tuller 2005) [10].

2.2.5 Μοντέλο ετερογένειας **sites** Γ

Η διαφοροποίηση των ρυθμών υποκατάστασης μεταξύ των νουκλεοτιδικών sites έχει αναγνωριστεί ως ένα χαρακτηριστικό της εξέλιξης των DNA ακολουθιών [33].Η παράλειψη των διαφορετικών ρυθμών υποκατάστασης μπορεί να έχει ως αποτέλεσμα λανθασμένη κατασκευή της φυλογενετικής ανάλυσης [53].

Για να συμπεριληφθεί το φαινόμενο αυτό στις φυλογενετικές αναλύσεις έχουν προταθεί διάφορα μοντέλα για τους ρυθμούς ετερογένειας μεταξύ των sites.Στη παρούσα εργασία μελετάμε το διακριτό μοντέλο ετερογένειας Γ το οποίο χρησιμοποιείται στις φυλογενετικές αναλύσεις του RAxML [43].

Το διακριτό μοντέλο Γ αποτελεί προσέγγιση του συνεχούς μοντέλου.Η συνάρτηση πυκνότητας πιθανότητας για τη συνεχή κατανομή Γ δίνεται από τη παρακάτω σχέση:

$$g(r;\alpha,\beta) = \frac{\beta^{\alpha} r^{\alpha-1} e^{-\beta r}}{\Gamma(\alpha)} \tag{2.9}$$

Ο μέσος της κατανομής Γ δίνεται από τη σχέση $E = \frac{a}{b}$.Το διακριτό μοντέλο Γ προσεγγίζει τη συνεχή κατανομή χρησιμοποιώντας K κατηγορίες με ίση πιθανότητα.Οι ρυθμοί υποκατάστασης παίρνουν τιμές στο διάστημα $(0,\infty)$.Το διάστημα αυτό χωρίζεται σε K τμήματα στα οποία αντιστοιχίζονται οι ρυθμοί υποκατάστασης με ίση πιθανότητα.Επιπλέον κάθε τέτοιο τμήμα αντιστοιχίζεται σε ένα διάστημα της κατανομής Γ.Ο ρυθμός υποκατάστασης σε κάθε κατηγορία παίρνει τιμή ίση με το μέσο ή το διάμεσο του διαστήματος της κατανομής Γ στο οποίο αντιστοιχίζεται.Οι διαφορετικές κατηγορίες συμπεριλαμβάνονται ώστε να υπάρχει ακρίβεια στις φυλογενετικές αναλύσεις, ωστόσο έχουν σημαντικό αντίκτυπο στην απόδοση των εργαλείων λογισμικού που πραγματοποιούν τις αναλύσεις.Αυτό γίνεται εύκολα κατανοητό αν πάρουμε υπόψιν ότι για κάθε πιθανή κατάσταση, σε κάθε site κάθε προγονικού κόμβου υπολογίζονται K τιμές.Συνεπώς οι υπολογισμοί πιθανοφάνειας και οι απαιτήσεις μνήμης αυξάνονται κατά ένα παράγοντα K.Ο πιο κοινός αριθμός κατηγοριών είναι $K = 4$ [52].

2.3 Υλοποίηση φυλογενετικής ανάλυσης στο **RAxML**

Αναλυτική περιγραφή του υπολογισμού της φυλογενετικής ανάλυσης στο πρόγραμμα RAxML μπορεί να βρεθεί στην εργασία του Kozlov (Models, Optimizations, and Tools for Large-Scale Phylogenetic Inference, Handling Sequence Uncertainty, and Taxonomic Validation) [30].

Ο αλγόριθμος του Felsenstein όπως παρουσιάστηκε σε προηγούμενη ενότητα αποτελεί ένα ακριβή και πρακτικό τρόπο για τον υπολογισμό της PLF.Η ανάλυση του RAxML εφαρμόζει τον αλγόριθμο του Felsenstein.Χρησιμοποιεί φυλογενετικά δέντρα χωρίς ρίζα.Αρχικά επιλέγει με τυχαίο τρόπο τη ρίζα του δέντρου.Στη συνέχεια πραγματοποιεί post-order διάσχιση του φυλογενετικού δέντρου από τα φύλλα προς ρίζα και αναδρομικά υπολογίζει τα διανύσματα πιθανότητας ή όπως ονομάζονται Conditional Likelihood Vectors(CLV's) σε κάθε εσωτερικό κόμβο.Τα διανύσματα στους κόμβους που είναι πρόγονοι του κόμβου της ρίζας χρησιμοποιούνται για τον υπολογισμό της τελικής πιθανοφάνειας του δέντρου.

2.3.1 Υπολογισμός πίνακα μετάβασης **P** σε κλαδί φυλογενετικού δέντρου

Όπως έχει περιγραφεί στην ενότητα 2.2.5 ο πίνακας $P(t)$ ορίζει τις πιθανότητες μετάβασης για οποιοδήποτε ζεύγος υποκατάστασης μετά από κάποιο συγκεκριμένο χρονικό διάστημα t.Η σχέση

$$P(t) = e^{Qt}$$

υπολογίζει οποιαδήποτε τιμή του πίνακα P.Στη γενική περίπτωση του μοντέλου εξέλιξης GTR για να υπολογιστούν οι τιμές του πίνακα P χρειάζεται να αποσυνθέσουμε το πίνακα Q με τη βοήθεια των ιδιοδιανυσμάτων και των ιδιοτιμών του.Δηλαδή χρειάζεται να υπολογιστεί ο αντιστρεπτός πίνακας U και ο διαγώνιος πίνακας Λ όπου $\Lambda = diag(\lambda_i)$ τέτοιοι ώστε

$$Q = U\Lambda U^{-1}$$

.

Οι τιμές λ_i είναι οι ιδιοτιμές του πίνακα Q, ενώ ο πίνακας U αποτελείται από τα αντίστοιχα ιδιοδιανύσματα του Q.Ως εκ τούτου οι τιμές του πίνακα P υπολογίζονται από τη σχέση:

$$P_{i,j}(t) = \sum_{k=1}^{4} e^{\lambda_k t}\, U_{i,k}\, U_{i,k}^{-1} \tag{2.10}$$

Η παράμετρος t εκφράζει χρονικό διάστημα και είναι ανάλογη του μήκους του κλαδιού που ενώνει ένα κόμβο με τον προγονικό του κόμβο.Επομένως ο πίνακας P είναι ξεχωριστός για κάθε κλαδί του δέντρου και υπολογίζεται στη αρχή της φυλογενετικής ανάλυσης και κάθε φορά που τροποποιείται το μήκος του αντίστοιχου κλαδιού κατά τη διάρκεια της αναζήτησης του φυλογενετικού δέντρου που μεγιστοποιεί τη πιθανοφάνεια του.

2.3.2 Υπολογισμός Διανυσμάτων Υπο Συνθήκη Πιθανότητας

Κάθε $CLV(ConditionalLikelihoodVectors)$ περιέχει 4 τιμές σε κάθε site,που αντιστοιχίζονται στις 4 πιθανές καταστάσεις που μπορεί να βρίσκεται, μια για κάθε νουκλεοτιδική βάση.Το RAxML χρησιμοποιεί το μοντέλο ετερογένειας Γ με 4 ρυθμούς υποκατάστασης.Συνεπώς, με βάση αυτά που έχουν αναφερθεί για το μοντέλο στην ενότητα 2.2.6 το μέγεθος κάθε διανύσματος αυξάνεται στα 16 στοιχεία ανά site.

Τα διανύσματα που βρίσκονται στους κόμβους στα φύλλα του δέντρου αρχικοποιούνται με τιμές πιθανοφάνειας μετά τη παρατήρηση της κατάστασης του site.Επομένως οι τιμές που αναθέτονται είναι 1.0 ή 0.0 ανάλογα με τη νουκλεοτιδική βάση που έχει παρατηρηθεί.Για παράδειγμα αν παρατηρηθεί η νουκλεοτιδική βάση T σε κάποιο site θέτουμε L(A) = 0.0,L(C) = 0.0,L(G) = 0.0,L(T) = 1.0.

Το CLV για κάποιο εσωτερικό κόμβο v μπορεί να υπολογιστεί από τα προγονικά διανύσματα των δύο απογόνων p και q.Η σχέση που υπολογίζει τα στοιχεία του διανύσματος v είναι:

$$CLV_{s,c,i}(t) = (\sum_{j=1}^{4} P_{i,j}(r_c b_{pv})\ CLV_{s,c,j}^{(p)})\ (\sum_{k=1}^{4} P_{i,k}(r_c b_{qv})\ CLV_{s,c,k}^{(q)}) \tag{2.11}$$

όπου s = 1... m είναι κάποιο site της αλληλουχίας, c = 1...4 είναι οι 4 κατηγορίες των ρυθμών υποκατάστασης, i = 1...4 είναι οι πιθανές καταστάσεις (A,C,G,T),r_c είναι ο εξελικτικός ρυθμός που αντιστοιχίζεται στη κατηγορία c,b_{pv} και b_{qv} είναι τα μήκη των κλαδιών μεταξύ των κόμβων (p,v) και (r,v) αντίστοιχα, $P(r_c b_{pv})$ και $P(r_c b_{qv})$ είναι οι πίνακες P για τα αντίστοιχα μήκη των κλαδιών b_{pv} και b_{qv}.

Το τελικό διάνυσμα CLV προκύπτει αφού πολλαπλασιαστεί με τα στοιχεία του ανεστραμένου πίνακα U των ιδιοδιανυσμάτων.Αυτοί οι υπολογισμοί επιτρέπουν την απλοποίηση των υπολογισμών κατά τη διάρκεια της βελτιστοποίησης στα μήκη των κλαδιών που ενώνουν του κόμβους και των υπολογισμών στη ρίζα του δέντρου. Ο ανεστραμένος πίνακας των ιδιοδιανυσμάτων αναφέρεται στο λογισμικό του RAxML ως EV (Eigenvectors), ενώ οι πρωταρχικές τιμές του CLV ως x1px2.Επομένως το τελικό διάνυσμα προκύπτει από τη σχέση:

$$CLV_{s,c,k}(t) = (\sum_{i=1}^{4} U_{k,i}^{-1}\ CLV_{s,c,i}^{(v)}) \tag{2.12}$$

2.4 Επιλογή Αλγορίθμου **RAxML** για υλοποίηση σε **FPGA**

Από το RAxML απομονώνουμε τη συνάρτηση που υπολογίζει το διάνυσμα πιθανότητας για κάποιο εσωτερικό κόμβο με είσοδο τα διανύσματα πιθανότητας των κόμβων-απογόνων του. Η συνάρτηση πραγματοποιεί υπολογισμούς με βάση τη θεωρητική συνάρτηση για τη γενική περίπτωση υπολογισμού της πιθανοφάνειας για κάποιο εσωτερικό κόμβο με δύο απογόνους οι οποίοι είναι εξίσου εσωτερικοί κόμβοι. Ακόμα εφαρμόζει επιπλέον πράξεις, από τις οποίες κάποιες αφορούν το scaling στις τιμές του διανύσματος εξόδου και κάποιες είναι συγκεκριμένες αποκλειστικά για το RAxML.Η επιλογή της συγκεκριμένης συνάρτησης ως αναφορά για τη κατασκευή των επιταχυντών που θα βρίσκονται στο σύστημα μνήμης έγινε με το κριτήριο του υπολογιστικού φόρτου τόσο στο χρόνο εκτέλεσης όσο και στις απαιτήσεις μνήμης τον οποίο καταλαμβάνει η συγκεκριμένη συνάρτηση.Στο RAxML καθώς και σε όλα τα γνωστά προγράμματα φυλογενετικών αναλύσεων που χρησιμοποιούν τη μέθοδο της μέγιστης πιθανοφάνειας η συνάρτηση PLF κυριαρχεί τόσο στο χρόνο εκτέλεσης όσο και στις απαιτήσεις μνήμης για την εκτέλεση της κατά 85%-95% [24].

Παρακάτω παρουσιάζεται ο αλγόριθμος του RAxML που υπολογίζει τη συνάρτηση φυλογενετικής πιθανοφάνειας με είσοδο τα διανύσματα πιθανότητας δυο εσωτερικών κόμβων και έξοδο το διάνυσμα πιθανότητας του κοινού τους προγόνου. Ο αλγόριθμος του RAxML που χρησιμοποιούμε ως αλγόριθμο αναφοράς είναι ο εξής:

Algorithm 1 Phylogenetic Likelihood Function

procedure PLF
  $inputs \leftarrow x1, x2, left, right, EV$
  $output \leftarrow x3$
  $scale \leftarrow 1$
  for $i < n$ **do**
    for $j < 4$ **do**
      for $k < 4$ **do**
        $umpx1 \leftarrow 0.0$
        $umpx2 \leftarrow 0.0$
        for $l < 4$ **do**
          $umpx1 \leftarrow umpx1 + x1[j * 4 + l] * left[j * 16 + k * 4 + l]$
          $umpx2 \leftarrow umpx2 + x2[j * 4 + l] * right[j * 16 + k * 4 + l]$
        end for
        $x1px2[k] \leftarrow umpx1 * umpx2$
      end for
      for $k < 4$ **do**
        for $l < 4$ **do**
          $x3[j * 4 + l] \leftarrow x1px2[k] * EV[4 * k + l]$
        end for
      end for
      for $scale$ && $l < 16$ **do**
        $scale \leftarrow (ABS(x3[l] < minlikelihood)$
      end for
      if $scale$ **then**
        for $l < 4$ **do**
          $x3[j * 4 + l] \leftarrow x3[j * 4 + l] * twoto256$
        end for
      end if
    end for
  end for

2.5 Απαιτήσεις Μνήμης Φυλογενετικής Συνάρτησης Πιθανοφάνειας

Όλες οι μοριακές ακολουθίες που δίνουμε ως είσοδο στη φυλογενετική ανάλυση έχουν το ίδιο μήκος s. Οι ακολουθίες αυτές βρίσκονται στα φύλλα του δέντρου και είναι γνωστές ακολουθίες, το οποίο σημαίνει ότι για μια συγκεκριμένη θέση k της αλλη-

λουχίας, σε κάθε νουκλεοτίδιο ανατίθεται πιθανότητα ίση με 0 ή 1 ανάλογα με το αν έχει παρατηρηθεί ή όχι στη συγκεκριμένη θέση της αλληλουχίας.Κάθε θέση k λοιπόν αποθηκεύει πληροφορία ενός διανύσματος 4 θέσεων της μορφής $[A, C, G, T]$, όπου τα A,C,G,T παίρνουν τιμές 0 ή 1 με βάση την ακολουθία που έχει παρατηρηθεί. Οι απαιτήσεις μνήμης για την αποθήκευση αυτών των αλληλουχιών θεωρούνται αμελητέες σε σχέση με τους εσωτερικούς κόμβους.Για να αναπαραστήσουμε τη πληροφορία του διανύσματος σε κάποια θέση k δεν απαιτούνται περισσότερα από 4 *bits*.Κάθε *bit* αντιστοιχίζεται σε μια νουκλεοτιδική βάση και δείχνει αν παρατηρείται ή όχι στην αλυσίδα του DNA.Επομένως με μια ακέραια τιμή μεγέθους $32bits$ μπορούμε να αποθηκεύσουμε 8 νουκλεοτίδια. Αυτό γίνεται δυνατό γιατί ξέρουμε εκ των προτέρων ποιο νουκλεοτίδιο βρίσκεται σε κάθε θέση οπότε στο διάνυσμα πιθανοτήτων έχουμε 1 *bit* που δείχνει αν μια νουκλεοτιδική βάση βρίσκεται ή όχι στη συγκεκριμένη θέση. Αν υποθέσουμε ότι έχουμε ένα σύνολο δεδομένων με 1,481 διαφορετικά είδη και 20,000,000 θέσεις για τη κάθε αλυσίδα DNA παρατηρούμε ότι απαιτούνται μόλις 13GB.

Για κάθε θέση στην αλληλουχία συνολικού μήκους s των εσωτερικών κόμβων η μνήμη αποθηκεύει ένα διάνυσμα πιθανοτήτων το οποίο περιλαμβάνει τις πιθανότητες στη συγκεκριμένη θέση της αλληλουχίας να παρατηρήσουμε κάποιο από τα 4 πιθανά νουκλεοτίδια A,C,T,G.Χρησιμοποιώντας αριθμούς διπλής ακρίβειας για κάθε πιθανότητα υπολογίζεται ότι για τους $n-2$ εσωτερικούς κόμβους απαιτούνται $(n-2) \cdot 8 \cdot 4 \cdot s$ bytes. Αντικαθιστώντας στη σχέση αυτή τις τιμές που χρησιμοποιήσαμε προηγουμένως για το μήκος κάθε αλληλουχίας και το πλήθος των ειδών έχουμε ότι απαιτείται μνήμη μεγέθους περίπου $1TB$. Η ποσότητα αυτή τετραπλασιάζεται αν συνυπολογίσουμε ότι σε πολλές μελέτες αξιοποιείται το μοντέλο Γ για την ετερογένεια στους ρυθμούς υποκατάστασης το οποίο χρησιμοποιεί 4 διαφορετικούς πίνακες υποκατάστασης και υπολογίζει 16 πιθανότητες για κάθε θέση της αλληλουχίας.Τέτοια περίπτωση είναι η φυλογενετική ανάλυση που πραγματοποιεί το RAxML.Από τα παραπάνω συμπεραίνουμε ότι στις απαιτήσεις μνήμης κυριαρχεί η ποσότητα μνήμης που χρειαζόμαστε για την αποθήκευση των διανυσμάτων πιθανότητας για τους εσωτερικούς κόμβους.

Η αρχιτεκτονική που αναπτύσσουμε συμβιβάζει μια μικρή επιβάρυνση σε υπολογιστικό κόστος με αντάλλαγμα τη συμπίεση μεγάλης ποσότητας μνήμης. Η αρχιτεκτονική μας στηρίζεται στη παρατήρηση ότι κατά την αναδρομική διάσχιση του φυλογενετικού δέντρου μπορούμε να αποκτούμε τα ζητούμενα διανύσματα με κάποιους επιπλέον υπολογισμούς που στηρίζονται στις εξισώσεις που παρουσιάσαμε για τη πιθανοφάνεια ενός κόμβου. Όταν τα διανύσματα πιθανοτήτων στους κόμβους παιδιά έχουν χρησιμοποιηθεί για τον υπολογισμό του προγονικού κόμβου δε χρειάζεται να βρίσκονται για περισσότερο χρόνο στη μνήμη.Μπορούμε να εξοικονομήσουμε χώρο με αντικατάσταση των διανυσμάτων αυτών στη μνήμη με κάποια άλλα και συνεπώς στη μνήμη να μην βρίσκονται $n-2$ διανύσματα εσωτερικών κόμβων για τον υπολογισμό της πιθανοφάνειας του φυλογενετικού δέντρου αλλά $x < n-2$. Το σύνολο των x διανυσμάτων που αποθηκεύονται στη μνήμη μπορεί να μεταβάλλεται δυναμικά με στοιχεία από το μεγάλο σύνολο

των $n-2$ διανυσμάτων κατά τη διάρκεια των υπολογισμών.

Κεφάλαιο 3

Σχετική Ερευνητική Δουλειά

Η συνεισφορά της παρούσας μελέτης βρίσκεται από τη μια πλευρά στην εξοικονόμηση πόρων μνήμης με μερική αποθήκευση των διανυσμάτων πιθανότητας στους κόμβους του φυλογενετικού δέντρου και στην επιτάχυνση της εκτέλεσης των υπολογισμών της φυλογενετικής συνάρτησης πιθανοφάνειας ώστε να μειωθεί στο ελάχιστο ο χρόνος με τον οποίο επιβαρύνεται ο συνολικός χρόνος εκτέλεσης από τον υπολογισμό των επιπλέον κλήσεων της PLF.Το κεφάλαιο αυτό παρουσιάζει τη σχετική ερευνητική δουλεία πάνω και στα δύο αυτά σκέλη της εργασίας.

3.1 Εξοικονόμηση Πόρων Μνήμης

Δύο κοινές προσεγγίσεις για τη μείωση των απαιτήσεων μνήμης είναι οι **out-of-core** αλγόριθμοι και οι αλγόριθμοι που εφαρμόζουν ανταλλαγή ποσότητας μνήμης με κάποια σχετικά μικρή επιπλέον ποσότητα χρόνου για εκτέλεση υπολογισμών.Τα δύο είδη αλγορίθμων βρίσκουν εφαρμογή σε προγράμματα κατά τη διάρκεια της εκτέλεσης των οποίων τα δεδομένα προς επεξεργασία έχουν τέτοιο μέγεθος που είναι πρακτικά αδύνατο να αποθηκευτούν στο σύνολο τους στη κύρια μνήμη RAM του συστήματος που εκτελεί τους υπολογισμούς ή να πραγματοποιηθούν αναλύσεις μεγάλης έκτασης με τη χρήση αυτών των προγραμμάτων.Στη συνέχεια παρουσιάζονται σχετικές εργασίες που υλοποιούν αυτές τις προσεγγίσεις.

3.1.1 **Out-of-Core** Αλγόριθμοι

Οι Out-of-Core αλγόριθμοι ή αλγόριθμοι εξωτερικής μνήμης(External Memory algorithms) θεωρούν ότι όλα τα προς επεξεργασία δεδομένα αποθηκεύονται σε κάποια εξωτερική μονάδα μνήμης(π.χ σκληρό δίσκο) και υπάρχει συνεχής μεταφορά δεδομένων μεταξύ εξωτερικής και κύριας μνήμης.Συνεπώς το πρόβλημα της χρήσης μεγάλης ποσότητας δεδομένων αντιμετωπίζεται με τη χρήση ανάλογου μεγέθους εξωτερικής μνήμης.Η

επικοινωνία Εισοδου/Εξόδου(Ε/Ε) μεταξύ της γρήγορης κύριας μνήμης και της πιο αργής εξωτερικής μνήμης(όπως είναι οι σκληροί δίσκοι) μπορεί να αποτελέσει το κύριο σημείο περιορισμού της απόδοσης στο χρόνο εκτέλεσης.Για το λόγο αυτό αναπτύσσονται μέθοδοι, ειδικά σχεδιασμένοι για να ελαχιστοποιήσουν την επιβάρυνση από τη μεταφορά δεδομένων μεταξύ δίσκου και RAM με έλεγχο της αποθήκευσης και της μετακίνησης Ε/Ε.Καλή απόδοση στη μεταφορά δεδομένων στους αλγορίθμους εξωτερικής μνήμης επιτυγχάνεται με τη κατάλληλη τροποποίηση ώστε να υπάρχει ο μέγιστος δυνατόν βαθμός τοπικότητας στα δεδομένα.Αλγόριθμοι εξωτερικής μνήμης και αντίστοιχες κατάλληλες δομές δεδομένων έχουν εφαρμοστεί ήδη σε πλήθος εφαρμογών για ένα μεγάλο εύρος επιστημονικών υπολογισμών όπως ο πολλαπλασιασμός πινάκων,FFT υπολογισμοί,η υπολογιστική γεωμετρία κλπ [49].

RapidNJ

Εφαρμογή αλγόριθμου που αξιοποιεί μέθοδο τύπου out-of-core για τη χρήση εξωτερικής μνήμης σε υπολογισμό φυλογενετικών αναλύσεων εντοπίζεται για φυλογενετικές αναλύσεις με χρήση της μεθόδου Neighbor-Joining.Η μέθοδος neighbour-joining(NJ) αποτελεί μια ευρέως χρησιμοποιούμενη μέθοδο για φυλογενετικές αναλύσεις της τάξης των εκατοντάδων ή συχνά και χιλιάδων ειδών.Ωστόσο η πραγματοποίηση φυλογενετικών αναλύσεων για δεκάδες ή εκατοντάδες χιλιάδες είδη είναι υπολογιστικά αδύνατη.Ο αλγόριθμος RapidNJ που παρουσιάστηκε από τους Martin Simonsen, Thomas Mailund και Christian Pedersen [41] δίνει τη δυνατότητα μείωσης του εκτελέσιμου χρόνου.Αυτό επιτυγχάνεται με επιτάχυνση του χρόνου αναζήτησης του ζεύγους των κόμβων που γίνονται join,ενώ χρησιμοποιούνται τα ίδια κριτήρια βελτιστοποίησης με την αρχική έκδοση του αλγορίθμου NJ. Σε μεταγενέστερη εργασία τους(Building very large neighbour-joining trees) [42] επεκτείνουν το εργαλείο RapidNJ στο ERapidNJ.Το ERapidNJ διαθέτει 3 διαφορετικές μεθόδους για τη κατασκευή δέντρων και επιλέγει αυτόματα κάθε φορά μια από αυτές βασιζόμενο στις απαιτήσεις μνήμης του δοσμένου data set και των διαθέσιμων πόρων μνήμης του συστήματος.Για μικρά data sets χρησιμοποιείται η αρχική μέθοδος RapidNJ ενώ για μεσαίου μεγέθους χρησιμοποιείται μια μέθοδος που επεκτείνει την RapidNJ ώστε να μειωθεί σημαντικά η κατανάλωση μνήμης με μια αναλογικά πολύ μικρότερη μείωση στην απόδοση του χρόνου εκτέλεσης της αναλύσης.Τέλος για μεγάλου μεγέθους δεδομένα εισόδου χρησιμοποιείται η μέθοδος RapidDiskNJ,η οποία αποτελεί επίσης επέκταση της RapidNJ για αποδοτική αποθήκευση τόσο στην εσωτερική όσο και στην εξωτερική μνήμη με κατάλληλη χρήση δομών δεδομένων ώστε να αποφευχθεί η σελιδοποίηση από το λειτουργικό σύστημα και να μειωθεί η επιβάρυνση στο χρόνο εκτέλεσης από τις προσβάσεις στην εξωτερική μνήμη.Παράλληλα αναπτύσσεται ευριστική συνάρτηση που χρησιμοποιείται και στις 3 προαναφερθείς μεθόδους για το περιορισμό της επίδρασης στην απόδοση από περιττά

δεδομένα που βρίσκονται στη μνήμη και μείωση του χώρου αναζήτησης.

ML-based PLF

Η εργασία των Izquierdo-Carrasco,Stamatakis [25] υλοποιεί out-of-core μέθοδο για το χειρισμό της χρήσης εξωτερικής μνήμης σε φυλογενετική ανάλυση, η οποία πραγματοποιείται με τη χρήση φυλογενετικής συνάρτησης πιθανοφάνειας(PLF) που χρησιμοποιεί το κριτήριο της μέγιστης πιθανοφάνειας(ML-Maximum Likelihood.Η κατασκευή φυλογενετικών δέντρων με το κριτήριο της μέγιστης πιθανοφάνειας προσφέρει ιδιαίτερα μεγάλη ακρίβεια.Ωστόσο αποτελεί πρόκληση η αντιμετώπιση του αντίστοιχου κόστους αυτής της λύσης τόσο υπολογιστικά όσο και σε πόρους μνήμης.Με βάση τη μελέτη των Izquierdo-Carrasco,Stamatakis η έλλειψη πόρων μνήμης αποτελεί το κύριο περιοριστικό παράγοντα για μεγάλης κλίμακας φυλογενετικές αναλύσεις, ειδικά στις αναλύσεις που βασίζονται στο γονιδίωμα.

Αντίθετα με τους NJ αλγορίθμους οι απαιτήσεις μνήμης από μια PLF συνάρτηση κυριαρχείται από ένα σύνολο διανυσμάτων πιθανότητας.Η φυλογενετική ανάλυση με χρήση της PLF προσφέρεται για εφαρμογή out-of-core εκτέλεσης εξαιτίας του τρόπου πρόσβασης στα δεδομένα της μνήμης. Η πιθανοφάνεια σε κάθε δέντρο υπολογίζεται δεδομένης μιας ρίζας, η οποία επιλέγεται αυθαίρετα,με διεξαγωγή post-order διάσχισης των κόμβων του δέντρου.Το πλήθος των τιμών σε κάθε διάνυσμα πιθανότητας υπολογίζεται αναδρομικά συνδυάζοντας τις τιμές των διανυσμάτων σε αριστερό και δεξί κόμβο-απόγονο.Με αυτό το τρόπο, μια διάσχιση των κόμβων του δέντρου για τον υπολογισμό της πιθανοφάνειας του, προχωράει από τα φύλλα προς τη ρίζα.Η πρόσβαση στα διανύσματα πιθανότητας είναι γραμμική και ο τρόπος πρόσβασης δίνεται από τη τοπολογία του δέντρου.Το γεγονός αυτό δίνει τη δυνατότητα τα διανύσματα πιθανότητας να αποθηκεύονται γραμμικά στη μνήμη, το ένα μετά το άλλο, προσφέροντας μεγάλο βαθμό τοπικότητας.

Δεδομένης της τοπικότητας των διανυσμάτων πιθανότητας στη μνήμη που προσφέρει η μέθοδος πρόσβασης στα διανύσματα πιθανότητας σε συνδυασμό με τη χρησιμοποίηση κατάλληλων δομών και αλγορίθμων αντικατάστασης που εκμεταλλεύονται αυτή τη τοπικότητα δίνεται η δυνατότητα μείωση της χρησιμοποιούμενης RAM ακόμα και με περιορισμένη χρήση της διαθέσιμης RAM η τιμή του miss rate στη RAM παραμένει ιδιαίτερα χαμηλή.Η out-of-core υλοποίηση των Izquierdo-Carrasco, Stamatakis εκτελεί υπολογισμούς της PLF πολύ πιο αποτελεσματικά από την πρότυπη υλοποίηση στην οποία χρησιμοποιείται σελιδοποίηση από το λειτουργικό σύστημα.Ωστόσο η μείωση των απαιτήσεων μνήμης με μετακίνηση δεδομένων από/προς το δίσκο και τη μνήμη αυξάνει τουλάχιστον κατά μια τάξη μεγέθους το συνολικό χρόνο εκτέλεσης εξαιτίας των αργών πράξεων ανάγνωσης και εγγραφής από και προς το δίσκο.Το γεγονός αυτό παράγει μια μη πρακτική προσέγγιση ακόμα και όταν χρησιμοποιείται παράλληλο σύστημα αρχείων υψηλής απόδοσης.

3.1.2 Ανταλλαγή μνήμης με χρόνο εκτέλεσης(**time-memory trade-off**)

Ο όρος time-memory trade-off ή αλλιώς time-space trade-off αναφέρεται στη προσέγγιση η οποία με στόχο την εξοικονόμηση των χρησιμοποιούμενων πόρων μνήμης εφαρμόζει μέθοδο μείωσης της ποσότητας χρησιμοποιούμενης μνήμης με κάποια άλλη ποσότητα χρόνου που προστίθεται στον αρχικό χρόνο εκτέλεσης της εφαρμογής.Αυτή η στρατηγική έχει εφαρμοστεί σε διάφορα πεδία όπως αυτό της χρονοδρομολόγησης πακέτων [51].Ακόμα τεχνικές time-memory trade-off έχουν εφαρμοστεί σε αναγνώριση γλώσσας [11].Στο πλαίσιο των φυλογενετικών αναλύσεων έχει εφαρμοστεί αλγόριθμος time-memory trade-off για υπολογισμούς της συνάρτησης PLF από τους Izquierdo-Carrasco,Gagneur,Stamatakis [24] όπου χρησιμοποιούνται καινοτόμες μέθοδοι λογισμικού.

Σε αυτή την εργασία η ποσότητα μνήμης που δεσμεύεται για την φυλογενετική ανάλυση δεν αντιστοιχεί στο πλήθος των διανυσμάτων πιθανότητας των κόμβων-προγόνων.Το σύνολο των διανυσμάτων που αποθηκεύεται στη μνήμη αλλάζει δυναμικά κατά τη διάρκεια της φυλογενετικής ανάλυσης με βάση κάποια στρατηγική αντικατάστασης.Ταυτόχρονα δημιουργείται κατάλληλη δομή η οποία αποθηκεύει ποια διανύσματα βρίσκονται στη μνήμη και άλλα δεδομένα ώστε να παρακολουθεί την αντιστοίχιση των διανυσμάτων πιθανότητας στις διαθέσιμες θέσεις μνήμης και την εφαρμοζόμενη στρατηγική αντικατάστασης.Η δομή ενημερώνεται μετά από κάθε κλήση της PLF για τον υπολογισμό της πιθανοφάνειας, είτε μετά από εφαρμογή του αλγορίθμου SPR(Subtree Prunning and Regrafting) που εκτελεί μερική αναδιάταξη των κόμβων του φυλογενετικού δέντρου, είτε λόγω πλήρους αναδιάταξης του δέντρου, για να διατηρείται η συνέπεια στα δεδομένα.

Το επιπλέον υπολογιστικό κόστος που προκαλείται από το πλήθος των επανυπολογισμών των διανυσμάτων που βρίσκονται εκτός μνήμης είναι συγκριτικά μικρότερο όταν εφαρμόζεται κατάλληλη στρατηγική αντικατάστασης των διανυσμάτων που βρίσκονται στη μνήμη.Το γεγονός αυτό επιτρέπει τον υπολογισμό φυλογενετικών αναλύσεων με χρήση τη PLF πάνω σε μεγαλύτερα datasets, ειδικά όταν ο περιοριστικός παράγοντας για τη πραγματοποίηση της ανάλυσης είναι η διαθέσιμη RAM.Οι έννοιες που παρουσιάζονται στην εργασία αυτή μπορούν να εφαρμοστούν σε όλα τα προγράμματα που πραγματοποιούν αναλύσεις με χρήση της PLF όπως τα GARLI,PHYML, MrBayes.

3.2 Επιτάχυνση της Φυλογενετικής Συνάρτησης Πιθανοφάνειας

Η φυλογενετική συνάρτηση πιθανοφάνειας αποτελεί μια ευρέως χρησιμοποιούμενη μέθοδος για την εκτίμηση και την επιλογή μεταξύ διαφορετικών εξελικτικών σεναρίων που αναπαριστώνται γραφικά από φυλογενετικά δέντρα.Το πλεονέκτημα αυτής της μεθόδου είναι η ακρίβεια που προσφέρει στην ανάλυση, ενώ το βασικό της μειονέκτημα είναι

το υπολογιστικό κόστος.Στη συνέχεια παρουσιάζονται εργασίες που επιχειρούν να μειώσουν αυτό ακριβώς το υπολογιστικό κόστος.

3.2.1 Χρήση τεχνολογίας αναδιατασσόμενης λογικής(**FPGA**)

Μια σειρά από 4 εργασίες του The Exelixis Lab επιχειρούν να επιταχύνουν τη συνάρτηση PLF η οποία χρησιμοποιεί το κριτήριο της μέγιστης πιθανοφάνειας.Οι δύο πρώτες εργασίες, εφαρμόζουν δυο εναλλακτικές προσεγγίσεις για το παραλληλισμό των πράξεων της PLF σε αναδιατασσόμενη λογική.

Η πρώτη εργασία με τίτλο "Exploring FPGAs for Accelerating the Phylogenetic Likelihood Function" [3] παρουσιάζει σχεδίαση που μπορεί να διεξάγει πλήρης διασχίσεις χρησιμοποιώντας τον αλγόριθμο του Felsenstein πάνω σε ισορροπημένα δυαδικά δέντρα χρησιμοποιώντας το GTR μοντέλο ως μοντέλο νουκλεοτιδικής υποκατάστασης.Χρησιμοποιεί μονάδες υπολογισμού πιθανοφάνειας τοποθετημένες σε δενδρική διάταξη σε ένα ενιαίο deep pipeline.Υλοποιούνται τρείς φάσεις μέχρι να υπολογιστεί ο βαθμός πιθανοφάνειας του φυλογενετικού δέντρου.Στη πρώτη φάση υπολογίζονται τα διανύσματα που αντιστοιχίζονται στα φύλλα του δέντρου. Η δεύτερη φάση είναι η φάση υπολογισού των εσωτερικών κόβων του δέντρου ενώ η τρίτη είναι ο υπολογισμός του βαθού πιθανοφάνειας από τα διανύσατα πιθανοφανειών των θέσεων της ρίζας.Με βάση τα αποτελέσματα που δίνονται η συγκεκριμένη αρχιτεκτονική συγκρινόμενη με ένα μονοπύρηνο σύστημα εκτελεί γρήγορα τον υπολογισμό της PLF και αποδοτικά σε σχέση με τη χρήση της μνήμης ενώ παραμένει ανταγωνιστική όταν συγκρίνεται με ενα πολυπύρηνο σύστημα, 16 πυρήνων, όπου το λογισμικό χρησιμοποιεί παράλληλη υλοποίηση που βασίζεται στο OpenMP.Στην επόμενη εργασία ("A reconfigurable architecture for the phylogenetic likelihood function") [5] υπάρχει επέκταση της αρχικής αρχιτεκτονικής η οποία εκμεταλλεύται τις δυνατότητες παραλληλισμού που προσφέρει η PLF ώστε η PLF να υπολογίζεται για οποιαδήποτε τοπολογία φυλογενετικού δέντρου στην ίδια ταχύτητα που εφάρμοζε τους υπολογισμούς της η πρώτη σχεδίαση.Η σχεδίαση είναι εντελώς ανεξάρτητη από το μέγεθος και το σχήμα του φυλογενετικού δέντρου.Το γεγονός αυτό τη καθιστά καταλληλότερη για τη πραγματοποίηση αναλύσεων που στηρίζονται σε δεδομένα από πραγματικές υποθέσεις.

Η τρίτη εργασία ("A generic and versatile architecture for inference of evolutionary trees under maximum likelihood") [4] προτείνει αρχιτεκτονική η οποία είναι αποτελεσματική στον υπολογισμό της πιθανοφάνειας φυλογενετικών δέντρων για όλα τα πιθανά βιολογικά δεδομένα εισόδου(DNA,RNA,αλληλουχίες πρωτεϊνών).Αξιοσημείωτη βελτίωση σε σχέση με τις προηγούμενες εργασίες είναι ότι αυτή η έκδοση περιλαμβάνει στους υπολογισμούς το παράγοντα του ρυθμού ετερογένειας.Με τη βοήθεια των μοντέλων που περιγράφουν το ρυθμό ετερογένειας στο νέο υπολογισμό της PLF συμπεριλαμβάνεται το βιολογικό γεγονός ότι τα sites στις αλληλουχίες εισόδου εξελίσσονται στο χρόνο με διαφορετικές ταχύτητες.Επιπλέον αυτή η σχεδίαση παρέχει όλη τη λει-

τουργικότητα που μπορεί να αποδοθεί σε κάποιο συν-επεξεργαστή(co-processor) για την υποστήριξη ενός προγράμματος που χρησιμοποιεί το κριτήριο της μέγιστης πιθανοφάνειας για τη πραγματοποίηση αναλύσεων πάνω σε πραγματικές υποθέσεις.

Η τελευταία εργασία της σειράς είναι η "An Optimized Recongurable System for Computing the Phylogenetic Likelihood on DNA Data" [8].Η αρχιτεκτονική που παρουσιάζει η συγκεκριμένη εργασία συνδυάζει προσεγγίσεις που αναπτύχθηκαν στις προηγούμενες 2 εργασίες της σειράς.Παράλληλα προστίθενται βελτιώσεις που αφορούν τη χρήση πόρων καθώς και την απόδοση.Επιπλέον υλοποιείται διεπαφή τύπου FIFO για την επικοινωνία της μονάδας που υλοποιεί τη σχεδίαση με άλλες μονάδες όπως η εξωτερική μνήμη.Η αρχιτεκτονική είναι προσαρμοσμένη στις υπολογιστικές απαιτήσεις στρατηγικών αναζήτησης βέλτιστων φυλογενετικών δέντρων.

Οι προσομοιώσεις που δημιουργούνται για την αξιολόγηση της αρχιτεκτονικής χρησιμοποιούν εκτελέσεις του RAxML πάνω σε πραγματικές ακολουθίες DNA.Για την εκτίμηση της αρχιτεκτονικής που πραγματοποιεί τους υπολογισμούς της PLF, η απόδοση της συγκρίνεται με την απόδοση των υπολογισμών της PLF που πραγματοποιείται από το RAxML όταν εκτελείται πάνω σε CPU τύπου Intel i7 2600 με συχνότητα λειτουργίας τα 3.4 GHz.Ο συγκεκριμένος τύπος CPU προσφέρει τη δυνατότητα χρήσης AVX εντολών, πλάτους 256-bits.Συνεπώς για τη σύγκριση χρησιμοποιείται η έκδοση RAxML-Light-1.0.5 που αξιοποιεί τέτοιου τύπου εντολές.Επιπλέον ως μονάδα μέτρησης χρησιμοποιούνται τα VEUPS(Vector Entry Updates Per Second), η οποία δείχνει το πλήθος των διανυσμάτων που μπορούν να επεξεργαστούν στη μονάδα του χρόνου.Μετριέται ότι η απόδοση της αρχιτεκτονικής όταν εφαρμόζεται πάνω σε FPGA που με τους κατάλληλους πόρους υποστηρίζει 8 πυρήνες της αρχιτεκτονικής υπολογισμού της πιθανοφάνειας γίνεται έως 4 φορές καλύτερη σε σχέση με την δυνατότητα επεξεργασίας της συγκρινόμενης CPU.

Μια ακόμα εργασία που χρησιμοποιεί τεχνολογία αναδιατασσόμενης λογικής για την επιτάχυνση των υπολογισμών PLF είναι η "High Performance Phylogenetic Analysis With Maximum Parsimony on Recongurable Hardware" των Server Kasap και Khaled Benkrid [27].Η εργασία αυτή προτείνει σχεδίαση για εφαρμογή σε αναδιατασσόμενη λογική για την επιτάχυνση φυλογενετικών αναλύσεων που χρησιμοποιούν το κριτήριο της μέγιστης φειδωλότητας(Maximum Parsimony).Το κριτήριο της φειδωλότητας αναζητά τον αριθμό των αλλαγών που συμβαίνουν στους χαρακτήρες της αλληλουχίας και ο στόχος είναι να εξηγήσει τις εξελικτικές διάφορες με το μικρότερο δυνατό αριθμό αλλαγών.Η σχεδίαση αποτελείται από ένα γραμμικό συστολικό πίνακα(linear systolic array) που αποτελείται από 20 μονάδες επεξεργασίας κάθε μια από τις οποίες εκτελεί τον αλγόριθμο του Sankoff για κάποια διαφορετική τοπολογία του φυλογενετικού δέντρου.Ο αλγόριθμος του Sankoff είναι αλγόριθμος δυναμικού προγραμματισμού για τον υπολογισμό της βαθμολογίας φειδωλότητας(parsimony score) ενός δέντρου.Η αρχιτεκτονική εφαρμόζεται πάνω στους κόμβους ενός υπερυπολογιστή(supercomputer) από FPGA, τον Maxwell.Σε πειράματα με 12 taxa η απόδοση

της ανάλυση με τη χρήση της προτεινόμενης αρχιτεκτονικής σε ένα κόμβο του υπερυπολογιστή μπορεί να επιταχυνθεί κατά 4 φορές συγκρινόμενη με την απόδοση που προσφέρει κάποιο πρόγραμμα λογισμικού.Σύμφωνα με την εργασία ο αριθμός των taxa στην ανάλυση μπορεί εύκολα να κλιμακωθεί.

Η εργασία "FPGA acceleration of the phylogenetic likelihood function for Bayesian MCMC inference methods" [56] περιγράφει τη δημιουργία ενός co-processor με χρήση FPGA με αλγόριθμο αναφοράς για τη συνάρτηση PLF,τον αλγόριθμο που χρησιμοποιεί το πρόγραμμα MrBayes 3 για ανάλυση με στατιστική Bayes.Ο επιταχυντής χρησιμοποιεί αρχιτεκτονική deep pipelining για τους υπολογισμούς πιθανοφάνειας, με παράλληλη εκτέλεση πολλαπλών αριθμητικών πράξεων διπλής-ακρίβειας.Η αρχιτεκτονική του επιταχυντή αναπτύσσεται στη πλατφόρμα Annapolis Micro Systems WILD-STAR II Pro η οποία περιλαμβάνει μια Xirtex-2 Pro 100 FPGA.Η αρχιτεκτονική σε αυτή τη πλατφόρμα υλοποιείται για λειτουργία σε συχνότητα 165 MHz και καταναλώνει σχεδόν όλους τους διαθέσιμους πόρους σε λογικές μονάδες και πολλαπλασιαστές.Επιπλέον για την υλοποίηση της αρχιτεκτονικής χρησιμοποιείται η Virtex-6 SX 475 FPGA.Σε αυτή τη πλατφόρμα ο επιταχυντής λειτουργεί κοντά στη συχνότητα των 310MHz.Με βάση τις μετρήσεις που παρουσιάζει η εργασία για αναλύσεις με μεγάλα σύνολα δεδομένων ο επιταχυντής στη Virtex-6 SX 475 FPGA επιτυγχάνει εως και 10 φορές λιγότερο χρόνο εκτέλεσης σε σχέση με το λογισμικό όταν αυτό εκτελείται σε επεξεργαστή σειράς Intel Xeon 5500-series.

3.2.2 Χρήση τεχνολογίας **GPU**

Για την επιτάχυνση του υπολογισμού των φυλογενετικών αναλύσεων έχουν χρησιμοποιηθεί και άλλες τεχνολογίες πέραν των FPGA όπως οι GPU.Η παρούσα ενότητα παρουσιάζει τις εργασίες των Cheng Ling et al. [32] , Frederico Pratas et al. [38],Sandun Rajapaksa et al. [39].

Το πρόγραμμα MrBayes αποτελεί ένα δημοφιλές εργαλείο φυλογενετικών αναλύσεων με χρήση του κριτηρίου Bayes [21].Όπως συμβαίνει και με άλλα προγράμματα φυλογενετικών αναλύσεων το υπολογιστικό κόστος είναι ιδιαίτερα υψηλό και έχει σαν αποτέλεσμα ένα ανεπιθύμητα υψηλό χρόνο εκτέλεσης.Οι εργασίες των Cheng Ling et al. και των Frederico Pratas et al. στοχεύουν στη μείωση του εκτελέσιμου χρόνου του MrBayes με αρχιτεκτονικές για την επιτάχυνση των υπολογισμών της φυλογενετικής συνάρτησης πιθανοφάνειας του προγράμματος.

Η πρώτη έρευνα προτείνει τη μέθοδο $goMC^3$ για την επιτάχυνση των υπολογισμών του αλγορίθμου MC^3 που χρησιμοποιεί το MrBayes.Η προτεινόμενη μέθοδος υλοποιείται πάνω σε ξεχωριστές επεξεργαστικές μονάδες γραφικών(GPUs).Με σκοπό τη βελτίωση της απόδοσης η εργασία επωφελείται από τις δυνατότητες παραλληλισμού σε διαφορετικά επίπεδα(multi-granularity parallelism) που παρέχουν οι GPUs, την ε-

φαρμοζόμενη στρατηγική ρύθμισης των νημάτων που εκτελούνται παράλληλα στη GPU και τις βελτιστοποιήσεις στο επίπεδο κώδικα.Η προτεινόμενη μέθοδος $goMC^3$ εκμεταλλεύεται τη δυνατότητα παράλληλων υπολογισμών με πολλαπλά νήματα που προσφέρει η πλατφόρμα CUDA, το οποίο δίνει τη δυνατότητα μείωσης του κόστους μεταφοράς δεδομένων μεταξύ διαφορετικών υπολογιστικών μονάδων σε σχέση με άλλες μεθόδους.Η προτεινόμενη μέθοδος συγκρίνεται με άλλες διαθέσιμες μεθόδους που βασίζονται στη τεχνολογία των GPUs.Για την εκτίμηση και σύγκριση της απόδοσης όλες οι σχεδιάσεις εφαρμόζονται πάνω στον ίδιο αριθμό από CPUs και GPUs.Η σύγκριση πραγματοποιείται με κριτήρια το χρόνο εκτέλεσης, το speedup και τη δυνατότητα κλιμάκωσης της ανάλυσης για μεγαλύτερα datasets.Ο $goMC^3$ εφαρμοζόμενος σε κάρτα γραφικών τύπου Tesla C2075 επιτυγχάνει καλύτερη απόδοση σε χρόνο εκτέλεσης συγκρινόμενος με όλες τις υπόλοιπες μεθόδους που χρησιμοποιούν την τεχνολογία των GPUs ενώ ταυτόχρονα δίνει τη δυνατότητα μεγαλύτερης κλιμάκωσης της φυλογενετικής ανάλυσης για περισσότερους οργανισμούς και μεγαλύτερες αλληλουχίες σε σύγκριση με κάποιες από αυτές.

Η επόμενη έρευνα συγκρίνει τη δυνατότητα επιτάχυνσης της συνάρτησης PLF του προγράμματος MrBayes για 3 διαφορετικές τεχνολογίες όταν υλοποιούν παραλληλισμό υψηλής λεπτομέρειας.Οι τεχνολογίες που συγκρίνονται είναι γενικού σκοπού ομογενείς πολυπύρηνοι επεξεργαστές(διπύρηνοι και τετραπύρηνοι επεξεργαστές Intel και AMD), ετερογενείς πολυπήρηνες επεξεργαστικές μονάδες(IBM Cell/BE) και GPUs.Η εισαγωγή παραλληλισμού όταν χρησιμοποιείται η τεχνολογία ομογενών πολυπύρηνων επεξεργαστών γίνεται εύκολα με χρήση εργαλείων όπως οι οδηγίες μεταγλωτιστή του OpenMP API ή τα POSIX Threads.Ο IBM Cell/BE αποτελείται από 9 πυρήνες εκ των οποίων ένας πυρήνας είναι γενικού σκοπού(PowerPC Processor Element-PPE) και 8 ειδικού σκοπού(Synergistic Processing Elements-SPEs).Ο PPE είναι ένας απλός επεξεργαστής που σχεδιάζεται για την εκτέλεση του λειτουργικού συστήματος και το συντονισμό των υπολογισμών στους SPEs.Οι SPEs είναι επεξεργαστές απλούστερης σχεδίασης με σκοπό την εκτέλεση των παράλληλων εργασιών.Κάθε SPE περιλαμβάνει μια μικρή τοπική μνήμη ενώ συνολικά οι πυρήνες επικοινωνούν με ένα δίαυλο, τον Element Interconnect Bus (EIB).Ο παραλληλισμός στον IBM Cell/BE επιτυγχάνεται με τη χρήση pthreads, αλλά σε αντίθεση με τους πολυπύρηνους επεξεργαστές γενικού σκοπού,ο χρήστης είναι υπεύθυνος για τη μεταφορά των δεδομένων και τη κατάλληλη τοποθέτηση τους στις τοπικές μνήμες των SPEs.Στα πειράματα της εργασίας συγκρίνεται η απόδοση που επιτυγχάνεται από τις παραπάνω τεχνολογίες όταν χρησιμοποιούνται για παραλληλισμό.Τα αποτελέσματα δείχνουν ότι η απόδοση κάθε τεχνολογίας σχετίζεται με το κομμάτι του κώδικα που παραλληλοποιείται.Οι πυρήνες στους επεξεργαστές γενικού σκοπού μοιράζονται μνήμη cache, η οποία βρίσκεται εντός του chip,γεγονός που βοηθά την απόδοση όταν η αναλύση κλιμακώνεται για μεγαλύτερα σύνολα δεδομένων καθώς παρέχει γρήγορη μεταφορά δεδομένων και συγχρονισμό μεταξύ των πυ-

ρήνων.Οι GPUs επιτυγχάνουν καλύτερη απόδοση για το παράλληλο τμήμα του κώδικα αλλά η κοστοβόρα μεταφορά των δεδομένων έχει εμφανή συνέπεια στην απόδοση.Τέλος η απόδοση του IBM Cell/BE επηρεάζεται έντονα από τη μη αποδοτική εκτέλεση του σειριακού τμήματος του κώδικα στον PPE πυρήνα.

Η εργασία των Sandun Rajapaksa et al. παρουσιάζει μέθοδο επιτάχυνσης φυλογενετικών αναλύσεων που βασίζονται στο κριτήριο της μέγιστης πιθανοφάνειας με χρήση τεχνολογίας GPU και της πλατφόρμας CUDA αλλά και αξιοποίηση μονάδων CPU.Οι αναλύσεις που μελετώνται πραγματοποιούνται από το πρόγραμμα DNAml του πακέτου PHYLIP.Το DNAml είναι το μοναδικό εργαλείο που υποστηρίζει τον υπολογισμό των φυλογενετικών αναλύσεων με μέθοδο που βασίζεται στο κριτήριο της μέγιστης πιθανοφάνειας σε CUDA πλατφόρμα αξιοποιώντας τις δυνατότητες παραλληλισμού της GPU. Οι βελτιώσεις που πραγματοποιούνται σχετίζονται με την απλοποίηση του κώδικα,τη διευκόλυνση μελλοντικών επεκτάσεων με διαφορετικά παράλληλα πρωτόκολλα επικοινωνίας και τη βελτίωση της δυνατότητας κλιμάκωσης για μεγαλύτερα σύνολα δεδομένων.Ο παραλληλισμός με χρήση της τεχνολογίας GPU για το DNAml βοηθά τους βιοπληροφορείς να εργαστούν πάνω σε μεγαλύτερα σύνολα δεδομένων τα οποία προγενέστερα μπορούσαν εξαιτίας του μεγάλου χρόνου επεξεργασίας.Σε αυτή τη βελτιωμένη έκδοση επιτυγχάνεται σχεδόν ο μισός χρόνος επεξεργασίας για σύνολα δεδομένων της τάξης των 500 είδη οργανισμών.Τα αποτελέσματα της εργασίας δείχνουν ότι η ποσότητα του χρόνου που εξοικονομείται είναι υψηλότερη για μεγαλύτερο πλήθος από οργανισμούς.

Κεφάλαιο 4

Αρχιτεκτονική φυλογενετικής συνάρτησης πιθανοφάνειας

Σε προηγούμενο κεφάλαιο περιγράψαμε αναλυτικά τους υπολογισμούς που απαιτούνται για τη φυλογενετική συνάρτηση η οποία βασίζεται στο κριτήριο της μέγιστης πιθανοφάνειας.Στο κεφάλαιο αυτό παρουσιάζεται η σχεδίαση επιταχυντή για τον υπολογισμό μιας PLF κλήσης.Όπως θα δειχθεί στο 5^o κεφάλαιο, για να ελαττωθεί η ποσότητα μνήμης που δεσμεύεται κατά τη διάρκεια μιας φυλογενετικής ανάλυσης απαιτείται να υπολογιστούν μια ποσότητα από επιπρόσθετες PLF κλήσεις.Στόχος είναι η υλοποίηση των υπολογισμών κάθε PLF κλήσης από κατάλληλη μονάδα επιταχυντή ώστε να μειωθεί στο ελάχιστο ο χρόνος εκτέλεσης κάθε PLF κλήσης και συνολικά ο χρόνος για τον υπολογισμό των επιπρόσθετων κλήσεων.

Τα διανύσματα πιθανότητας των εσωτερικών κόμβων καταναλώνουν το μεγαλύτερο μέρος της μνήμης που χρησιμοποιείται για την αποθήκευση ενός φυλογενετικού δέντρου [23].Με βάση αυτό,επιλέγεται η σχεδίαση του επιταχυντή να μπορεί να υπολογίσει κάθε διάνυσμα πιθανότητας εσωτερικού κόμβου.Από τον αλγόριθμο του RAxML που υλοποιεί τους υπολογισμούς της PLF, απομονώνουμε το κομμάτι που υλοποιεί τον υπολογισμό ενός διανύσματος πιθανότητας εσωτερικού κόμβου στον οποίο οι απόγονοι του είναι εσωτερικοί κόμβοι του δέντρου.Ο επιταχυντής που παρουσιάζεται σε αυτό το κεφάλαιο έχει σαν αλγόριθμο αναφοράς αυτό το τμήμα του RAxML αλγορίθμου.

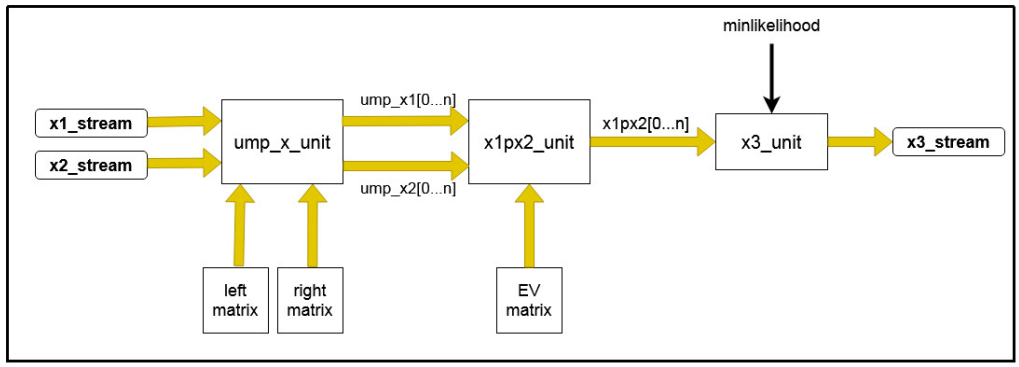
Η μονάδα υπολογισμού plf_accel_unit είναι εκείνη που πραγματοποιεί όλες τις απαραίτητες πράξεις για τον υπολογισμό ενός διανύσματος πιθανότητας μήκους N θέσεων.Στις παρακάτω ενότητες περιγράφουμε αναλυτικά τους υπολογισμούς που πραγματοποιούνται από τον αλγόριθμο αναφοράς και την αρχιτεκτονική της υλοποίησης τους σε αναδιατασσόμενη λογική.Επιπλέον περιγράφεται επέκταση αυτής της αρχιτεκτονικής για την υλοποίηση επιταχυντή που δίνει τη δυνατότητα υπολογισμού κλήσεων της PLF για τον κοινό πρόγονο περισσότερων από 2 κόμβους με μια κλήση προς τον επιταχυντή.

Ο αλγόριθμος που χρησιμοποιείται για την ανάπτυξη του επιταχυντή υλοποιεί τις σχέσεις 2.11 και 2.12.Τα δεδομένα εισόδου του επιταχυντή είναι τα δύο διανύσματα πιθανότητας των κόμβων-απογόνων,οι πίνακες πιθανοτήτων μετάβασης **left,right** και ο πίνακας EV.Στο τέλος των υπολογισμών ο αλγόριθμος του RAxML εφαρμόζει **scaling** στις τιμές του αποτελέσματος ώστε καμία από αυτές να μην βρίσκεται κάτω από ένα επιτρεπτό όριο.Το όριο αυτό δίνεται στην είσοδο της συνάρτησης μέσω της ακέραιας μεταβλητής **minlikelihood**.Στον αλγόριθμο αναφοράς έχουμε μια ακολουθία από εμφωλευμένα **loops**.Το εξωτερικό **loop**(LOOP1) πραγματοποιεί N επαναλήψεις, όσο είναι και το πλήθος των **sites** της αλληλουχίας του DNA.Το σώμα του **loop** υπολογίζει το διάνυσμα πιθανότητας μήκους 16 θέσεων που αντιστοιχίζεται σε κάθε **site**.Το LOOP2 αποτελεί τη βασική μονάδα υπολογισμού.

Η σχεδίαση έχει κύριο σκοπό να βελτιώσει το ρυθμό παραγωγής αποτελεσμάτων (**throughput**), εκμεταλλευόμενη τις δυνατότητες παραλληλισμού που προσφέρουν οι πράξεις του αλγορίθμου (πολλαπλασιασμοί και προσθέσεις).Ακόμα εκμεταλλεύεται τις δυνατότητες παραλληλισμού δεδομένων που υπάρχουν μέσα στον αλγόριθμο. Αρχικά πραγματοποιείται αποσύνθεση(**decomposition**) του αλγορίθμου σε υπολογιστικά απλούστερες εργασίες που μπορούν να εκτελεστούν παράλληλα πάνω σε διαφορετικά δεδομένα.Στη συνέχεια παρουσιάζεται η προτεινόμενη σχεδίαση για κάθε επιμέρους εργασίες.Τελικά οι επιμέρους αρχιτεκτονικές εντάσσονται σε μια ενιαία βασική διασωλήνωση (**pipeline**).

4.1 Σχεδίαση της μονάδας **plf_accel_unit**

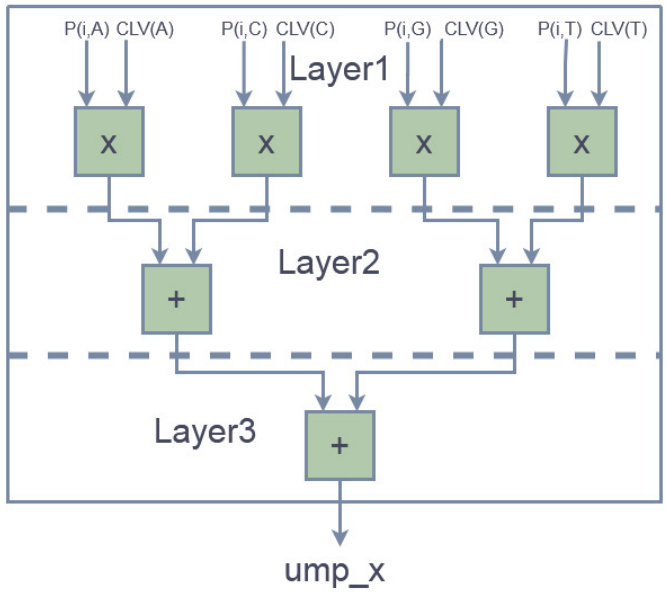
Στην ενότητα αυτή εξετάζουμε τη σχεδίαση των βασικών μονάδων που θα υλοποιηθούν σε αναδιατασσόμενη λογική.Η σχεδίαση που υλοποιούμε, η οποία παρουσιάζεται στο διάγραμμα της εικόνας 4.1, διαβάζει από τη μνήμη τα δυο διανύσματα θέσης από τους απογόνους και παράγει στην έξοδο το διάνυσμα θέσης του προγόνου.Η διαδικασία αυτή επαναλαμβάνεται διαδοχικά N φορές, όσα και τα **sites** της αλληλουχίας.Στην ενότητα αυτή παρουσιάζουμε τη βασική αρχιτεκτονική που υπολογίζει τις τιμές πιθανοφάνειας σε ένα **site**, για ένα ρυθμό ετερογένειας.Η μονάδα αυτή απότελεί βάση για τη κατασκευή του συνολικού **block**.Όπως παρουσιάζεται και στην εικόνα 4.1 η αρχιτεκτονική αποτελείται από 3 βασικές μονάδες, τις **ump_x_unit**,**x1px2_unit** και **x3_unit**.Οι επόμενες ενότητες περιγράφουν τη λειτουργία για κάθε μια από αυτές τις μονάδες και με ποιο τρόπο η αρχιτεκτονική τους υλοποιεί τα στάδια του RAxML αλγορίθμου.



Σχήμα 4.1: Το παραπάνω διάγραμμα παρουσιάζει τη διαδρομή δεδομένων για τον υπολογισμό του διανύσματος σε ένα site της αλληλουχίας του προγονικού κόμβου.Το datapath αποτελείται από 3 κύριες μονάδες που υλοποιούν τα διάφορα στάδια υπολογισμού της PLF.Οι μονάδες αυτές είναι οι ump_x_unit,x1px2_unit και x3_unit.

Η αριθμητική που χρησιμοποιείται στη σχεδίαση είναι floating point double precision, το οποίο σημαίνει ότι κάθε τιμή του διανύσματος πιθανότητας σε κάποια θέση αναπαριστάται από ένα σύνολο 64 bits.Η εφαρμογή single precision αριθμητικής μπορεί να μειώσει τις απαιτήσεις μνήμης κατά 50% σε σχέση με τη floating point double precision ωστόσο υπάρχει η πιθανότητα να εισάγει αριθμητική αστάθεια με αριθμητικές υπερχειλίσεις, ειδικά σε datasets με περισσότερα των 1000 taxa [9].Οι μονάδες που πραγματοποιούν πολλαπλασιασμούς και προσθέσεις στη σχεδίαση είναι διπλής ακρίβειας.

4.1.1 Υπολογισμός των τιμών **ump_x**



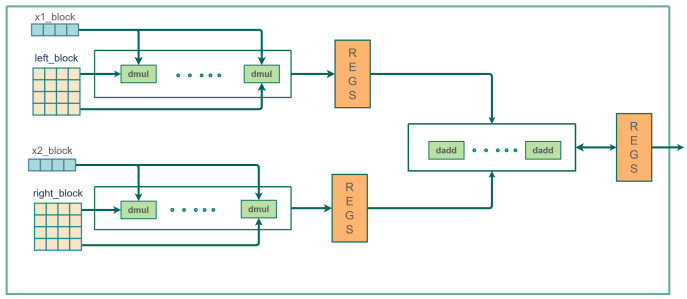
Σχήμα 4.2: Το παραπάνω διάγραμμα παρουσιάζει την αλληλουχία των πράξεων που περιγράφονται στη σχέση 4.3 και υπολογίζονται στη μονάδα ump_x.

Ο υπολογισμός για κάθε θέση του διανύσματος πιθανότητας ξεκινάει με τον υπολογισμό των αθροισμάτων των γινομένων μεταξύ των διανυσμάτων x1,x2 και των αντίστοιχων πινάκων υποκατάστασης left,right.Οι υπολογισμοί αυτοί εντοπίζονται στο LOOP4 του αλγορίθμου και αποτελούν τα δύο αθροίσματα της σχέσης 4.1.Ο αλγόριθμος που χρησιμοποιείται στο RAxML δίνει συμβατικά στη τιμή του πρώτου αθροίσματος που αφορά τον απόγονο που βρίσκεται στο αριστερό κλαδί με διάνυσμα πιθανότητας x1 την ονομασία ump_x1 και ump_x2 στο άθροισμα που αφορά τον απόγονο που βρίσκεται στο δεξί κλαδί με διάνυσμα πιθανότητας x2.Οι τιμές ump_x1,ump_x2 που παράγονται από τα συγκεκριμένα αθροίσματα αφορούν τις πιθανοφάνειες για μια μόνο κατάσταση από το σύνολο των τεσσάρων πιθανών καταστάσεων {A,C,G,T} που μπορεί να βρίσκεται η συγκεκριμένη θέση της αλληλουχίας του προγόνου.Κάθε επανάληψη του LOOP4 υπολογίζει το ζεύγος των αθροισμάτων από το γινόμενο των οποίων προκύπτει η πιθανοφάνεια κάθε κατάστασης.Η σχέση 4.3 παρουσιάζει τις πράξεις που προκύπτουν αν αναλυθεί οποιοδήποτε από τα δύο αθροίσματα της σχέσης 4.1 και το σχήμα της εικόνας 4.1 απεικονίζει την αλληλουχία των πράξεων.Το άθροισμα της σχέσης 4.3 είναι απλοποιημένο σε σχέση με αυτό της σχέσης 4.1 αφού έχουν αφαιρεθεί όλες οι παράμετροι που δε μεταβάλλονται.Κάθε τέτοιο άθροισμα υπολογίζεται σε ένα συγκεκριμένο site s

για ένα συγκεκριμένο ρυθμό ετερογένειας c.

$$\sum_{j=A}^{T} P_{i,j}\ CLV_j = P_{i,A}CLV_A + P_{i,C}CLV_C + P_{i,G}CLV_G + P_{i,T}CLV_T \tag{4.1}$$

όπου το i παίρνει τιμές από το σύνολο {A,C,G,T}).



Σχήμα 4.3: Σε αυτή την εικόνα παρουσιάζεται η σχεδίαση της μονάδας ump_x_unit.Οι αριθμητικές μονάδες που χρησιμοποιούνται εκτελούν τους υπολογισμούς με pipeline αρχιτεκτονική.Το πλήθος των αριθμητικών μονάδων που εκτελούν τους πολλαπλασιασμούς και τις προσθέσεις, όπως και το πλήθος των τιμών εισόδου των διανυσμάτων πιθανότητας και των τιμών εξόδου εξαρτάται από τη τεχνολογία που είναι διαθέσιμη για την υλοποίηση της αρχιτεκτονικής.Κάθε τεχνολογία δίνει διαφορετική πρόσβαση σε πόρους για την υλοποίηση της σχεδίασης και επομένως διαφορετικές δυνατότητες παραλληλισμού.

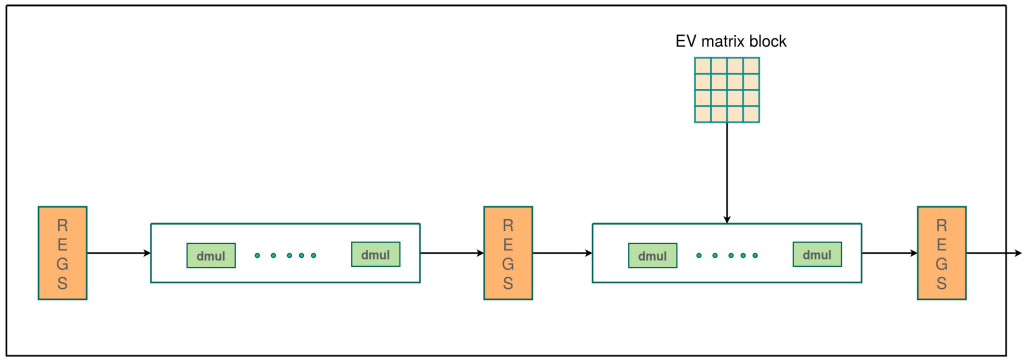
Σε κάθε PLF κλήση οι τιμές στους πίνακες υποκατάστασης παραμένουν σταθερές για όλα τα sites των διανυσμάτων x1,x2.Η μονάδα διαβάζει από τη μνήμη τις τιμές των left,right ,τις οποίες αποθηκεύει εσωτερικά σε κατάλληλα σύνολα καταχωρητών.Στη συνέχεια χρησιμοποιεί τις τιμές αυτές για κάθε νέο ζεύγος τιμών x1,x2 που διαβάζει ο επιταχυντής από τη μνήμη.Μόλις ολοκληρωθεί η ανάγνωση των πινάκων υποκατάστασης,σε κάθε κύκλο ρολογιού,έρχονται από τη μνήμη μια-μια οι τιμές του διανύσματος πιθανότητας.Η λογική που παρουσιάζεται στην εικόνα 4.2 υλοποιεί του υπολογισμούς της εικόνας 4.1.Τόσο οι πολλαπλασιαστές όσο και οι αθροιστές που χρησιμοποιούνται στη σχεδίαση υλοποιούνται με pipeline.Το γεγονός αυτό δίνει τη δυνατότητα σε κάθε κύκλο ρολογιού να δέχονται νέα είσοδο και κάθε μονάδα να υπολογίζει ταυτόχρονα πολλαπλές τιμές.Κάθε πολλαπλασιαστής αφού υπολογίσει το γινόμενο, στέλνει το αποτέλεσμα σε σύνολο καταχωρητών με το οποίο συνδέεται στην έξοδο του το σύνολο των πολλαπλασιαστών.Από εκεί οι τιμές ανά δύο οδηγούνται στην είσοδο του πρώτου συνόλου αθροιστών.Μόλις κάποιος αθροιστής υπολογίσει ένα αποτέλεσμα, το αποθηκεύει στο επόμενο σύνολο καταχωρητών.Η πράξη του τελικού αθροίσματος όπως παρουσιάζεται στο Layer 3 του σχήματος 4.2 υλοποιείται από το ίδιο σύνολο αθροιστών.

Η αρχιτεκτονική που προτείνουμε εκτελεί παράλληλα τον υπολογισμό των τιμών ump_x1 και ump_x2.Το σχήμα της εικόνας 4.2 παρουσιάζει τη διαδρομή των δεδομένων(datapath) για τον υπολογισμό κάθε ump_x διανύσματος.Το πλήθος πολλαπλασιαστών και αθροιστών στη σχεδίαση, και ως συνέπεια το πλήθος των πράξεων που εκτελούνται παράλληλα, εξαρτάται άμεσα από τους πόρους της τεχνολογίας που είναι διαθέσιμη για την υλοποίηση της σχεδίασης καθώς και από το πλήθος των τιμών των διανυσμάτων πιθανότητας που μπορούν να διαβαστούν στην είσοδο.

4.1.2 Υπολογισμός διανύσματος **x1px2**

Το επόμενο βήμα των υπολογισμών πραγματοποιεί η μονάδα x1px2_unit.Οι πράξεις που υλοποιεί η μονάδα φαίνονται στο Layer 1 του σχήματος σχήμα 4.5 ενώ η αρχιτεκτονική της φαίνεται στο σχήμα 4.4.

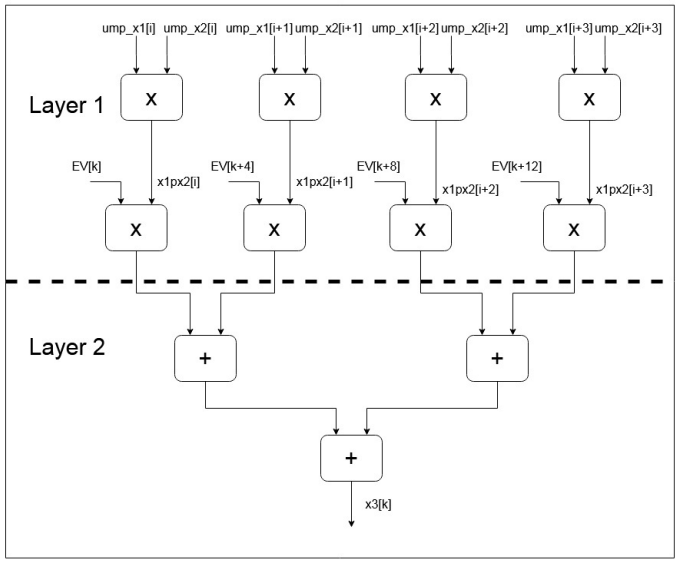
Η διεπαφή της μονάδας x1px2_unit συνδέεται στην είσοδο με τα σύνολα των καταχωρητών εξόδου από τη μονάδα ump_x_unit.Πριν ξεκινήσει οποιοδήποτε υπολογισμό φορτώνει τις τιμές του πίνακα EV.Οι υπολογισμοί ξεκινούν με το πολλαπλασιασμό των αθροισμάτων ump_x1 και ump_x1 ώστε να προκύψουν οι αρχικές τιμές του προγονικού διανύσματος πιθανοφάνειας CLV όπως περιγράφονται από την 4.1 για τις 4 πιθανές καταστάσεις{A,C,G,T}.Οι τιμές αυτές ονομάζονται συμβατικά στο RAxML, x1px2 και αποθηκεύονται σε κατάλληλο σύνολο καταχωρητών.Στη συνέχεια οι τιμές που προκύπτουν πολλαπλασιάζονται με τις τιμές του πίνακα EV όπως περιγράφεται στη σχέση 4.2 και αποθηκεύονται σε νέο σύνολο καταχωρητών.Οι τιμές αυτές προωθούνται στην έξοδο της μονάδας.Όπως και στη μονάδα ump_x_unit οι υπολογισμοί πραγματοποιούνται από pipeline αθροιστές και πολλαπλασιαστές.Η χρήση διαφορετικών συνόλων καταχωρητών για την αποθήκευση των αποτελεσμάτων μετά από κάθε πράξη δίνει τη δυνατότητα οι μονάδες που εκτελούν τις διαφορετικές εργασίες να αποτελέσουν ένα ενιαίο pipeline.Με αυτό το τρόπο μπορούν να υπολογίζονται παράλληλα διανύσματα πιθανότητας διαφορετικών sites της αλληλουχίας.



Σχήμα 4.4: Σε αυτό το σχήμα παρατηρείται η αρχιτεκτονική της μονάδας υπολογισμού **x1px2_unit**.Η μονάδα x1px2_unit δέχεται στην είσοδο τις τιμές εξόδου από το ump_x_unit τις τιμές από το RAxML-specific πίνακα ιδιοτιμών EV.Οι πράξεις με αυτά τα δεδομένα υπολογίζουν τις κλιμακωμένες τιμές x1px2.Το x1px2_unit συνδέεται στην έξοδο με το x3_unit, το οποίο υπολογίζει τις τελικές τιμές.

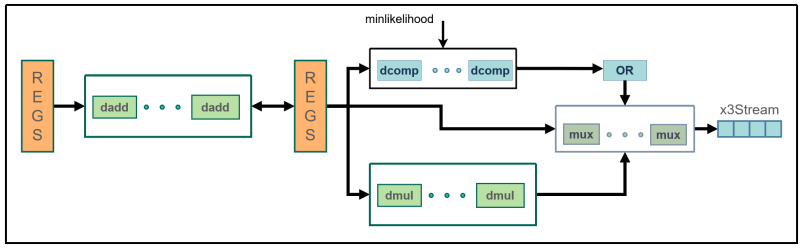
4.1.3 Υπολογισμός διανύσματος **x3**

Οι υπολογισμοί του διανύσματος πιθανότητας ολοκληρώνονται στη μονάδα x3_unit.Η μονάδα αυτή εκτελεί τις απαιτούμενες προσθέσεις των γινομένων που προέκυψαν από το x1px2_unit για την ολοκλήρωση του υπολογισμού της σχέσης 4.2, η οποία δίνει θεωρητικά τις τελικές τιμές του προγονικού διανύσματος πιθανότητας.Ωστόσο ο αλγόριθμος του RAxML ολοκληρώνει τον υπολογισμό με ένα RAxML-specific έλεγχο ώστε όλες οι τιμές να μην υπερβαίνουν ένα κατώτατο επιτρεπτό όριο.Όταν υπολογίζεται η πιθανοφάνεια φυλογενετικών δέντρων μεγάλου μεγέθους με βάση τον αλγόριθμο του Felsenstein υπάρχει ρίσκο αριθμητικής υπορροής κατά τη διάρκεια υπολογισμού των διανυσμάτων πιθανότητας στους εσωτερικούς κόμβους.Στο RAxML το πρόβλημα αυτό αντιμετωπίζεται με την εφαρμογή αριθμητικής κλιμάκωσης.Στη περίπτωση που όλες οι τιμές του διανύσματος πιθανότητας σε ένα site βρίσκονται κάτω από ένα συγκεκριμένο όριο,αυτές πολλαπλασιάζονται με μια σταθερά.Με το τρόπο αυτό αποφεύγεται η υπορροή.Το διάγραμμα 4.6 παρουσιάζει αναλυτικά τη σχεδίαση της μονάδας x3_unit.



Σχήμα 4.5: Γραφική αναπαράσταση των πράξεων για τον υπολογισμών των πρωταρχικών τιμών του διανύσματος πιθανότητας εξόδου, πριν την εφαρμογή κάποιου scaling.Οι πολλαπλασιασμοί στο πρώτο επίπεδο υπολογίζονται από τη μονάδα **x1px2_scaled_unit**, ενώ οι πράξεις στο δεύτερο επίπεδο από τη μονάδα site_vector_basic_unit.

Για το scaling η αρχιτεκτονική χρησιμοποιεί ένα σύνολο από συγκριτές που ελέγχουν για κάθε τιμή του αρχικού διανύσματος x3 αν υπερβαίνει τη τιμή minlikelihood.Σύμφωνα με τον αλγόριθμο του RAxML ακόμα και μια τιμή του διανύσματος να υπερβαίνει το minlikelihood πραγματοποιείται κλιμάκωση σε όλες τις τιμές του x3.Για το λόγο αυτό, το αποτέλεσμα από κάθε πράξη σύγκρισης εισάγεται σε μονάδα που εκτελεί τη λογική πράξη OR και το αποτέλεσμα δίνεται σε ένα σύνολο από πολυπλέκτες.Η λογική τιμή που έρχεται ως έξοδος από τη μονάδα OR επιλέγει αν η τελική έξοδος θα είναι το αρχικό διάνυσμα x3 ή οι τιμές x3 που έχουν κλιμακωθεί με κατάλληλη τιμή.

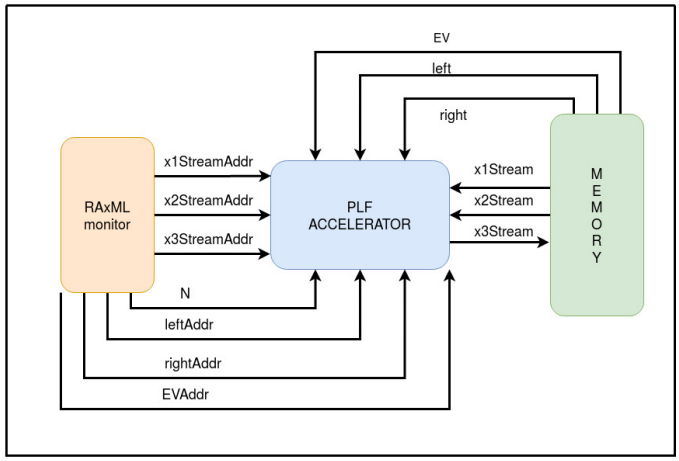


Σχήμα 4.6: Το σχήμα παρουσιάζει την αρχιτεκτονική της μονάδας υπολογισμού site_vector_basic_unit.Στην είσοδο δέχεται το σύνολο των τιμών x1px2 που έχουν κλιμακωθεί με τις τιμές του πίνακα EV και στην έξοδο παράγει τις τελικές τιμές πιθανοφάνειας του διανύσματος εξόδου.

4.1.4 Διεπαφή της μονάδας του επιταχυντή

Μέχρι στιγμής έχουμε παρουσιάσει το μονοπάτι δεδομένων σε αρχιτεκτονική διασωλύνωσης για τον υπολογισμό των τιμών σε ένα site του διανύσματος πιθανότητας.Σε αυτή την ενότητα παρουσιάζουμε τη διεπαφή της μονάδας.Η διεπαφή του plf_accel_unit παρουσιάζεται στο σχήμα 4.7.

Ο επιταχυντής επικοινωνεί με τη κύρια μνήμη και το RAxML monitor, ένα λογισμικό που βρίσκεται ανάμεσα στο πρόγραμμα του RAxML και τον επιταχυντή το οποίο παρουσιάζεται αναλυτικά στο επόμενο κεφάλαιο.Το monitor στέλνει στον επιταχυντή όλες τις απαραίτητες διευθύνσεις για να λάβει από τη μνήμη τα δεδομένα που χρειάζονται για την εκτέλεση της PLF κλήσης και τον ακέραιο αριθμό N που δείχνει το πλήθος των sites της αλληλουχίας.Τα αποτελέσματα του επιταχυντή αποθηκεύονται απευθείας στη μνήμη.



Σχήμα 4.7: Διεπαφή της μονάδας **plf_accel_unit**

4.2 Επέκταση του επιταχυντή για εκτέλεση περισσότερων **PLF** κλήσεων

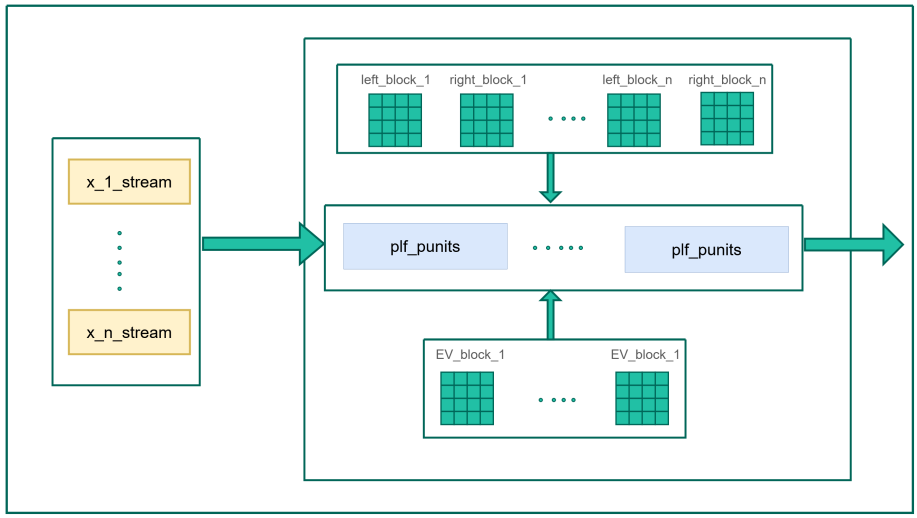
Αυτή η ενότητα παρουσιάζει την αρχιτεκτονική της μονάδας plf_accel_ts_unit.Η plf_accel_ts_unit αποτελεί επέκταση της αρχιτεκτονικής υπολογισμού PLF κλήσεων της προηγούμενης ενότητας.Υλοποιείται με σκοπό τον υπολογισμό πολλών διαδοχικών κλήσεων της PLF μέσα από μια μόνο κλήση της συνάρτησης υλικού.Με τον όρο διαδοχικές εννοούνται οι PLF κλήσεις που υπολογίζουν το διάνυσμα πιθανότητας κόμβου-προγόνου από περισσότερους από 2 κόμβους-απογόνους που συνδέονται μεταξύ τους σε δενδρική δομή.Με την plf_accel_ts_unit επιτυγχάνεται μεγαλύτερη συμπίεση δεδομένων στη μνήμη,σε μια φυλογενετική ανάλυση, μειώνοντας ακόμα περισσότερο τη ποσότητα του χρόνου που

προστίθεται στο χρόνο εκτέλεσης λόγω επανυπολογισμού διανυσμάτων που δε βρίσκονται στη μνήμη.

Ο λόγος που προτιμάται αυτή η αρχιτεκτονική δενδρικής δομής αντι κάποιας αρχιτεκτονικής όπου απλώς υπολογίζει παράλληλα PLF κλήσεις για ασύνδετους μεταξύ τους κόμβους είναι ο τρόπος που παράγονται και στέλνονται PLF κλήσεις από το monitor προς τον επιταχυντή.Οι κλήσεις παράγονται με post-order διάσχιση του φυλογενετικού δέντρου.Σε αυτή την ενότητα θα παρουσιαστεί μια γενική αρχιτεκτονική επιταχυντή που εκτελεί όλες τις απαραίτητες PLF κλήσεις για τον υπολογισμό του κοινού προγόνου για n κόμβους, όπου $n > 2$.

Η σχεδίαση του plf_accel_ts_unit στηρίζεται στην αρχιτεκτονική του plf_accel_unit.Αρχικά παρουσιάζεται η μονάδα plf_block_x_N,η οποία δίνει τη δυνατότητα ταυτόχρονης εκτέλεσης N ανεξάρτητων μεταξύ τους PLF κλήσεων.Στη συνέχεια η αρχιτεκτονική του plf_accel_ts_unit χρησιμοποιεί τη παράλληλη εκτέλεση για ένα σύνολο κλήσεων στο οποίο υπάρχουν PLF κλήσεις που τα δεδομένα εισόδου τους εξαρτώνται από τα δεδομένα εξόδου κάποιες άλλες PLF κλήσεις.Με αυτό το τρόπο μπορεί να υπολογίσει ένα κόμβο-πρόγονο από N κόμβους απογόνους του όπως φαίνεται στο σχήμα της εικόνας 4.8.

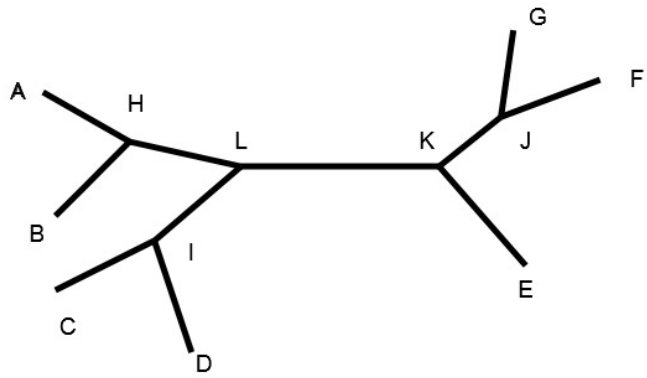
Το διάγραμμα της εικόνας 4.8 παρουσιάζει τη βασική σχεδίαση για παράλληλη εκτέλεση ανεξάρτητων μεταξύ τους PLF κλήσεων.Η μονάδα περιλαμβάνει σύνολα καταχωρητών για την αποθήκευση των τιμών όλων των απαραίτητων πινάκων που χρειάζονται για τον υπολογισμό των PLF κλήσεων και ένα σύνολο από μονάδες υπολογισμού plf_accel_unit.Οι μονάδες αυτές λειτουργούν ανεξάρτητα η μια από την άλλη και δε συνδέονται με κανένα τρόπο μεταξύ τους.Το πλήθος N των μονάδων plf_accel_unit που μπορούν να ενσωματωθούν σε μια plf_block_x_N μονάδα εξαρτάται από τους πόρους της διαθέσιμης τεχνολογίας και τις ανάγκες της εφαρμογής που προκύπτουν από το μέγεθος των φυλογενετικών αναλύσεων.



Σχήμα 4.8: Το διάγραμμα παρουσιάζει τη σχεδίαση επιταχυντή, ο οποίος αποτελείται από πολλαπλά cores του plf_accel_unit και επιτρέπει τη παράλληλη εκτέλεση πολλαπλών κλήσεων της PLF.

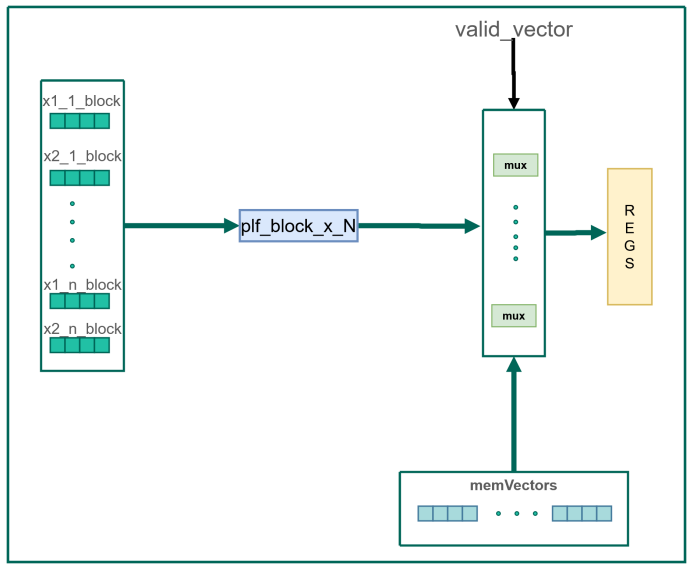
Στο διάγραμμα της εικόνας 4.11 παρουσιάζεται η βασική σχεδίαση του επιταχυντή plf_accel_ts_unit.Ο επιταχυντής αποτελείται από 2 επίπεδα.Στο πρώτο επίπεδο ο επιταχυντής αρχικά διαβάζει από τη μνήμη τα διανύσματα πιθανότητας εισόδου.Κάθε ζεύγος διανυσμάτων αντιστοιχίζεται μέσα στο plf_block_x_N σε ένα plf_punit.Όπως φαίνεται στο σχήμα 4.8 οι PLF κλήσεις για τον υπολογισμού του κοινού προγόνου N κόμβων χωρίζονται σε επίπεδα.Στο πρώτο επίπεδο υπολογίζονται οι κλήσεις με είσοδο τα διανύσματα των N κόμβων-απογόνων ενώ το τελευταίο επίπεδο αποτελείται από μια κλήση που υπολογίζει το κοινό-πρόγονο.Σε κάθε επίπεδο υλοποιείται και ένα πλήθος παράλληλων PLF κλήσεων. Οι κλήσεις αυτές υλοποιούνται από ένα σύνολο μονάδων plf_block_x_N που επικοινωνούν μεταξύ τους.Μια μονάδα plf_block_x_N υλοποιεί τις κλήσεις σε ένα επίπεδο και στέλνει τα αποτελέσματα στη μονάδα που αντιστοιχίζεται στο επόμενο σε σειρά επίπεδο.Κάθε plf_block_x_N δεν εκτελεί το ίδιο πλήθος κλήσεων.Το πλήθος των κλήσεων που εκτελεί κάθε μονάδα διαφέρει ανάλογα με τη συγκεκριμένη υλοποίηση της αρχιτεκτονικής.

Οι κόμβοι που βρίσκονται στη μνήμη και δίνονται ως ορίσματα στη κλήση του επιταχυντή υπάρχει περίπτωση να μη βρίσκονται στην ίδια απόσταση από το κοινό τους πρόγονο.Αυτό σημαίνει ότι υπάρχει περίπτωση τα δεδομένα που στέλνονται στο επόμενο επίπεδο από το plf_block_x_N να μην είναι αποτέλεσμα κάποιου υπολογισμού αλλά το αρχικό διάνυσμα εισόδου.Ακολουθεί παράδειγμα με βάση το δέντρο της εικόνας 4.10 για τη καλύτερη κατανόηση.



Σχήμα 4.9: Το σχήμα παρουσιάζει ένα υποθετικό φυλογενετικό δέντρο.

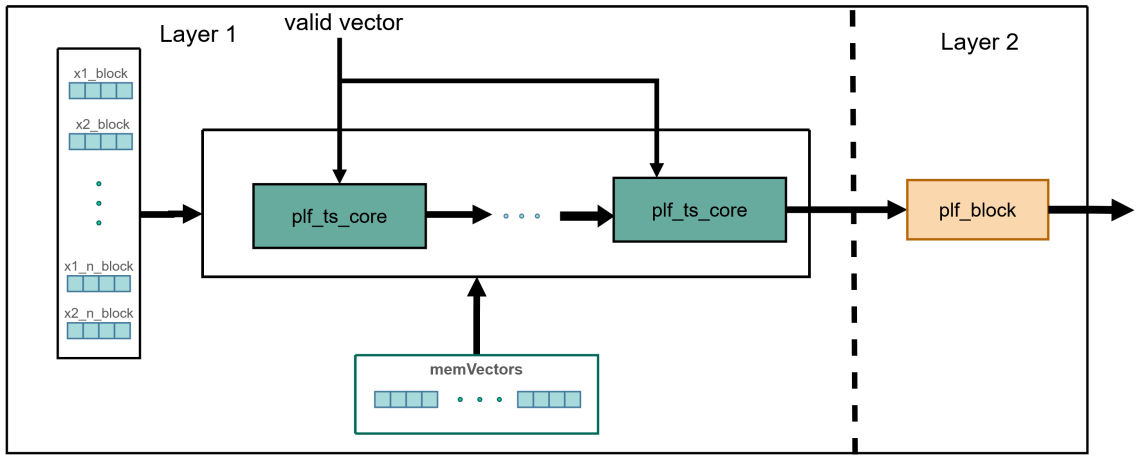
Στο σχήμα 4.10 παρουσιάζεται ένα υποθετικό φυλογενετικό δέντρο.Θεωρουμε κλήση προς τον επιταχυντή για τον υπολογισμό του κόμβου L και ότι οι κόμβοι A,B,I βρίσκονται στη μνήμη.Παρατηρούμε ότι οι απόγονοι του κόμβου L,που απαιτούνται για τον υπολογισμό του, δε βρίσκονται στην ίδια απόσταση από αυτόν.Ο υπολογισμός του κόμβου L αποτελείται από δύο στάδια.Στο πρώτο στάδιο υπολογίζεται ο κόμβος H και στο δεύτερο στάδιο με τη χρήση του αποτελέσματος και των τιμών του κόμβου I που έρχονται από τη μνήμη υπολογίζεται το διάνυσμα πιθανότητας του κόμβου L.Η σχεδίαση του επιταχυντή πρέπει να μπορεί να αποφασίσει σε ποιο στάδιο υπολογισμού χρησιμοποιούνται τα δεδομένα κάθε διανύσματος εισόδου.Κάθε μονάδα plf_block_x_N που μπορεί να εκτελεί παράλληλους PLF υπολογισμούς συνοδεύεται από ένα σύνολο πολυπλεκτών.Τα διανύσματα πιθανότητας δίνονται ως είσοδος τόσο στη μονάδα plf_block_x_N όσο και στο σύνολο των πολυπλεκτών.Οι πολυπλέκτες λαμβάνουν ως δεύτερο όρισμα τα αποτέλεσμα από τους υπολογισμούς του plf_block_x_N και στη συνέχεια με τη βοήθεια του διανύσματος valid_vector αποφασίζουν αν θα προχωρήσει το αποτέλεσμα του plf_block_x_N ή οι τιμές του αρχικού διανύσματος.Το διάνυσμα valid_vector αποτελείται από boolean τιμές και δίνει τη πληροφορία σε κάθε επίπεδο ποιοι υπολογισμοί είναι έγκυροι.



Σχήμα 4.10: Το σχήμα παρουσιάζει το plf_ts_core_unit, που αποτελεί το βασικό υπολογιστικό block του plf_accel_ts_unit.Η μονάδα πραγματοποιεί υπολογισμούς με τη βοήθεια του plf_block_N που εκτελεί παράλληλα PLF κλήσεις.Το valid vector δείχνει για κάθε υπολογιζόμενο διάνυσμα από το plf_block_N αν βρίσκεται στη μνήμη ή όχι.Αν κάποιο διάνυσμα δε βρίσκεται στη μνήμη στην έξοδο περνάει η τιμή που υπολογίστηκε από το plf_block_N.Σε αντίθετη περίπτωση στην έξοδο περνάει η τιμή που διαβάζεται από τα memVectors.

Κάθε κλήση από το λογισμικό προς τον επιταχυντή παίρνει σαν ορίσματα τις διευθύνσεις μνήμης των κόμβων για τους οποίους υπολογίζεται ο κοινός πρόγονος.Το πλήθος των ορισμάτων είναι συγκεκριμένο και εξαρτάται από τη διαθέσιμη τεχνολογία η οποία καθορίζει το πλήθος των PLF κλήσεων που μπορούν να ενσωματωθούν σε μια αρχιτεκτονική και επομένως το πλήθος των κόμβων για τους οποίους δίνεται η δυνατότητα να υπολογιστεί ο κοινός πρόγονος με κλήση προς τον επιταχυντή.Η κλήση που θα δημιουργηθεί από το λογισμικό προς τον επιταχυντή στη περίπτωση που έχουμε κόμβους που δε βρίσκονται στη ίδια απόσταση από το κοινό πρόγονο, λαμβάνει ως ορίσματα τις διευθύνσεις μνήμης για τους κόμβους που βρίσκονται στη μνήμη ενώ τα υπόλοιπα ορίσματα θεωρούνται "σκουπίδια".Στο παράδειγμα που εξετάζεται αν ήταν διαθέσιμος επιταχυντής που μπορεί να υπολογίσει το κοινό πρόγονο για 4 κόμβους θα λάμβανε στη κλήση τις διευθύνσεις για τους κόμβους A,B,I και το 4^o όρισμα θα ήταν κάποια τυχαία μη έγκυρη τιμή.Οι τιμές του valid_vector πρέπει να είναι τέτοιες ώστε οι πολυπλέκτες να προωθήσουν στο επόμενο στάδιο τη τιμή από τον υπολογισμό του κόμβου H και το διάνυσμα πιθανότητας του κόμβου I μη λαμβάνοντας υποψιν τους υπολογιμούς που θα γίνουν στο plf_block_x_N με τις τιμές του κόμβου I.Η αρχιτεκτονική του plf_block_x_N παρουσιάζεται στην εικόνα 4.11.

Το δεύτερο στάδιο του plf_accel_ts_unit αποτελείται από ένα απλό plf_accel_unit που υπολογίζει το τελικό αποτέλεσμα.



Σχήμα 4.11: Στο παραπάνω σχήμα φαίνεται η abstract έκδοση του plf_accel_ts_unit.Υπολογίζει ένα κόμβο-πρόγονο από Ν κόμβους-απογόνους.Κάθε υλοποίηση του plf_accel_ts_unit απαιτείται με βάση τους πόρους της διαθέσιμης πλατφόρμας να προσδιορίσει τη παράμετρο Ν.Με μια κλήση του επιταχυντή υπολογίζονται οι PLF κλήσεις που προκύπτουν από μια post-order διάσχιση,η οποία ολοκληρώνεται σε Ν γνωστά διανύσματα,ενός εξεταζόμενου φυλογενετικού δέντρου σε μια φυλογενετική ανάλυση.Στο πρώτο layer του plf_accel_ts_unit οι κύριες υπολογιστικές μονάδες είναι οι plf_ts_core.Η σχεδίαση του plf_ts_core φαίνεται στην εικόνα 4.10.Στην έξοδο κάθε plf_ts_core δίνονται τα διανύσματα πιθανότητας από κάθε αναδρομική διάσχιση του δέντρου.Αυτά είτε υπολογίζονται από PLF κλήσεις είτε λαμβάνονται από το σύστημα μνήμης.Συνολικά στην έξοδο του layer 1 δίνονται τα διανύσματα των κόμβων της PLF κλήσης που υπολογίζουν το κοινό κόμβο-πρόγονο.

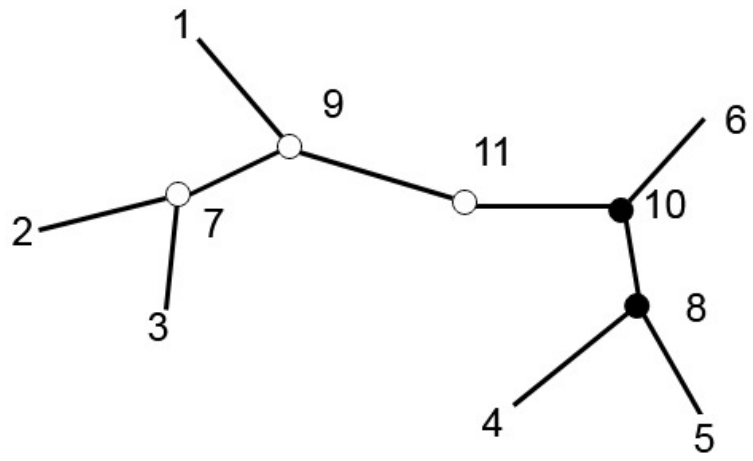
Κεφάλαιο 5

Υλοποίηση συστήματος διαχείρισης **PLF** κλήσεων του **RAxML**

Στο 5^o κεφάλαιο παρουσιάζουμε τη σχεδίαση και υλοποίηση συστήματος λογισμικού,το οποίο στην εργασία αναφέρεται ως **monitor** και διαχειρίζεται τις PLF κλήσεις που παράγονται στο RAxML.Συνδέεται με το RAxML και τον επιταχυντή που υπολογίζει κλήσεις της PLF.Κατά τη διάρκεια της φυλογενετικής ανάλυσης οι PLF κλήσεις του RAxML δεν υπολογίζονται απευθείας από το RAxML αλλά στέλνονται στο **monitor**, το οποίο στη συνέχεια αποφασίζει ποιες κλήσεις θα υπολογιστούν από τη μονάδα του επιταχυντή.

Το RAxML παράγει ένα σύνολο κλήσεων της PLF για τη κατασκευή ενός πρωταρχικού φυλογενετικού δέντρου και στη συνέχεια εφαρμόζει ευριστικό αλγόριθμο που αναδιατάσσει τη τοπολογία του αρχικού δέντρου με κριτήριο τη βελτιστοποίηση της συνολικής πιθανοφάνειας του.Σε κάθε εκτέλεση του RAxML το **monitor** δέχεται ως παραμέτρους το πλήθος των κόμβων στα φύλλα του δέντρου και το ποσοστό των κόμβων που δεν αποθηκεύονται στη μνήμη.Το ποσοστό αυτό ονομάζεται ποσοστό συμπίεσης.Με βάση το ποσοστό συμπίεσης και τις κλήσεις του RAxML το **monitor** παράγει και στέλνει για υπολογισμό στη μονάδα του επιταχυντή ένα νέο σύνολο PLF κλήσεων.Το σύνολο αυτό περιλαμβάνει ένα μέρος των κλήσεων που προέρχονται από το RAxML και όλες τις επιπλέον κλήσεις που απαιτούνται για τον υπολογισμό των κλήσεων του RAxML.

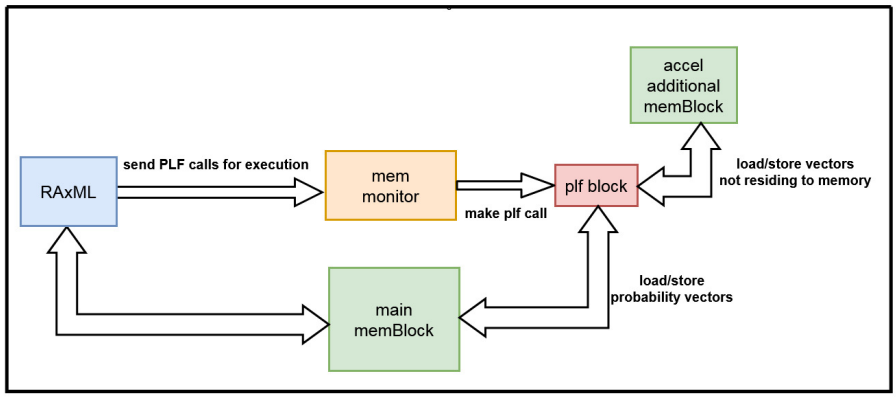
Η εργασία υλοποιεί δύο εκδοχές του **monitor**.Η βασική λειτουργικότητα που υλοποιείται και στις δύο εκδοχές είναι η αποστολή κλήσεων της PLF στον επιταχυντή και η εκτέλεση για κάθε κλήση της PLF των επιπλέον αναδρομικών κλήσεων που απαιτούνται για τον υπολογισμό των διανυσμάτων που δεν αποθηκεύονται στη μνήμη και τα οποία χρειάζονται στον υπολογισμό της αρχικής κλήσης.Με τον όρο αναδρομικές κλήσεις αναφέρονται οι κλήσεις της PLF σε μια **post-order** διάσχιση του φυλογενε-



Σχήμα 5.1: Στο παραπάνω σχήμα παρατηρείται ένα απλουστευμένο παράδειγμα φυλογενετικού δέντρου, στο οποίο με λευκούς κύκλους απεικονίζονται οι κόμβοι που αποθηκεύονται στη μνήμη και με μαύρους κύκλους αυτοί που δεν αποθηκεύονται.Οι κόμβοι με ID από 1 έως 6 θεωρείται ότι βρίσκονται τα φύλλα του δέντρου και a priori αποθηκεύονται στη μνήμη.Η κλήση που υπολογίζει το κόμβο 11 είναι η {9,10,11}.Ο κόμβος 9 βρίσκεται στη μνήμη και αρκεί ο επιταχυντής να διαθέτει τη κατάλληλη διεύθυνση μνήμης για να λάβει τα δεδομένα.Αντίθετα ο κόμβος 10 δεν βρίσκεται στη μνήμη.Αναδρομικές είναι όλες αυτές οι κλήσεις που θα χρειαστεί να εκτελέσουμε με post-order διάσχιση του δέντρου για τον υπολογισμό του διανύσματος στο κόμβο 10.

τικού δέντρου που πραγματοποιείται κάθε φορά που ζητούνται τα δεδομένα ενός διανύσματος πιθανότητας σε κάποιο κόμβο που δε βρίσκεται στη μνήμη.Κάθε φορά που πραγματοποιείται κλήση της PLF η οποία απαιτεί κόμβους που δε βρίσκονται στη μνήμη διεξάγεται post-order διάσχιση του δέντρου μέχρι να εντοπιστούν κόμβοι που βρίσκονται στη μνήμη.Δεδομένου ότι δεν αποθηκεύονται όλα τα διανύσματα πιθανότητας στη μνήμη, κάθε φορά που υπολογίζεται κάποιο διάνυσμα υπάρχει πιθανότητα επανυπολογισμού κάποιου από τα προγονικά διανύσματα πιθανότητας ώστε να καταστεί δυνατόν να υπολογιστεί ο προγονικός κόμβος της αρχικής κλήσης.Οι κλήσεις αυτές υπολογίζονται από τον επιταχυντή του συστήματος που υλοποιείται σε αναδιατασσόμενη λογική.Οι επιπλέον κλήσεις που πραγματοποιούνται από το σύστημα διαχείρισης της μνήμης επηρεάζουν το χρόνο εκτέλεσης της φυλογενετικής ανάλυσης.Με αυτό το σύστημα υλοποιείται η ανταλλαγή πόρων μνήμης με κάποιο επιπλέον χρόνο εκτέλεσης λόγω των επιπρόσθετων υπολογισμών.Η διαφορά μεταξύ των δύο εκδόσεων βρίσκεται στο τρόπο με τον οποίο υπολογίζεται το πλήθος των αναδρομικών κλήσεων.Ο τρόπος διαφέρει ανάλογα με την αρχιτεκτονική του επιταχυντή.

Στη πρώτη έκδοση θεωρείται ότι ο επιταχυντής που υπολογίζει τις κλήσεις της PLF μπορεί να υπολογίσει το προγονικό διάνυσμα πιθανότητας που προκύπτει από δύο διανύσματα πιθανότητας απογόνων.Στη δεύτερη έκδοση του συστήματος θεωρούμε ότι οι



Σχήμα 5.2: Σε αυτό το σχήμα παρουσιάζεται ο τρόπος με τον οποίο επικοινωνούν και ανταλλάσσουν δεδομένα το RAxML, το monitor, η μνήμη και η μονάδα του επιταχυντή που εκτελεί τους υπολογισμούς των διανυσμάτων πιθανότητας των κόμβων του φυλογενετικού δέντρου με βάση την PLF.Η διαχείριση των PLF κλήσεων υλοποιείται από το monitor.Κάθε κλήση του RAxML δίνεται στο monitor.Το monitor για κάθε κλήση αποφασίζει αρχικά αν θα υπολογιστεί.Στη περίπτωση που αποφασίσει θετικά για τον υπολογισμό της κλήσης που έρχεται από το RAxML στέλνει τη κλήση αυτή καθώς και όλες τις επιπλέον κλήσεις που πιθανόν να χρειαστεί να υπολογιστούν στη μονάδα του επιταχυντή.Ο επιταχυντής δέχεται από το monitor τις διευθύνσεις των διανυσμάτων πιθανότητας που αντιστοιχίζονται στην υπολογιζόμενη κλήση και τους πίνακες πιθανοτήτων εισόδου.Στη περίπτωση που ο επιταχυντής υπολογίσει το διάνυσμα πιθανότητας ενός κόμβου που δε βρίσκεται στη μνήμη και το διάνυσμα αυτό χρειάζεται ως είσοδος σε επόμενο υπολογισμό αποθηκεύεται προσωρινά στην επιπρόσθετη μνήμη που επικοινωνεί αποκλειστικά με τον επιταχυντή.

κλήσεις της PLF υπολογίζονται από επιταχυντή δενδρικής δομής που παρουσιάζεται στο κεφάλαιο 5.Επεκτείνοντας τη συνάρτηση φυλογενετικής πιθανοφάνειας του RAxML υλοποιείται νέα συνάρτηση η οποία εφαρμόζει υπολογισμούς δενδρικής δομής.Ο επιταχυντής που υλοποιεί τη νέα συνάρτηση μπορεί να υπολογίσει σε μια κλήση του περισσότερες από μια PLF κλήσεις σε μια post-order διάσχιση του δέντρου.Η χρήση του επιταχυντή που επεκτείνει τους υπολογισμούς της φυλογενετικής συνάρτησης για περισσότερες κλήσεις αναμένεται να μειώνει δραστικά τις επιπλέον κλήσεις που πραγματοποιεί το σύστημα διαχείρισης μνήμης ενώ ταυτόχρονα ο χρόνος ολοκλήρωσης των υπολογισμών του είναι συγκρίσιμος με τον αρχικό επιταχυντή για αλληλουχίες με μεγάλο αριθμό από sites.Με αυτό το τρόπο αναμένεται για τον ίδιο βαθμό συμπίεσης δεδομένων να μειωθεί το χρονικό κόστος.Το γεγονός αυτό θα δώσει τη δυνατότητα εφαρμογής μεγαλύτερου βαθμού συμπίεσης δεδομένων στη μνήμη κατά τη διεξαγωγή μιας φυλογενετικής ανάλυσης.

5.1 Σχεδίαση διαχειριστή μνήμης

Το σύστημα μνήμης δέχεται στην είσοδο κλήσεις της φυλογενετικής συνάρτησης πιθανοφάνειας.Οι κλήσεις που δέχεται έχουν τη μορφή τριπλέτας $\{x1, x2, x3\}$, όπου $x1, x2, x3$ είναι οι μοναδικοί κωδικοί των διανυσμάτων πιθανότητας.Τα διανύσματα x1,x2

δινονται στην είσοδο της PLF ενώ το x3 παράγεται από την PLF στην έξοδο.

Η βασική λογική πάνω στην οποία σχεδιάζουμε το σύστημα μνήμης είναι η αποθήκευση ενός ποσοστού μονάχα των δεδομένων που ορίζεται από παράμετρο εισόδου στο σύστημα και η διαχείριση των υπόλοιπων διανυσμάτων με βάση τη γνώση της τριπλέτας που τα υπολογίζει.Το σύστημα διαχείρισης μνήμης κρατάει τη πληροφορία για κάθε διάνυσμα αν αποθηκεύεται ή όχι.Είναι απαραίτητο στη μνήμη να αποθηκεύονται όλα τα διανύσματα που βρίσκονται στα φύλλα έτσι ώστε κάθε διάσχιση του δέντρου για τον υπολογισμό ενός διανύσματος ανεξάρτητα από το ποσοστό των διανυσμάτων που δε βρίσκονται στη μνήμη και το πλήθος των κλήσεων που πραγματοποιούνται να καταλήγει σε κάτι που μπορεί να υπολογιστεί.Άλλωστε όπως δείξαμε και στο κεφάλαιο 2, η ποσότητα της μνήμης που απαιτείται για την αποθήκευση αυτών των διανυσμάτων θεωρείται αμελητέα σε σχέση με τις απαιτήσεις για το σύνολο των εσωτερικών κόμβων.

Από το σύνολο των εσωτερικών κόμβων επιλέγουμε με τυχαίο τρόπο για αποθήκευση όσα διανύσματα μας υποδεικνύει η παράμετρος συμπίεσης *compressFactor*.Οι κλήσεις της φυλογενετικής συνάρτησης που το αποτέλεσμα τους αναφέρεται σε διάνυσμα που βρίσκεται στη μνήμη υπολογίζονται και η μνήμη ενημερώνεται για τις νέες τιμές των διανυσμάτων. Στη περίπτωση που το διάνυσμα πρόγονος της τριπλέτας δεν αποθηκεύεται στη μνήμη, το σύστημα δε πραγματοποιεί κάποιο υπολογισμό αλλά μονάχα ενημερώνει τη κατάλληλη δομή για τα νέα διανύσματα τα οποία χρησιμοποιούνται για τον υπολογισμό του.Τα διανύσματα απόγονοι που υπολογίζουν ένα διάνυσμα πρόγονο που κατοικεί στη μνήμη μπορεί να αποθηκεύονται ή όχι. Αξιοποιώντας τη γνώση που διατηρεί το σύστημα για αυτά τα διανύσματα κάνουμε post-order διάσχιση στο κλαδί του φυλογενετικού δέντρου όπου το ζητούμενο διάνυσμα είναι ο κοινός πρόγονος με αναδρομικούς υπολογισμούς μέχρι τον υπολογισμό του ζητούμενου διανύσματος που κατοικεί στη μνήμη.Η κατάλληλη ισορροπία σε αυτό το σύστημα υπολογίζεται μεταξύ της ποσότητας μνήμης που εξοικονομείται και του εκτελέσιμου χρόνου που απαιτείται για τον επανυπολογισμό διανυσμάτων που δε βρίσκονται στη μνήμη.

5.1.1 Παραγωγή αιτημάτων προς τη μνήμη

Όπως έχει αναφερθεί οι κλήσεις που δέχεται το σύστημα μνήμης είναι κλήσεις της συνάρτησης φυλογενετικής πιθανοφάνειας που παράγονται από το RAxML. Το RAxML ξεκινάει τη φυλογενετική ανάλυση με τη κατασκευή ενός αρχικού δέντρου που περιλαμβάνει όλους τους οργανισμούς.Η πιθανοφάνεια ενός τέτοιου δέντρου εκκίνησης βελτιώνεται προοδευτικά με την εφαρμογή μικρών τοπολογικών αλλαγών στο δέντρο.Με αυτό το τρόπο το RAxML κατασκευάζει 20 διαφορετικά φυλογενετικά δέντρα, τα οποία συγκρίνει μεταξύ τους με βάση τη μέγιστη πιθανοφάνεια που έχει αποκτήσει το καθένα.

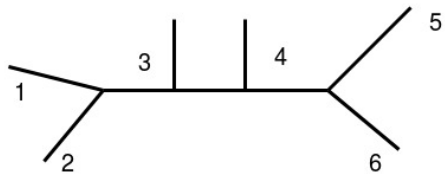
Κύριο αλγοριθμικό πρόβλημα που αντιμετωπίζουν οι φυλογενετικοί αναλυτές είναι ο τεράστιος αριθμός από πιθανές διαφορετικές τοπολογίες δέντρων για ένα συγκεκριμένο σύνολο από n ακολουθίες. Μπορεί να υπολογιστεί εύκολα ότι ακόμα και με πολύ μικρό αριθμό αρχικών κόμβων στα φύλλα ενός φυλογενετικού δέντρου παράγεται μια τεράστια ποσότητα από πιθανά δυαδικά άρριζα φυλογενετικά δέντρα.Χαρακτηριστικά για $n = 50$ παράγονται $2.84 * 10^{76}$ πιθανά δέντρα. Επομένως ο χώρος των φυλογενετικών δέντρων, που περιλαμβάνει το σύνολο των δέντρων που μπορούν να κατασκευαστούν από ένα πλήθος n κόμβων στα φύλλα, για αναλύσεις με περισσότερες των 20-25 ακολουθιών είναι πρακτικά αδύνατο με τις σημερινές δυνατότητες της τεχνολογίας να επεξεργαστεί στο σύνολο του. Για το λόγο αυτό εφαρμόζονται ευριστικοί αλγόριθμοι (heuristic algorithms) αναζήτησης πάνω στο χώρο των φυλογενετικών δέντρων προκειμένου να βρούμε το δέντρο που μεγιστοποιεί τη παράμετρο της πιθανοφάνειας.Είναι σημαντικό να σημειωθεί ότι με την εφαρμογή των ευριστικών αλγορίθμων δεν υπάρχει εγγύηση ότι έχει βρεθεί το δέντρο με τη μέγιστη πιθανοφάνεια.Οι ευριστικοί αλγόριθμοι βρίσκουν το δέντρο με τη μέγιστη πιθανοφάνεια από το υποσύνολο του χώρου των φυλογενετικών δέντρων που περιλαμβάνει ένα πλήθος από δέντρα που έχουν εξεταστεί.

5.1.2 Ευριστικοί Αλγόριθμοι

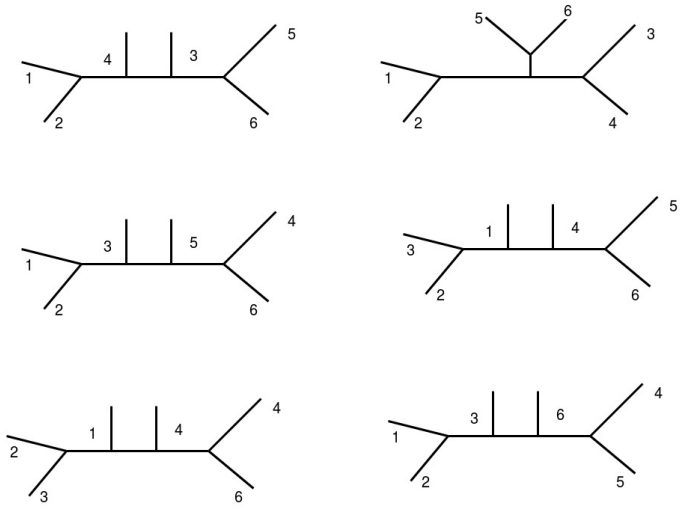
Οι βασικοί ευριστικοί αλγόριθμοι που χρησιμοποιούνται είναι ο Nearest Neighbour Interchange(NNI) και ο Subtree Pruning and Regrafting(SPR). Οι SPR κινήσεις επιβαρύνουν υπολογιστικά πολύ περισσότερο σε σχέση με τον NNI αλγόριθμο, αλλά μπορούν να παράγουν αναλύσεις με φυλογενετικά δέντρα τα οποία έχουν σημαντικά καλύτερες τιμές πιθανοφάνειας.

Αλγόριθμος **NNI**

Ο NNI εκτελεί την απλούστερη τοπολογική κίνηση.Ανταλλάσσει τις θέσεις των δύο υποδέντρων που συνδέονται από μια κοινή ακμή.Οι διάφορες μεταβολές της πιθανοφάνειας στο φυλογενετικό δέντρο υπολογίζοντα τοπικά για κάθε κίνηση που παράγει ο NNI. Αυτό έχει σα συνέπεια κάθε κίνηση του NNI να υπολογίζεται ιδιαίτερα γρήγορα.Παρότι οι κινήσεις του NNI δεν επιτρέπουν εντατική αναζήτηση στο χώρο των φυλογενετικών δέντρων, έχουν την ικανότητα να βελτιώσουν σημαντικά την αρχική τοπολογία.Ωστόσο είναι φανερό ότι ο αλγόριθμος NNI μπορεί να παγιδευτεί σε τοπικά μέγιστα μέσα στο χώρο των φυλογενετικών δέντρων.



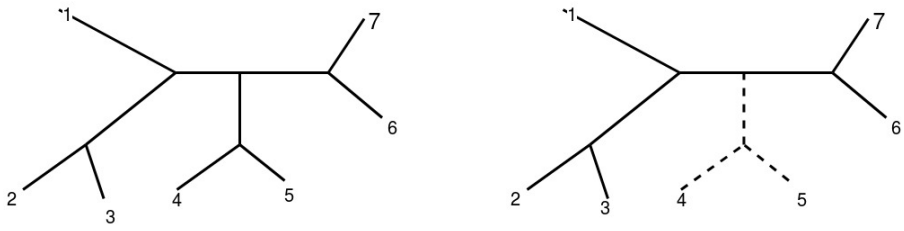
Σχήμα 5.3: Σε αυτό το σχήμα φαίνεται το αρχικό φυλογενετικό δέντρο πάνω στο οποίο θα εφαρμοστεί ο NNI αλγόριθμος.



Σχήμα 5.4: Σε αυτό το σχήμα παρατηρούνται όλες οι νέες τοπολογίες που προκύπτουν με την εφαρμογή του NNI αλγορίθμου στο αρχικό δέντρο.

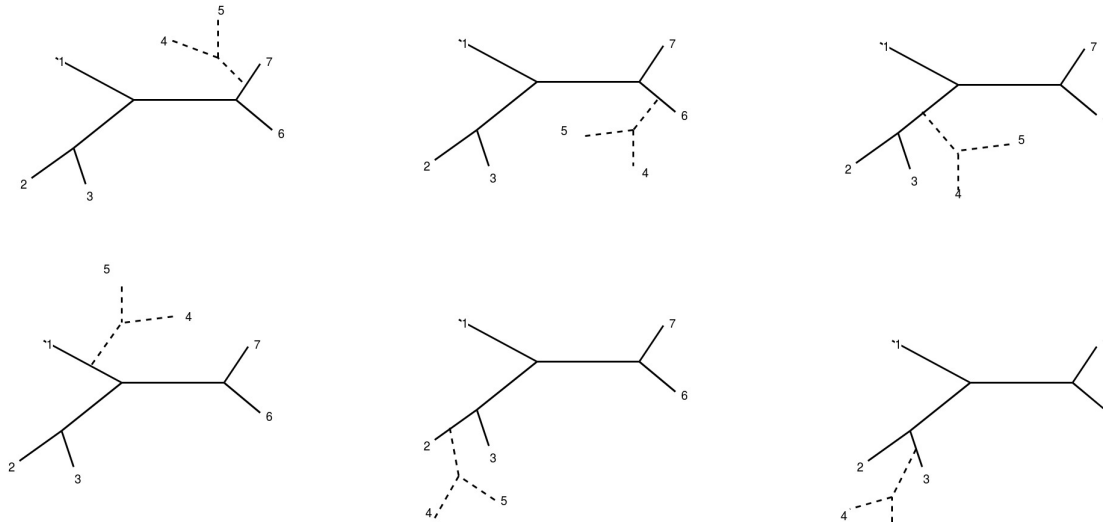
Αλγόριθμος **SPR**

Οι αναδιατάξεις πραγματοποιούνται με SPR(subtree pruning and regrafting) κινήσεις.Ο SPR είναι ένας ευριστικός αλγόριθμος που πραγματοποιεί υψηλού εύρους αναζητήσεις μέσα στο χώρο των φυλογενετικών δέντρων με αναδιατάξεις συγκεκριμένων τοπολογιών.Ξεκινάει από μια συγκεκριμένη τοπολογία και προχωράει με τη κοπή ενός υποδέντρου σε κάποιο σημείο του αρχικού δέντρου και την επανατοποθέτηση του σε ένα άλλο σημείο.Μόλις βρει κάποιο καλύτερο δέντρο, το χρησιμοποιεί ως δέντρο εκκίνησης για την επόμενη SPR κίνηση.Στα σχήματα 4.4 και 4.5 παρουσιάζονται σε ένα απλό παράδειγμα όλα τα βήματα που ακολουθεί ο SPR αλγόριθμος.



Σχήμα 5.5: Στο αριστερό δέντρο του σχήματος 4.5 φαίνεται το αρχικό δέντρο του παραδείγματος πάνω στο οποίο θα εφαρμοστεί ο αλγόριθμος SPR.Στο δέντρο στα δεξιά παρατηρείται με διακεκομμένη γραμμή το υποδέντρο που επιλέγεται για αποκοπή(pruning).

Η τοπολογία του δέντρου που χρησιμοποιείται στο παράδειγμα φαίνεται στο σχήμα 4.5.Αρχικά ο αλγόριθμος επιλέγει το κλαδί που θα αποκοπεί.Σκοπός του βήματος pruning είναι η αποκοπή κατάλληλου υποδέντρου το οποίο όταν σε επόμενο βήμα επανενωθεί σε διαφορετικά σημεία του υπόλοιπου δέντρου κάποια από τις νέες τοπολογίες που θα προκύψουν να υπολογιστεί με μεγαλύτερη πιθανοφάνεια από το αρχικό δέντρο.Στο σχήμα 4.5 παρατηρούμε με διακεκομμένη γραμμή το κλαδί που αποκόπτεται από το αρχικό δέντρο.Στην εικόνα 4.6 μπορούμε να παρατηρήσουμε όλα τα πιθανά δέντρα που μπορούν να προκύψουν μετά την εφαρμογή του βήματος regrafting.Στο βήμα αυτό επιλέγεται το κλαδί στο οποίο θα επανενωθεί το υποδέντρο που αποκόπηκε.Το 4.5 δείχνει τις νέες τοπολογίες που προκύπτουν για το φυλογενετικό δέντρο με την εφαρμογή του regrafting σε διάφορα σημεία του δέντρου.Οι κλήσεις της PLF για τον υπολογισμό της πιθανοφάνειας του δέντρου μετά την εφαρμογή κάποιας SPR κίνησης γίνονται μόνο για τους εσωτερικούς κόμβους που επηρεάζεται η πιθανφοάνεια τους.Για παράδειγμα δεν πραγματοποιούνται κλήσεις για τους κόμβους του υποδέντρου που επανασυνδέεται.



Σχήμα 5.6: Σε αυτό το σχήμα παρατηρούνται όλες οι νέες τοπολογίες που προκύπτουν όταν επανασυνδεθεί το υποδέντρο που αποκόπηκε στα διάφορα πιθανά σημεία.

5.1.3 Εφαρμογή αλγορίθμου **LSR** στο **RAxML**

Το RAxML κατασκευάζει ένα πρωταρχικό δέντρο εκκίνησης το οποίο βασίζεται στο κριτήριο της μέγιστης φειδωλότητας και το βελτιστοποιεί με μια παραλλαγή του αλγόριθμου SPR, τον Lazy Subtree Rearrangement(LSR) [22].Με στόχο να μειώσει τις υπολογιστικές απαιτήσεις του SPR αλγορίθμου, ο LSR συνδυάζει δύο τεχνικές.Πρώτον αναθέτει μια μέγιστη απόσταση μεταξύ του σημείου κοπής(prunning point) και του σημείου εισαγωγής για τις SPR κινήσεις για να περιορίσει το μέγεθος της περιοχής επανυπολογισμού.Η μέγιστη απόσταση του SPR ορίζεται στην αρχή της εκτέλεσης του προγράμματος.Στη συνέχεια ο LSR βελτιστοποιεί μόνο τα κλαδί που προέρχεται από το σημείο κοπής του υποδέντρου και τα 3 κλαδιά που δημιουργούνται στο σημείο εισαγωγής του υποδέντρου.

Οι κινήσεις του LSR επαναλαμβάνονται πολλές φορές, κάθε φορά χρησιμοποιώντας το βέλτιστο δέντρο μέχρι εκείνη τη στιγμή.Η τιμή της συνολικής πιθανοφάνειας βελτιστοποιείται ξανά για τα 20 καλύτερα δέντρα που έχουν βρεθεί, ρυθμίζοντας τα μήκη όλων των κλαδιών.Οι κινήσεις του LSR και οι βελτιστοποιήσεις για 20 καλύτερα δέντρα επαναλαμβάνονται μέχρι η εκτέλεση να οδηγηθεί σε κάποιο μέγιστο μέσα στο χώρο των φυλογενετικών δέντρων και να μην είναι δυνατόν να βρεθεί καλύτερο δέντρο.

5.2 Υλοποίηση βασικής έκδοσης διαχειριστή μνήμης

5.2.1 Βασικές δομές του συστήματος

Για την υλοποίηση της παραπάνω λογικής διαχείρισης των διανυσμάτων πιθανότητας και των κλήσεων της PLF ορίζουμε τις ακόλουθες δομές δεδομένων σε γλώσσα προγραμματισμού C.Με τις δομές αυτές οργανώνεται και αποθηκεύεται όλη η απαραίτητη πληροφορία.

- **probVecInfo_t:**

```
typedef struct
{
  int nodeId;
  bool stored;
  double *addr;
  double *left;
  double *right;
  double *ev;
}probVecInfo_t;
```

  Ο τύπος δομής probVecInfo_t εμπεριέχει όλα τα απαραίτητα δεδομένα για κάθε κόμβο ενός φυλογενετικού δέντρου.Η μεταβλητές nodeId και stored επιστρέφουν το μοναδικό κωδικό του κόμβου και τη πληροφορία αν το διάνυσμα πιθανότητας του κόμβου βρίσκεται στη μνήμη αντίστοιχα.Ο δείκτης addr επιστρέφει τη διεύθυνση του διανύσματος πιθανότητας στη περίπτωση που αυτό βρίσκεται στη μνήμη.Σε διαφορετική περίπτωση επιστρέφει τη τιμή NULL.Οι δείκτες left,right,ev δείχνουν στις διευθύνσεις μνήμης που είναι αποθηκευμένοι οι πίνακες που αντιστοιχούν στους απαραίτητους πίνακες για τον υπολογισμό του διανύσματος του κόμβου με τη χρήση της PLF.

- **callInfo_t:**

```
typedef struct
{
  int x1,
      x2,
      x3;
}callInfo_t;
```

  Κάθε μεταβλητή τύπου δομής callInfo_t αναπαριστά μια κλήση της PLF.Καταχωρεί τριπλέτες ακεραίων $\{x1, x2, x3\}$, όπου $x1, x2$ είναι τα μοναδικά αναγνωριστικά για τα διανύσματα-απογόνους και το $x3$ το αντίστοιχο αναγνωριστικό για το διάνυσμα-πρόγονο.Κάθε κλήση που φτάνει στο σύστημα έχει αυτή τη μορφή.Ακόμα η μονάδα MonitorFile είναι τύπου callInfo_t.

- **runtimeInfo_t:**

```
typedef struct {
  double *xAddr[3];
  double *left;
  double *right;
  double *ev;
}runtimeInfo_t;
```

Μια μεταβλητή τύπου *runtimeInfo_t* κρατάει τις διευθύνσεις μνήμης για τα δεδομένα εισόδου σε μια κλήση της φυλογενετικής συνάρτησης πιθανοφάνειας.Στο πίνακα xAddr βρίσκονται στις 2 πρώτες θέσεις οι διευθύνσεις για τα διανύσματα πιθανότητας των κόμβων-απογόνων x1,x2 και στη τρίτη η διεύθυνση που θα αποθηκευτεί το αποτέλεσμα, το υπολογιζόμενο διάνυσμα του κόμβου-προγόνου.Οι δείκτες left,right,ev αναφέρονται στις διευθύνσεις μνήμης που βρίσκονται οι αντίστοιχοι πίνακες υποκατάστασης και ιδιοδιανυσμάτων που είναι απαραίτητοι για την εκτέλεση της PLF.

- **runtimeInfo_tb_t:**

```
typedef struct
{
  double *xAddr[4];
  double *left[3];
  double *right[3];
  double *ev;
  bool valid[2];
}runtimeInfo_tb_t;
```

Ο τύπος δομής runtimeInfo_tb_t είναι ελαφρώς παραλλαγμένος σε σχέση με τον runtimeInfo_t για να εξυπηρετεί τις συγκεκριμένες απαιτήσεις για δεδομένα εισόδου από τη δεύτερη έκδοση της συνάρτησης που υπολογίζει το διάνυσμα πιθανότητας ενός κόμβου-προγόνου.Η ιδιαιτερότητα της συνάρτησης αυτής είναι ότι για κάθε PLF κλήση που δέχεται αναζητά αν τα διανύσματα των κόμβων-απογόνων βρίσκονται στη μνήμη και στη περίπτωση που αυτά δε βρίσκονται υπολογίζει με μια μοναδική κλήση σε μια νέα έκδοση της συνάρτησης που υλοποιεί την PLF το διάνυσμα του κοινού προγόνου.Η συνάρτηση αυτή παίρνει στην είσοδο 4 διευθύνσεις μνήμης που αντιστοιχίζονται σε 4 κόμβους του δέντρου.Οι διευθύνσεις αυτές αποθηκεύονται στα 4 πρώτα στοιχεία του διανύσματος xAddr.Από αυτούς τους κόμβους είναι δυνατόν να υπολογιστούν οι κόμβοι-απόγονοι της αρχικής κλήσης.Το διάνυσμα valid αποτελείται από 2 θέσεις και κάθε στοιχείο του λαμβάνει τιμές true ή false ανάλογα με το αν τα δεδομένα του αντίστοιχου κόμβου-απογόνου της αρχικής κλήσης χρειάζεται να υπολογιστούν ή βρίσκονται στη μνήμη.Στη περίπτωση που οι κόμβοι-απόγονοι της αρχικής κλήσης βρίσκονται στη μνήμη, το διάνυσμα memVec αποθηκεύει τις αντίστοιχες διευθύνσεις μνήμης.

- **accelMemInfo_t:**

```
typedef struct{
  double *addr;
  bool empty;

}accelMemInfo_t;
```

Η accelMemInfo αναφέρεται σε μια θέση της δευτερεύουσας μνήμης.Ο δείκτης addr δείχνει στα δεδομένα του διανύσματος πιθανότητας και η μεταβλητη empty δείχνει αν τα δεδομένα που βρίσκονται στη συγκεκριμένη θέση μνήμης απαιτούνται σε περαιτέρω υπολογισμούς ή γίνεται να αντικατασταθούν κάποιο νέο διάνυσμα.

5.2.2 Διαχείριση αιτημάτων προς τη μνήμη

Στην ενότητα αυτή παρουσιάζεται ο βασικός αλγόριθμος διαχείρισης των αιτημάτων προς τη μνήμη.

Algorithm 2 requestManager

procedure REQUESTMANAGER(CALLINFO_T INFO)

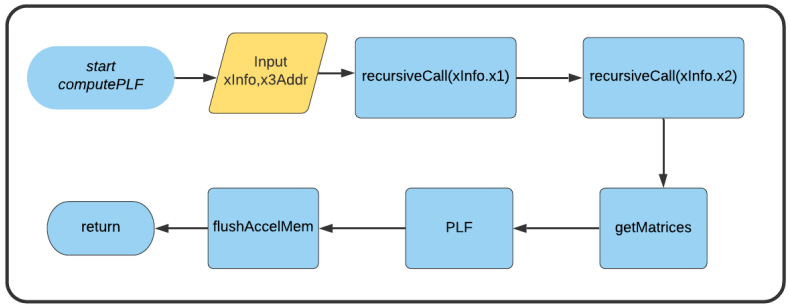
$inputs \leftarrow info$
if !$is\_stored(info.x3)$ **then**
  $MonitorEnumeration(info)$;
else
  $computePLF(info, getVecAddr(info.x3))$;
end if

Ο αλγόριθμος της *requestManager* δέχεται στην είσοδο τη τριπλέτα ακεραίων αριθμών $\{x1, x2, x3\}$ που αντιστοιχίζεται στην PLF κλήση.Η συνάρτηση *requestManager* αποφασίζει αν μια PLF κλήση θα υπολογιστεί ή η τριπλέτα των κόμβων της κλήσης θα αποθηκευτεί στο monitorFile.Αρχικά ο αλγόριθμος *requestManager* πραγματοποιεί κλήση της συνάρτησης is_stored, η οποία παίρνει ως όρισμα το id του διανύσματος προγόνου και επιστρέφει αν το διάνυσμα αποθηκεύεται στη μνήμη ή όχι.Αν δεν αποθηκεύεται καλείται η συνάρτηση MonitorEnumeration με όρισμα τη τριπλέτα εισόδου για να ενημερώσει τη καταχώρηση του που αντιστοιχίζεται στο κόμβο x3.Σε διαφορετική περίπτωση πραγματοποιείται κλήση της συνάρτησης computePLF για τον υπολογισμό της κλήσης.

5.2.3 Υπολογισμός των Κλήσεων της **PLF**

Η *computePLF* είναι η συνάρτηση που υπολογίζει μια PLF κλήση.Στην εικόνα 5.7 παρουσιάζεται το διάγραμμα ροής της συνάρτησης.Η παράμετρος εισόδου είναι η τριπλέτα

ακεραίων (x1,x2,x3) που αντιστοιχίζεται στην υπολογιζόμενη κλήση.Ορίζει τη μεταβλητή accelInputInfo, η οποία είναι τύπου runtimeInfo_t και αποθηκεύει τις διευθύνσεις μνήμης των δεδομένων που χρειάζονται για μια εκτέλεση της συνάρτησης PLF.Η computePLF καλεί τη συνάρτηση recursiveCall. Η recursiveCall παίρνει στην είσοδο το αναγνωριστικό id του κόμβου και επιστρέφει τη διεύθυνση μνήμης που έχει αποθηκευτεί το διάνυσμα.Οι διευθύνσεις αποθηκεύονται στο πίνακα xAddr της accelInputInfo.Αν το διάνυσμα δεν βρίσκεται στη μνήμη αλλά έχει υπολογιστεί με αναδρομική κλήση,η recursiveCall επιστρέφει και τη θέση που έχει αποθηκευτεί το διάνυσμα στο accelMemBlock, στις μεταβλητές x1,x2.Στη συνέχεια, με κλήση της συνάρτησης getMatrices ανακτώνται οι διευθύνσεις μνήμης που είναι αποθηκευμένοι οι πίνακες left,right,EV.Οι πίνακες αυτοί αντιστοιχίζονται στους κατάλληλους πίνακες υποκατάστασης(left,right) και ιδιοδιανυσμάτων(EV) που παρουσιάζονται στο κεφάλαιο 2.Στη συνέχεια η computePLF υπολογίζει το διάνυσμα x3 με κλήση της PLF και τέλος αποδεσμεύει τα block της accelMemBlock με τη flushAccelMem.



Σχήμα 5.7: Στο διάγραμμα ροής παρουσιάζεται η αλληλουχία των processes που εκτελούνται σε μια κλήσης της ComputePLF.

Algorithm 3 computePLF

procedure COMPUTEPLF(xInfo, x3Addr)
runtimeInfo_t accelInputInfo;
int x1 ← 0
int x2 ← 0

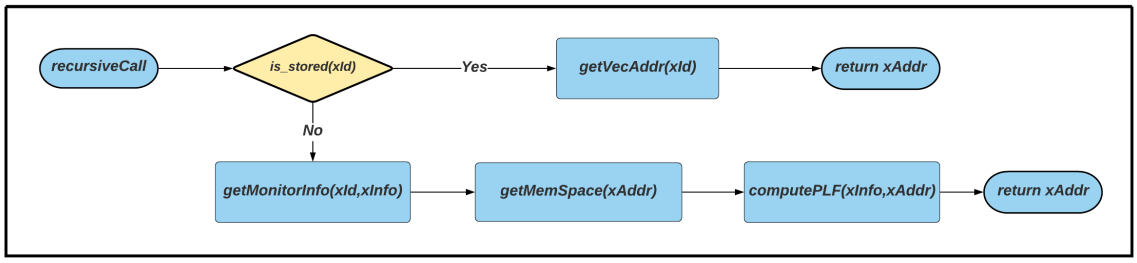
recursiveCall(*xInfo.x1*, *accelInputInfo.xAddr*[0], &*x1*);
recursiveCall(*xInfo.x2*, *accelInputInfo.xAddr*[1], &*x2*);
getMatrices(*xInfo.x3*, *accelInputInfo.left*, *accelInputInfo.right*, *accelInputInfo.EV*);

plf_accel(*accelInputInfo.xAddr*[0], *accelInputInfo.xAddr*[1], *x3Vec*, *accelInputInfo.left*,
accelInputInfo.right,accelInputInfo.EV,N);

flushAccelMem(*x1*, *x2*);

return 0

Η εικόνα 5.8 απεικονίζει το διάγραμμα ροής της συνάρτησης recursiveCall. Ο αλγόριθμος της recursiveCall ξεκινάει με τον έλεγχο αν το διάνυσμα με το αναγνωριστικό nodeId βρίσκεται στη μνήμη.Στο σημείο αυτό υπάρχουν δύο περιπτώσεις.Στη πρώτη περίπτωση το διάνυσμα του κόμβου απογόνου βρίσκεται στη μνήμη και με τη συνάρτηση getVecAddr(int x) βρίσκουμε τη διεύθυνση του διανύσματος.Στη δεύτερη περίπτωση το διάνυσμα δε βρίσκεται στη μνήμη και απαιτείται ο υπολογισμός του.Πριν τον υπολογισμό του διανύσματος χρειάζεται να δεσμευτεί ο κατάλληλος χώρος στο accelMemBlock για την αποθήκευση του.Το accelMemBlock αναπτύσσεται δυναμικά στη μνήμη. Αυτό σημαίνει ότι το monitor δεσμεύει μια αρχική ποσότητα μνήμης που αυξάνεται ανάλογα με τις απαιτήσεις των υπολογισμών.Το διάνυσμα πιθανότητας ενός κόμβου που αποθηκεύεται στην accelMemBlock βρίσκεται εκεί μέχρι να ολοκληρωθεί ο υπολογισμός του διανύσματος του κόμβου-προγόνου του.Μετά τον υπολογισμό η θέση ελευθερώνεται και μπορεί να αποθηκευτεί ένα άλλο διάνυσμα.Η συνάρτηση getMemSpace δεσμεύει ένα block στο accelMemBlock.Πρώτα ελέγχει αν υπάρχει διαθέσιμος χώρος στη μνήμη που έχει ήδη δεσμευτεί για το accelMemBlock.Αν υπάρχει επιστρέφει τη διεύθυνση μνήμης για το αντίστοιχο block και τον αριθμό του block.Αν δεν υπάρχει κάποιο διαθέσιμο, δεσμεύει καινούριο block στη μνήμη για την αποθήκευση του διανύσματος.Η getMonitorInfo(int x,callInfo_t xInfo) επιστρέφει τη τριπλέτα από το monitorFile που αντιστοιχίζεται στην PLF κλήση που υπολογίζει το ζητούμενο διάνυσμα-απογόνου.Στη συνέχεια εκτελείται αναδρομική κλήση της computePLF στην οποία δίνονται ως παράμετροι η τριπλέτα της κλήσης του κόμβου απογόνου και η διεύθυνση μνήμης που θα αποθηκευτεί το διάνυσμα.Με την ολοκλήρωση της κλήσης η διεύθυνση μνήμης αποθηκεύεται στην accelInputInfo.



Σχήμα 5.8: Η εικόνα παρουσιάζει το διάγραμμα ροής της recursiveCall.

Algorithm 4 recursiveCall

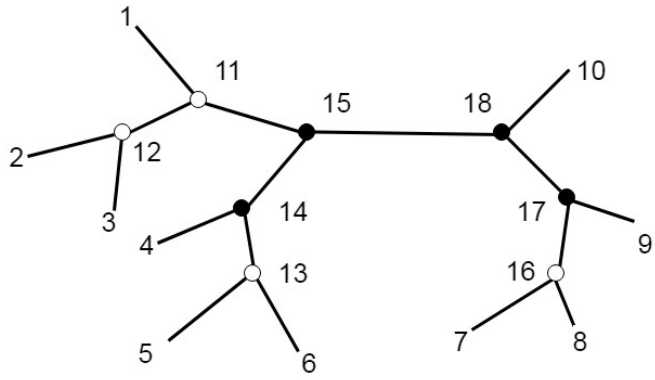
procedure RECURSIVECALL(xId, xAddr, accelMemBlockIndex)

```
int accelMemIdx ← 0;
callInfo_t info;

if is_stored(xId) then
    xAddr ← getVecAddr(nodeId);
else
    accelMemIdx ← getMemSpace();
    getMonitorInfo(xId, &info);
    computePLF(info, accelMemBlock[accelMemIdx].addr);
    xAddr ← accelMemBlock[accelMemIdx].addr;
    accelMemBlockIndex ← accelMemIdx;
end if

return 0
```

5.2.4 Εφαρμογή Λειτουργίας Συστήματος σε Φυλογενετικό Δέντρο

Σε αυτή την υποενότητα παρουσιάζεται ένα παράδειγμα λειτουργίας του συστήματος όταν δέχεται κλήσεις της PLF για τον υπολογισμό της πιθανοφάνειας του δέντρου που φαίνεται στο σχήμα 4.7.Θεωρούμε ότι η ανάλυση πραγματοποιείται με συμπίεση δεδομένων 50%.Η συμπίεση δεν αφορά το σύνολο των κόμβων αλλά μονάχα τους εσωτερικούς κόμβους του δέντρου.Όλοι οι κόμβοι που βρίσκονται στα φύλλα του δέντρου βρίσκονται στη μνήμη.Αυτό γίνεται για να υπάρχει η δυνατότητα υπολογισμού της πιθανοφάνειας του δέντρου ανεξάρτητα από το ποσοστό της συμπίεσης.Στο συγκεκριμένο παράδειγμα η συμπίεση της τάξης του 50% σημαίνει ότι μένουν εκτός μνήμης τα διανύσματα πιθανότητας για 4 κόμβους.Οι κόμβοι που μένουν εκτός μνήμης επιλέγονται με τυχαίο τρόπο.Στο παράδειγμα επιλέγονται οι κόμβοι 14,15,17,18.

Αρχικά η φυλογενετική ανάλυση επιλέγει με τυχαίο τρόπο τη ρίζα του δέντρου.Θωρούμε ότι η ρίζα βρίσκεται στο κλαδί που συνδέει τους κόμβους 15 και 18.Στη ρίζα αναθέτουμε τον ακέραιο αριθμό 19.Στη συνέχεια υλοποιείται post-order διάσχιση του δέντρου για τον υπολογισμό της αρχικής του πιθανοφάνειας.Στο σύστημα φτάνουν PLF κλήσεις οι οποίες αντιστοιχίζονται στην post-order διάσχιση του δέντρου.Ο παρακάτω πίνακας δείχνει το σύνολο των κλήσεων της PLF.



Σχήμα 5.9: Σε αυτό το σχήμα φαίνεται η τοπολογία του φυλογενετικού δέντρου στο οποίο πραγματοποιείται το παράδειγμα της ενότητας 4.2.6.Οι εσωτερικοί κόμβοι που αποθηκεύονται στη μνήμη αναπαριστώνται από λευκούς κόμβους ενώ αυτοί που μένουν εκτός μνήμης με μαύρους κόμβους.

x1	x2	x3
2	3	12
12	1	11
5	6	13
13	4	14
11	14	15
7	8	16
16	9	17
10	17	18
15	18	19

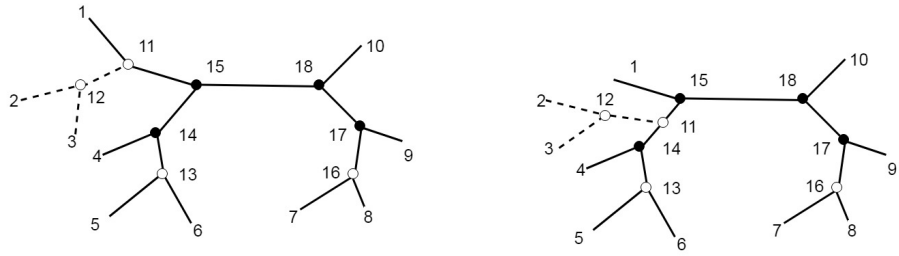
Από τις παραπάνω κλήσεις της PLF που στέλνονται στο σύστημα διαχείρισης της μνήμης κάποιες αφορούν κόμβους που δεν αποθηκεύονται στη μνήμη.Για παράδειγμα ο κόμβος 15 δεν αποθηκεύεται στη μνήμη και επομένως αν έρθει στο σύστημα ως είσοδος η τριπλέτα { 11,14,15 } η κλήση δεν θα υπολογιστεί.Αυτό που θα γίνει είναι να αποθηκευτεί η τριπλέτα της κλήσης στο *MonitorTable*.Το ίδιο γίνεται και για τους κόμβους 14,17,18.Αντίθετα τα δεδομένα του διανύσματος πιθανότητας του κόμβου 11

αποθηκεύονται στη μνήμη.Συνεπώς αν έρθει στην είσοδο του συστήματος κλήση της PLF για το κόμβο 11 θα υπολογιστεί.Η επιλογή της κλήσης που θα εκτελεστεί και της κλήσης που θα απορριφθεί αποθηκεύοντας τη τριπλέτα ακεραίων της κλήσης στο monitorTable υλοποιείται από τη *requestManager* που περιγράφηκε στη προηγούμενη ενότητα.

Στο παράδειγμα έχουμε θεωρήσει ότι η ρίζα βρίσκεται στο κλαδί που ενώνει τους κόμβους 15 και 18.Για να υπολογίσουμε τη ρίζα χρειάζεται να υπολογίσουμε τα διανύσματα πιθανότητας των κόμβων 15 και 18.Όταν φτάσει στο σύστημα κλήση PLF για τον υπολογισμό του διανύσματος πιθανότητας της ρίζας το σύστημα πραγματοποιεί εσωτερικά PLF κλήσεις για τον υπολογισμό των κόμβων αξιοποιώντας τη δομή *MonitorTable*.Ο monitorTable έχει αποθηκεύσει όλες τις τριπλέτες για τα διανύσματα που δεν αποθηκεύονται στη μνήμη.Με αυτό το τρόπο το σύστημα μπορεί να υπολογίζει κάθε διάνυσμα που δε βρίσκεται στη μνήμη με αναδρομικές κλήσεις της PLF.Η δομή θα έχει διαμορφωθεί ως εξής:

x1	x2	x3
4	13	14
11	14	15
16	9	17
10	17	18

Οι κλήσεις πραγματοποιούνται από το σύστημα μνήμης χωρίς να το αντιλαμβάνεται η φυλογενετική ανάλυση ή να επηρεάζεται με κάποιο τρόπο.Τα δεδομένα των διανυσμάτων που δεν κρατούνται στη μνήμη αποθηκεύονται προσωρινά σε μια μικρή επιπρόσθετη μνήμη του επιταχυντή.Τα δεδομένα ενός διανύσματος που βρίσκονται σε μια θέση μνήμης μπορούν να πανωγραφούν μόλις χρησιμοποιηθούν σε κάποιο υπολογισμό.Για παράδειγμα για τον υπολογισμό του κόμβου 14 θα δεσμευτεί μια θέση στη μνήμη του επιταχυντή.Τα δεδομένα αυτά χρησιμοποιούνται για τον υπολογισμό του διανύσματος πιθανότητας του κόμβου 15.Το διάνυσμα πιθανότητας του κόμβου 15 θα αποθηκεύσει τα δεδομένα του στην ίδια θέση μνήμης που δεσμεύτηκε από τον κόμβο 14 πανωγράφοντας τα δεδομένα του 14.



Σχήμα 5.10: Σε αυτό το σχήμα φαίνεται η τοπολογία του φυλογενετικού δέντρου του παραδείγματος της ενότητας 4.2.6 μετά την εφαρμογή του αλγορίθμου SPR.

Με την εφαρμογή SPR κίνησης, η οποία φαίνεται στην εικόνα 4.8, η ανάλυση πάραγει νέες κλήσεις για τον υπολογισμό της πιθανοφάνειας του δέντρου.Αρχικά παράγονται κάποιες PLF κλήσεις για τον υπολογισμό των κόμβων που επηρεάζονται από την SPR κίνηση.Ο κόμβος 11 διαγράφεται από το σημείο που βρισκόταν και δημιουργείται ένας νέος εσωτερικός κόμβος με τον ίδιο αναγνωριστικό αριθμό στη θέση που επανενώνεται το κλαδί.Ο νέος κόμβος υπολογίζεται από τη τριπλέτα 12,14,11 ενώ και ο κόμβος 15 υπολογίζεται από μια νέα τριπλέτα, την 1,11,15.

Η εφαρμογή του SPR αλγορίθμου δεν επηρεάζει τα διανύσματα πιθανότητας στους υπόλοιπους κόμβους του δέντρου πέρα από τους 11 και 15.Συνεπώς στη περίπτωση που όλοι οι κόμβοι αποθηκεύονται στη μνήμη αρκεί να υπολογίσουμε τις νέες τιμές των 11, 15 και τέλος τη πιθανοφάνεια της ρίζας.Στη περίπτωση όμως που εξετάζει το παράδειγμα,όπου υπάρχει συμπίεση δεδομένων, για να υπολογίσουμε το διάνυσμα πιθανότητας στη ρίζα του δέντρου χρειάζεται να υπολογιστούν επιπλέον PLF κλήσεις για τον επανυπολογισμό των κόμβων που δε βρίσκονται στη μνήμη.

Παρατηρούμε λοιπόν ότι αν όλα τα δεδομένα αποθηκεύονταν στη μνήμη, ο υπολογισμός της νέας τοπολογίας θα απαιτούσε 3 κλήσεις.Ωστόσο το νούμερο αυτό αυξάνεται από τη στιγμή που χρειάζεται να υπολογίσουμε κόμβους που δεν αποθηκεύονται στη μνήμη.Απαιτείται να υπολογίσουμε ξανά τον 14 για να μπορέσουμε να υπολογίσουμε την νέα τιμή του 11 αλλά και τους κόμβους 17 και 18.Συνεπώς έχουμε 3 επιπλέον PLF κλήσεις.Το σύστημα βγάζει από τη μνήμη μια σειρά κόμβους αλλά χρειάζεται να αυξήσει το πλήθος των κλήσεων για κάθε νέο υπολογισμό της φυλογενετικής πιθανοφάνειας του δέντρου.

Η υλοποίηση της PLF από κατάλληλη αρχιτεκτονική επιταχυντή συμβάλει ώστε για ένα συγκεκριμένο όριο συμπίεσης δεδομένων να μην υφίσταται επιπλέον κόστος στο χρόνο εκτέλεσης της ανάλυσης και το κόστος μετά από αυτό το όριο να παραμένει αναλογικά μικρό σε σχέση με τη συμπίεση.

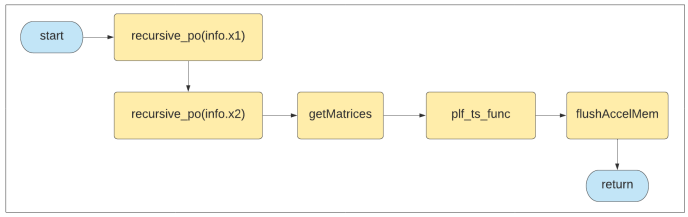
5.3 Επέκταση Συστήματος Διαχείρισης **PLF** Κλήσεων

Η επέκταση του συστήματος στηρίζεται στη δημιουργία νέας συνάρτησης για τους υπολογισμούς των διανυσμάτων πιθανότητας η οποία δέχεται 2^n διανύσματα στην είσοδο, όπου $n > 1$, και παράγει στην έξοδο το διάνυσμα πιθανότητας του κοινού προγόνου.Με αυτό το τρόπο μέσα σε μια κλήση ενσωματώνεται ο υπολογισμός περισσότερων του ενός διανυσμάτων πιθανότητας.Έχουμε επομένως τη δυνατότητα να μειώσουμε αρκετά το πλήθος των διανυσμάτων που βρίσκονται στη μνήμη καθώς μειώνεται δραστικά το πλήθος των επιπλέον αναδρομικών κλήσεων που πραγματοποιούνται εντός του συστήματος μνήμης και επομένως ο όποιος χρόνος επιβάρυνσης των επανυπολογισμών διανυσμάτων πιθανότητας.Η νέα συνάρτηση υλοποιείται από τον επιταχυντή δενδρικής δομής που παρουσιάστηκε στο 4^o κεφάλαιο.Στη παρούσα εργασία εξετάζουμε τη περίπτωση όπου $n = 2$ και συνεπώς έχουμε 4 διανύσματα πιθανότητας στην είσοδο της συνάρτησης.Αυτό που αλλάζει με τη προηγούμενη έκδοση του monitor είναι η συνάρτηση που πραγματοποιεί την post-order διάσχιση της δενδρικής δομής για τον υπολογισμό κάποιου διανύσματος.

5.3.1 Υπολογισμός Αναδρομικών Κλήσεων

Παρακάτω παρουσιάζεται η επέκταση του αλγορίθμου που πραγματοποιεί τις αναδρομικές κλήσεις από τη post-order διάσχιση του φυλογενετικού δέντρου.Η νέα έκδοση βασίζεται στη δυνατότητα με μια μοναδική κλήση να υπολογιστεί το διάνυσμα πιθανότητας που αντιστοιχεί στον κοινό πρόγονο 4 κόμβων.Οι κλήσεις που δέχεται το monitor παραμένουν ίδιες, δηλαδή διατηρούν τη μορφή της τριπλέτας {x1,x2,x3}.Η διαφορά είναι ότι σε ένα μοναδικό υπολογισμό μπορούν να ενσωματωθούν περισσότερες από μια PLF κλήσεις.Η νέα συνάρτηση που υπολογίζει τις κλήσεις του RAxML είναι η computePLF_po.Στην είσοδο λαμβάνει τη τριπλέτα x_info της υπολογιζόμενης κλήσης και επιστρέφει της διεύθυνση μνήμης xVector του διανύσματος-προγόνου.

Η *computePLF2* ορίζει τη μεταβλητή $accelInputInfo$, η οποία είναι τύπου $runtimeInfo\_tb\_t$ και αποθηκεύει τα δεδομένα εισόδου της συνάρτησης που εκτελεί τους PLF υπολογισμούς.Ακόμα ορίζει τους πίνακες accelMemAddr1 και accelMemAddr2, οι οποίοι δείχνουν σε block της accelMemBlock.Οι πίνακες x1Addr και x2Addr δείχνουν τις διευθύνσεις στη κύρια μνήμη που είναι αποθηκευμένα τα block που δείχνουν οι πίνακες accelMemAddr1 και accelMemAddr2 αντίστοιχα.Επιπλέον, η *computePLF2* χρησιμοποιεί τη συνάρτηση recursive_po για να λάβει τα δεδομένα των διανυσμάτων πιθανότητας x1,x2.Μια κλήση της recursive_po παίρνει στην είσοδο το id του κόμβου και μια σειρά από πίνακες και μεταβλητές στα οποία αποθηκεύει τα δεδομένα εξόδου.



Σχήμα 5.11: Η εικόνα παρουσιάζει το διάγραμμα ροής του αλγορίθμου computePLF2.

Algorithm 5 computePLF2

procedure COMPUTEPLF2

$inputs \leftarrow x\_info$
$output \leftarrow {}^*xVector$
$int\ accelMemAddrs1[2] \leftarrow \{0,0\},\ accelMemAddrs2[2] \leftarrow \{0,0\};$
$runtimeInfo\_tb\_t\ accelInputInfo;$
$double\ {}^*x1Addr[2],\ {}^*x2Addr[2];$

$recursive\_po(info.x1, x1Addr, accelInputInfo.left[0], accelInputInfo.right[0], accelInputInfo.ev[0],$ accelInputInfo.valid[0],accelMemAddrs1);

$accelInputInfo.xAddr[0] \leftarrow x1Addr[0];$
$accelInputInfo.xAddr[1] \leftarrow x1Addr[1];$

$recursive\_po(info.x2, x2Addr, accelInputInfo.left[1], accelInputInfo.right[1], accelInputInfo.ev[1],$ accelInputInfo.valid[1],accelMemAddrs2);

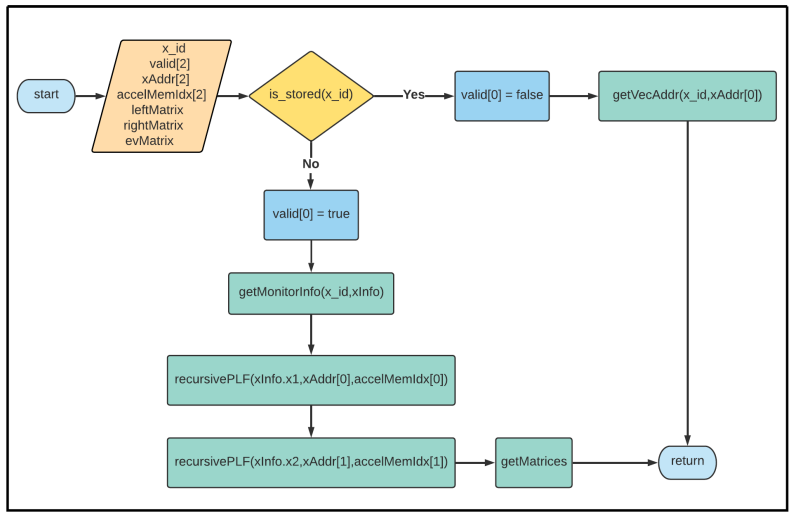
$accelInputInfo.xAddr[2] \leftarrow x2Addr[0];$
$accelInputInfo.xAddr[3] \leftarrow x2Addr[1];$
$getMatrices(info.x3, accelInputInfo.left[2], accelInputInfo.right[2], accelInputInfo.ev[2]);$

$plf\_accel\_ts\_unit(accelInputInfo);$
$flush\_accel\_mem(accelMemAddrs1, accelMemAddrs2)$

return 0

Η recursive_po αρχικά ελέγχει αν το διάνυσμα του κόμβου με αναγνωριστικό x_id βρίσκεται στη μνήμη.Αν η απάντηση είναι θετική ενημερώνει με τη τιμή false τη παράμετρο valid.Σε κάθε κόμβο-απόγονο αντιστοιχίζεται μια τιμή valid.Η τιμή της valid δείχνει αν χρειάζεται να εκτελεστεί PLF υπολογισμός για το αντίστοιχο διάνυσμα ή αν το διάνυσμα βρίσκεται στη μνήμη.Στη συνέχεια επιστρέφει στην xAddr[0] τη διεύθυνση του διανύσματος στη μνήμη.

Αν το διάνυσμα δε βρίσκεται στη μνήμη τότε η valid παίρνει την τιμή true.Η συνάρτηση λαμβάνει από το MonitorFile τη κλήση που υπολογίζει το διάνυσμα και καλεί τη συνάρτηση *recursiveCall* η οποία επιστρέφει τη διεύθυνση του διανύσματος.Επιπλέον, επιστρέφει τον αριθμό του block της accelMemBlock στο οποίο αποθηκεύτηκε το διάνυσμα που έχει υπολογιστεί.Η recursiveCall είναι ακριβώς όμοια με την αρχική έκδοση, με την μόνη διαφορά ότι η κλήση της computePLF αναφέρεται στην νέα έκδοση computePLF2 της συνάρτησης.Τέλος επιστρέφει τους κατάλληλους πίνακες left,right,ev.



Σχήμα 5.12: Η εικόνα παρουσιάζει το διάγραμμα ροής του αλγορίθμου recursive_po.

Algorithm 6 recursive_po

procedure RECURSIVE_PO

```
input ← nodeId
output ← xAddr[2], left, right, ev, valid, accelMemIdx

callInfo_t xInfo;

if is_stored(nodeId) then
    xAddr[0] ← getVecAddr(nodeId);
    valid ← false;
else
    valid ← true;
    getMonitorInfo(nodeId, xInfo);
    recursiveCall(xInfo.x1, xAddr[0], accelMemIdx[0]);
    recursiveCall(xInfo.x2, xAddr[1], accelMemIdx[1]);
    getMatrices(xInfo.x3, left, right, ev);
end if

return 0
```

5.3.2 Εφαρμογή Λειτουργίας Επέκτασης Συστήματος σε Φυλογενετικό Δέντρο

Χρησιμοποιώντας την επέκταση του συστήματος διαχείρισης των PLF κλήσεων,αναμένεται να μειωθεί το σύνολο των αναδρομικών κλήσεων προς τη μονάδα του επιταχυντή που απαιτούνται για τον υπολογισμό της πιθανοφάνειας του φυλογενετικού δέντρου.Στο παράδειγμα της ενότητας 5.2.4, με την εφαρμογή του νέου συστήματος, ο κόμβος 17 μπορεί πλέον να υπολογιστεί σε μια κλήση αντι για 2 που θα απαιτούσε στη βασική έκδοση του συστήματος.Το γεγονός ότι μπορούν να υπολογιστούν τρεις κλήσεις της PLF σε μια δε συνεπάγεται ότι θα υπολογίζονται υποτριπλάσιες κλήσεις από την ανάλυση.Ανάλογα με τη συμπίεση δεδομένων που εφαρμόζεται μπορεί να υπολογίζονται αρκετές κλήσεις που θα ενσωματώνουν μια ή δυο κλήσεις της PLF.Όσο αυξάνεται το πλήθος των taxa στα φύλλα και το βάθος του δέντρου και όσο αυξάνεται ο βαθμός συμπίεσης των δεδομένων τόσο η αναλογία μεταξύ των κλήσεων που εκτελεί το αρχικό σύστημα σε σχέση με την επέκταση του θα πλησιάζει το 3.Το κέρδος σε δυνατότητα συμπίεσης με την επέκταση του συστήματος είναι ικανοποιητική για μεγάλες αναλύσεις από τη στιγμή που κατασκευάζουμε επέκταση του αρχικού επιταχυντή, προσαρμοσμένο στην επέκταση του συστήματος διαχείρισης μνήμης.Με το νέο συνδυασμό συστήματος μνήμης και επιταχυντή ελαττώνεται ακόμα περισσότερο το χρονικό κόστος που επιβαρύνεται ο χρόνος εκτέλεσης της φυλογενετικής ανάλυσης και αυξάνεται περισσότερο το ποσοστό συμπίεσης που επιτυγχάνεται χωρίς αλλαγή στο χρόνο εκτέλεσης.

Κεφάλαιο 6

Υλοποίηση επιταχυντών σε αναδιατασσόμενη λογική(**FPGA**)

Στο 4^o κεφάλαιο παρουσιάστηκε αρχιτεκτονική για την υλοποίηση των υπολογισμών της PLF.Σε αυτό το κεφάλαιο παρουσιάζεται υλοποίηση της σχεδίασης με τη βοήθεια των εργαλείων της Xilinx πάνω στην FPGA της πλακέτας ZCU102.

6.1 Πλατφόρμα Υλοποίησης Αρχιτεκτονικής

Η πλατφόρμα υλοποίησης της αρχιτεκτονικής όπως εξηγήθηκε και στο κεφάλαιο 4 είναι καθοριστικός παράγοντας, καθώς από αυτή εξαρτώνται οι διαθέσιμοι πόροι.Σε αυτή την εργασία χρησιμοποιήθηκε η πλακέτα ZCU102. Η ZCU102 περιλαμβάνει ένα SoC PS (System on Chip Processing System), το οποίο δίνει τη δυνατότητα εκτέλεσης προγραμμάτων λογισμικού και μια μονάδα PL (Programmable Logic), η οποία δίνει τη δυνατότητα υλοποίησης αναδιατασσόμενης λογικής στη πλακέτα.Η FPGA της πλατφόρμας ZCU102 παρέχει ένα μεγάλο αριθμό από 274,080 LUTs, 2,520 DSP48s, 1,824 BRAMs, 548,160 FFs για να υποστηρίξουμε την αρχιτεκτονική που σχεδιάστηκε.Η εφαρμογή που πραγματοποιεί κλήσεις στον επιταχυντή εκτελείται πάνω στο σύστημα επεξεργασίας γενικού σκοπού της ZCU102 το οποίο αποτελείται από μια Cortex-A53 μονάδα επεξεργασίας εφαρμογών(application processing unit (APU)), 64-bit Arm v8 αρχιτεκτονικής με τέσσερις πυρήνες.Ακόμα η συγκεκριμένη πλατφόρμα παρέχει στον επεξεργαστή γενικού σκοπού κύρια μνήμη τύπου DDR4 και μεγέθους 4GB αλλά και 4 μονάδες DMA για τη γρήγορη μεταφορά δεδομένων από τη μνήμη προς τη μονάδα του επιταχυντή στην FPGA.Η συχνότητα ρολογιού που δουλεύει η μονάδα PS είναι 1199880126 Hz ενώ η συχνότητα στην FPGA μπορεί να κλιμακωθεί εως τα 600MHz.

6.2 Εργαλεία της **Xilinx**

6.2.1 **Vivado HLS**

Η αρχιτεκτονική του επιταχυντή υλοποιήθηκε χρησιμοποιώντας τις οδηγίες μεταγλωτισστή(compiler directives) που παρέχει το εργαλείο Vivado HLS.Το Vivado HLS παρέχει ένα σύνολο από ειδικές οδηγίες που μπορούν να εισαγθούν στον πηγαίο κώδικα και να καθοδηγήσουν το μεταγλωτιστή του συστήματος με σκοπό την υλοποίηση της επιθυμητής σχεδίασης.Κατάλληλη χρήση αυτών των οδηγιών μπορεί να δημιουργήσει μια καλή ισορροπία μεταξύ της απόδοσης, της ποσότητα των πόρων που χρησιμοποιούνται και τους χρονικούς περιορισμούς της συχνότητας ρολογιού.Οι οδηγίες του Vivado HLS προστίθεται απευθείας με κατάλληλο τρόπο μέσα στο C κώδικα που υλοποιεί την αρχιτεκτονική [1].Με την ολοκλήρωση της σύνθεσης του C κώδικα σε RTL block παράγεται μια αναφορά που μας δίνει πληροφορίες για την απόδοση της αρχιτεκτονικής και τους πόρους της αναδιατασσόμενης λογικής που καταλαμβάνει.Στη συνέχεια παρουσιάζονται τα βασικά σημεία κάθε αναφοράς για τον σχεδιαζόμενο επιταχυντή.

Timing: Πληροφορία για το καθορισμένο χρόνο κάθε κύκλου ρολογιού, τη χρονική απόκλιση και για την εκτίμιση του μικρότερου χρονικού διαστήματος που μπορεί να επιτευχθεί με βάση τη σχεδίαση.

Latency: Το πλήθος των κύκλων ρολογιού που χρειάζεται από την αρχή της εκτέλεσης μέχρι για να παραχθεί η έξοδος.

Initiation Interval: Ο αριθμός των κύκλων ρολογιού που απαιτείται για να μπορεί να δεχθεί καινούρια είσοδο ο επιταχυντής. Χωρίς κάποια βελτιστοποίηση με κάποια οδηγία προς το μεταγλωτιστή η παράμετρος του

Utilization Estimates:Παρουσίαση σε απόλυτα νούμερα αλλά και ποσοστιαία της χρησιμοποίησης των βασικών πόρων μιας αναδιατασσόμενης λογικής(BRAM,DSP48E,LUT,Flip-Flop).

6.2.2 **SDSoC**

Το SDSoC(Software-Defined System on Chip) παρέχει τη δυνατότητα ανάπτυξης ενσωματωμένων συστημάτων για τις πλατφόρμες **Zynq®-7000 All Programmable SoC** και **Zynq UltraScale+^{TM} MPSoC**.Η κύρια είσοδος είναι μια C/C++ εφαρμογή στην οποία μια ή περισσότερες συναρτήσεις μπορούν να καθοριστούν για υλοποίηση σε αναδιατασσόμενη λογική με χρήση των εργαλείων Vivado IDE και Vivado HLS. Αποτελείται από περιβάλλον ανάπτυξης που είναι βασισμένο στο ολοκληρωμένο περιβάλλον του Eclipse και ενσωματώνει μεταγλωτιστή συστήματος που μετατρέπει προγράμματα

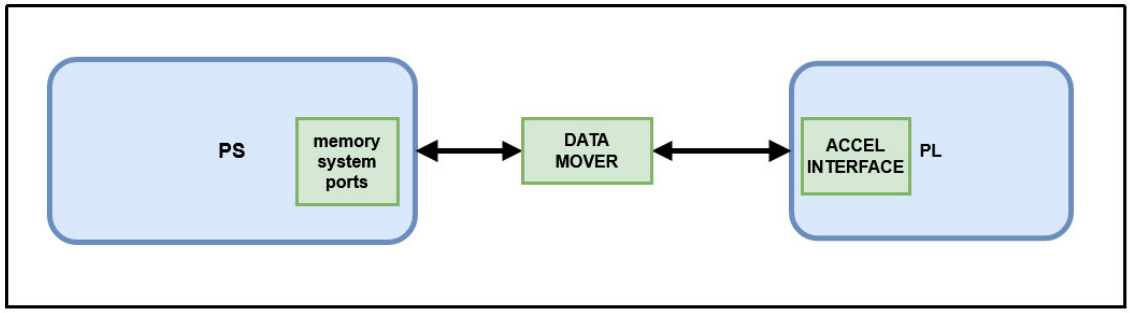
που έχουν κατασκευαστεί με χρήση των C-C++ σε πλήρη συστήματα υλικού/λογισμικού [2].

Ο μεταγλωτιστής συστήματος του SDSoC αναλύει το πηγαίο κώδικα του προγράμματος για να καθορίσει τη ροή δεδομένων μεταξύ των λειτουργιών που υλοποιούνται στον επεξεργαστή γενικού σκοπού και των λειτουργιών που υλοποιούνται στη προγραμματιζόμενη λογική.Στη συνέχεια υλοποιεί το πρόγραμμα παράγοντας ένα ενσωματομένο σύστημα σε chip (system-on-chip - SoC) το οποίο είναι προσαρμοσμένο στην εφαρμογή(application-specific).Ο μεταγλωτιστής του συστήματος παράγει μονάδες υλικού και λογισμικού που υλοποιούν το συγχρονισμό μεταξύ της εφαρμογής και του υλικού που βρίσκεται στην FPGA.Παράλληλα δίνουν τη δυνατότητα για υπολογισμούς και επικοινωνία με διασωλύνωση.

Η παραγωγή των επιταχυντών στην αναδιατασσόμενη λογική βασίζεται στο εργαλείο Vivado HLS, το οποίο καλείται αυτόματα από το SDSoC. Η σύνδεση των ξεχωριστών μονάδων του συστήματος γίνεται από το μεταγλωτιστή SDS++.Ο μεταγλωτιστής καλεί το εργαλείο Vivado Design Suite για να ενσωματώσει αυτόματα τους επιταχυντές σε μια ολοκληρωμένη σχεδίαση που περιλαμβάνει δομές επικοινωνίας και μεταφοράς δεδομένων μεταξύ του συστήματος επεξεργασίας που τρέχει την εφαρμογή και της αναδιατασσόμενης λογικής.Στο τελικό βήμα το SDSoC εκκινεί το εργαλείο Bootgen για να δημιουργήσει την bootable image που περιέχει την εκτελέσιμη εφαρμογή για τον επεξεργαστή γενικού σκοπού και το bitstream για την FPGA.

6.3 Υλοποίηση διεπαφής επιταχυντή

Η επικοινωνία του επιταχυντή με το PS που εκτελεί την εφαρμογή αποτελείται από 3 κύρια στοιχεία: τις θύρες του συστήματος μνήμης στο PS,τους μεταφορείς δεδομένων (data movers) μεταξύ επιταχυντή και PS και τη διεπαφή υλικού στον επιταχυντή.Το σχήμα 6.1 παρουσιάζει αυτήν την αρχιτεκτονική επικοινωνίας.



Σχήμα 6.1: Το δίκτυο μεταφοράς δεδομένων μιας πλατφόρμας μεταξύ τουPL και του PS αποτελείται από τρία βασικά στοιχεία: α) τις θύρες του συστήματος μνήμης στη πλευρά του PS, β)τους μεταφορείς δεδομένων μεταξύ του PS και των επιταχυντών και γ)τη διεπαφή του επιταχυντή.

Η διεπαφή του επιταχυντή για τη μεταφορά των δεδομένων ενός πίνακα μπορεί να είναι είτε τύπου **RAM interface** είτε τύπου **streaming interface**.Οι δύο τύποι έχουν διαφορές τόσο στο τρόπο που πραγματοποιείται η πρόσβαση στα δεδομένα όσο και στη μεταφορά των δεδομένων εντός του επιταχυντή.Με τη χρήση του RAM interface απαιτείται η μεταφορά όλων των δεδομένων του πίνακα εντός του επιταχυντή, τα οποία αποθηκεύονται σε BRAM μονάδες, πριν αυτά χρησιμοποιηθούν.Μετά τη μεταφορά του πίνακα στις BRAM του επιταχυντή μπορεί να πραγματοποιηθεί οποιαδήποτε τυχαία προσπέλαση στα δεδομένα.Οι μονάδες BRAM(Block RAM) είναι κομμάτια μνήμης RAM που ενσωματώνονται στη δομή της FPGA για την αποθήκευση μεγάλου όγκου δεδομένων εντός της FPGA.Το streaming interface δεν απαιτεί πόρους μνήμης εντός του επιταχυντή για την αποθήκευση όλων των δεδομένων του πίνακα.Δίνει τη δυνατότητα στον επιταχυντή να επεξεργάζεται δεδομένα του πίνακα και παράλληλα να διαβάζει νέα δεδομένα από τη μνήμη του PS.Η δυνατότητα αυτή επιτρέπει την επεξεργασία πινάκων με μεγάλο όγκο δεδομένων που δε μπορούν να αποθηκευτούν στην FPGA.Ο περιορισμός που επιβάλει το streaming interface είναι η σειρά με την οποία λαμβάνονται τα δεδομένα από τον επιταχυντή.Τα δεδομένα του πίνακα έρχονται στον επιταχυντή με αυστηρά ακολουθιακή σειρά.

Η εκτέλεση της συνάρτησης PLF χρειάζεται ως δεδομένα εισόδου τους πίνακες left,right,EV και τα δύο διανύσματα πιθανότητας εισόδου.Τα διανύσματα πιθανότητας αποτελούνται από ένα σύνολο N sites και κάθε site αποτελείται από 16 double τιμές.Το πλήθος N των sites είναι μεταβλητό και εξαρτάται από την εφαρμογή.Ωστόσο γνωρίζουμε ότι οι τιμές που μπορεί να λάβει, είναι πιθανό να είναι της τάξης του 10^6 ή και 10^7.Αυτό σημαίνει ότι το μέγεθος των δεδομένων είναι τέτοιο που δεν είναι δυνατό να χωρέσουν στις αποθηκευτικές μονάδες της FPGA.Γι΄ αυτό στη διεπαφή του επιταχυντή δημιουργούνται θύρες(ports) τύπου streaming interface για την ανάγνωση μέσω κατάλληλων ροών δεδομένων(data streams).Το SDSoC δίνει την οδηγία **SDS**

data access_pattern η οποία με τη παράμετρο **SEQUENTIAL** παρέχει αυτή τη δυνατότητα.Η οδηγία τοποθετείται ακριβώς πριν τη δήλωση της συνάρτησης και καθορίζει το τρόπο με τον οποίο θα λαμβάνονται από τη μνήμη τα δεδομένα εισόδου.Η απουσία της οδηγίας ή η εισαγωγή της οδηγίας με παράμετρο RANDOM έχει σαν αποτέλεσμα τα δεδομένα να λαμβάνονται μέσω ενός RAM interface.Αντίθετα η παράμετρος SEQUENTIAL καθορίζει ότι τα δεδομένα θα έρχονται μέσω ενός stream interface στη διεπαφή του επιταχυντή.

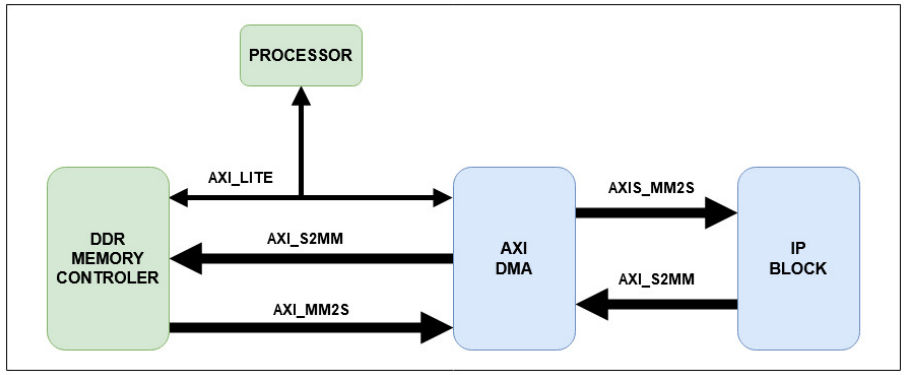
Ταυτόχρονα χρησιμοποιείται η οδηγία **SDS data copy** για να δηλωθεί ότι τα δεδομένα που λαμβάνονται από το streaming interface αντιγράφονται από τη μνήμη στον επιταχυντή.Επιπλέον με την οδηγία data copy δίνεται η δυνατότητα μέσω της παραμέτρου N να προσδιοριστεί το μέγεθος των διανυσμάτων πιθανότητας και οι πίνακες να περάσουν με τη μορφή δείκτη ως ορίσματα στη συνάρτηση που υλοποιείται στο PL.Οι οδηγίες access_patten και data copy είναι απαραίτητες για να υλοποιηθούν τα streaming ports για τη λήψη των δεδομένων από τα διανύσματα x1,x2,x3.

Οι πίνακες που χρησιμοποιούνται στους υπολογισμούς της PLF είναι σταθεροί για κάθε κλήση της PLF και μπορούν να αποθηκευτούν χωρίς πρόβλημα στον επιταχυντή.Για το λόγο αυτό χρησιμοποιείται RAM interface στη διεπαφή του επιταχυντή για τη λήψη των δεδομένων τους.

Κάθε μεταφορά δεδομένων μεταξύ ενός προγράμματος λογισμικού που εκτελείται στο PS και της συνάρτησης που υλοποιείται στο PL απαιτεί ένα data mover, ο οποίος αποτελείται από πόρους του υλικού που πραγματοποιούν τη μετακίνηση των δεδομένων και κάποιας συνάρτησης βιβλιοθήκης για τον έλεγχο.Όπως φάνηκε και πειραματικά η κατάλληλη επιλογή data mover παίζει σημαντικό ρόλο στη τελική απόδοση του συστήματος.Το περιβάλλον του SDSoC προσφέρει διάφορους τύπους από data movers ανάλογα με το είδος και το μέγεθος των δεδομένων που μεταφέρονται.

Κάθε επανάληψη στο κύριο loop του plf_accel_block υπολογίζει τις τιμές για ένα site του διανύσματος.Αυτό σημαίνει ότι απαιτείται να διαβαστούν 16 double τιμές από τις θύρες εισόδου των x1,x2 και να στέλνονται στη μνήμη 16 double τιμές από τη θύρα εξόδου του x3.Για να μπορούν οι θύρες του επιταχυντή να λαμβάνουν και να στέλνουν 16 double τιμές σε κάθε κύκλο ρολογιού θα πρέπει να έχουν bandwith ίσο με 1024 bits/cc.Αυτό δεν είναι εφικτό στη πλατφόρμα zcu102, καθώς οι AXI διεπαφές μεταξύ PS και PL έχουν μέγιστο πλάτος 128 bits.Με βάση αυτό το δεδομένο, οι τιμές κάθε site στέλνονται από τη μνήμη του PS στον επιταχυντή σε 8 πακέτα των 128 bits.Αυτό σημαίνει ότι για τον υπολογισμό ενός site του διανύσματος εξόδου,απαιτούνται 8cc για την ανάγνωση των 1024 bits που αντιστοιχίζονται στις 16 double τιμές κάθε site εισόδου.Τα δεδομένα πακετάρονται με τη βοήθεια των δομών(structs).Δημιουργείται η δομή **data128_t** η οποία αποτελείται από ένα πίνακα με 2 double τιμές.Τα ορίσματα της plf_accel_block που αναφέρονται στα δεδομένα των διανυσμάτων είναι δείκτες τύπου

data128_t.



Σχήμα 6.2: Η εικόνα παρουσιάζει το γενικό διάγραμμα που υλοποιείται με το πρωτόκολλο AXI DMA.Το διάγραμμα παρουσιάζει τη μεταφορά δεδομένων μεταξύ ενός επιταχυντή(IP block) και της μνήμης του PS.Ο processor και ο DDR memory controller βρίσκονται στο PS της πλατφόρμας, ενώ το IP block και το AXI DMA βρίσκονται στο PL.Ο processor συνδέεται με το AXI DMA μέσα από το δίαυλο AXI_LITE και παρακολουθεί τη μεταφορά δεδομένων.Οι δίαυλοι AXI_MM2S(Memory-Mapped to Streaming) και AXI_S2MM(Streaming to Memory-Mapped) που συνδέουν το AXI DMA με το memory controller οργανώνονται με το πρωτόκολλο AXI4 και παρέχουν στο DMA πρόσβαση στα δεδομένα της μνήμης.Οι αντίστοιχοι δίαυλοι που συνδέουν το IP block με το DMA είναι τύπου AXI4-streaming, οι στέλνουν και λαμβάνουν ένα συνεχές ρεύμα δεδομένων.

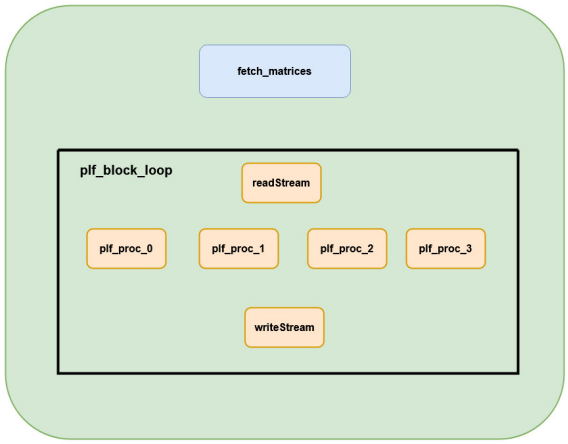
Το SDSoC παρέχει 3 διαφορετικούς τύπους data mover, οι οποίοι φαίνονται στο πίνακα της εικόνας 6.2.Ο ιδανικός data mover για τη μεταφορά των διανυσμάτων πιθανότητας το μέγεθος των οποίων δε ξεπερνάει τα 200.000 sites είναι ο axi_dma_simple. Παρότι απαιτεί περισσότερους πόρους υλικού για την υλοποίηση του σε σχέση με τους υπόλοιπους data movers, δίνει τη δυνατότητα γρήγορης μεταφοράς μεγάλης ποσότητας δεδομένων.Ο axi_dma_simple μπορεί να χρησιμοποιηθεί μονάχα για δεδομένα που βρίσκονται σε συνεχόμενες θέσεις στη μνήμη.Γι΄ αυτό απαιτεί από το πρόγραμμα που εκτελείται στο PS, να χρησιμοποιήσει για τη δέσμευση μνήμης τη συνάρτηση sds_alloc() που παρέχει το SDSoC. Οι άλλες επιλογές για data mover είναι ο axi_fifo και ο axi_dma_sg.Ο axi_fifo χρησιμοποιεί αισθητά λιγότερους πόρους αλλά ταυτόχρονα μεταφέρει αρκετά πιο αργά τα δεδομένα ενώ χρησιμοποιείται για μεταφορά δεδομένων έως 300 bytes.Από την άλλη μεριά ο axi_dma_sg, ο οποίος αποτελεί συνήθως τη προκαθορισμένη επιλογή του SDSoC, είναι πιο αργός από τον axi_dma_simple αλλά με λιγότερους περιορισμούς στην υλοποίηση του.Ο axi_dma_sg δεν περιορίζεται ούτε από το μέγεθος των δεδομένων που θα μεταφερθούν στον επιταχυντή ούτε από το τρόπο που αποθηκεύονται τα δεδομένα στη μνήμη του PS(π.χ συνεχόμενες θέσεις μνήμης).

Data Mover	Physical Memory Contiguity	Data Size (bytes)
AXI_DMA_SG	Either	> 300
AXI_DMA_Simple	Contiguous	< 32M
AXI_FIFO	Non-contiguous	< 300

Σχήμα 6.3: Ο πίνακας της εικόνας παρουσιάζει και συγκρίνει τους 3 τύπους data mover που παρέχει το SDSoC.

6.4 Υλοποίηση αρχιτεκτονικής του επιταχυντή

Η συνάρτηση που υλοποιείται από το SDSoC πάνω στο PL της ZCU102 είναι η plf_accel_block.Η συνάρτηση υλοποιεί 4 βασικές λειτουργίες.Αρχικά διαβάζει από τη μνήμη και αποθηκεύει σε καταχωρητές εντός του επιταχυντή όλα τα απαραίτητα δεδομένα για τους υπολογισμούς της PLF.Οι πίνακες left,right,EV από τη μνήμη του PS στον επιταχυντή με τη συνάρτηση fetch_matrices().Στη συνέχεια υλοποιεί ένα μεγάλο loop, το οποίο εκτελεί αριθμό επαναλήψεων ίσο με το πλήθος N των sites των διανυσμάτων πιθανότητας.Κάθε επανάληψη του loop υπολογίζει τις 16 πιθανοφάνειες που αντιστοιχίζονται σε ένα site.Κάθε επανάληψη αρχικά διαβάζει τα δεδομένα από τα διανύσματα πιθανότητας και ξεκινάει την επεξεργασία εφαρμόζοντας τους υπολογισμούς της PLF με κλήσεις της συνάρτησης plf_proc().Η συνάρτηση plf_proc εκτελεί τη βασική επεξεργασία, υλοποιώντας τα στάδια υπολογισμού της συνάρτησης PLF όπως παρουσιάστηκαν στο κεφάλαιο 4.Η plf_proc υλοποιεί το pipeline που υπολογίζει τις πιθανοφάνειες που αντιστοιχίζονται σε ένα site του διανύσματος πιθανότητας εξόδου,για ένα ρυθμό ετερογένειας.



Σχήμα 6.4: Σε αυτό το σχήμα φαίνεται το σύνολο των κλήσεων που πραγματοποιούνται στη συνάρτηση plf_accel_block που υλοποιεί τον επιταχυντή.Η συνάρτηση fetch_matrices φέρνει από τη μνήμη στον επιταχυντή τους απαραίτητους πίνακες για την εκτέλεση της PLF.Στη συνέχεια ακολουθεί το κύριο loop της συνάρτησης του επιταχυντή.Η κλήση readStream λαμβάνει από τη μνήμη τις τιμές των διανυσμάτων εισόδου x1,x2.Σε κάθε επανάληψη του loop λαμβάνει ένα site από κάθε διάνυσμα.Οι κλήσεις της plf_proc που ακολουθούν υπολογίζουν τις τιμές του διανύσματος εξόδου x3.Τέλος η κλήση writeStream στέλνει στη μνήμη του PS τις τιμές που υπολογίστηκαν.

Βασική απαίτηση από τον επιταχυντή που κατασκευάζουμε είναι να μπορεί επεξεργάζεται σε κάθε κλήση του μεγάλη ποσότητα δεδομένων από τα διανύσματα πιθανότητας.Επομένως ο χρόνος διεκπεραίωσης(throughput) γίνεται κυρίαρχος σε σχέση με το latency, στο συνολικό χρόνο που απαιτείται για τον υπολογισμό ενος διανύσματος πιθανότητας με μήκος πχ 50000 ή 100000 sites.Για αυτό το λόγο οι οδηγίες που προσθέτουμε στο κώδικα έχουν ως στόχο τη βελτιστοποίηση της παραμέτρου του initiation interval.

6.4.1 Οργάνωση δεδομένων στον επιταχυντή

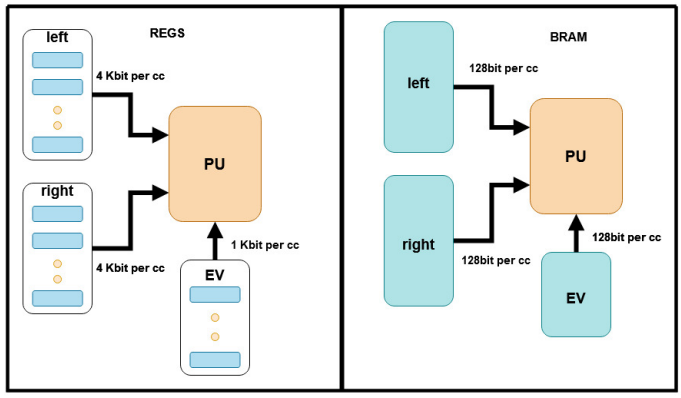
Ο τρόπος που αποθηκεύονται τα δεδομένα στον επιταχυντή μπορεί να αποτελέσει βασικό περιοριστικό παράγοντα στην απόδοση του επιταχυντή. Υπάρχουν δύο βασικοί τρόποι αποθήκευση δεδομένων στον επιταχυντή όπως παρουσιάζεται και στην εικόνα 6.5.Ο πρώτος τρόπος είναι με τη χρήση BRAM.Το πλεονέκτημα των BRAM είναι ότι δίνουν τη δυνατότητα αποθήκευσης μεγάλης ποσότητας δεδομένων στον επιταχυντή, ενώ το μειονέκτημα τους είναι ότι έχουν μικρό bandwith(ανάγνωση έως 2 δεδομένων ανά κύκλο ρολογιού).Ο δεύτερος τρόπος είναι η χρήση ενός αριθμού από μικρότερου μεγέθους BRAM ή ακόμα και συνόλων από καταχωρητές.Με αυτό το τρόπο επιτυγχάνεται μεγαλύτερο bandwith αλλά για την ίδια ποσότητα δεδομένων χρησιμοποιούνται περισσότεροι πόροι της FPGA.

Οι πίνακες left,right,ev αντιγράφονται από τη μνήμη του PS σε μονάδες BPAM εντός του επιταχυντή.Οι μονάδες BRAM δεν εξυπηρετούν τη σχεδίαση καθώς περιορίζουν τη

δυνατότητα πραγματοποίησης παράλληλων πράξεων.Οι BRAM δίνουν τη δυνατότητα ανάγνωσης έως δύο στοιχείων σε κάθε κύκλο ρολογιού.Ωστόσο κάθε στοιχείο ενός διανύσματος πιθανότητας που έρχεται από τη μνήμη πραγματοποιεί πράξεις με 4 στοιχεία του αντίστοιχου πίνακα υποκατάστασης.Το πρόβλημα αυτό λύνεται με τη βοήθεια της οδηγίας **ARRAY_PARTITION** του HLS.Η οδηγία αυτή δίνει τη δυνατότητα διάσπασης ενός ενιαίου πίνακα σε περισσότερους μικρότερου μεγέθους πίνακες που περιέχουν τα δεδομένα του αρχικού πίνακα.Το πλήθος των υποπινάκων στους οποίους διασπάται ο αρχικός πίνακας καθορίζεται από το όρισμα factor της οδηγίας.Επιπλέον, με το όρισμα variable καθορίζεται ο τρόπος με τον οποίο θα διαχωριστούν τα δεδομένα του αρχικού πίνακα.Υπάρχουν τρεις πιθανοί τρόποι διάσπασης των δεδομένων από μια BRAM.

- **block:** Ο αρχικός πίνακας διαιρείται σε ίσου μεγέθους blocks από συνεχόμενα στοιχεία του αρχικού πίνακα.
- **cyclic:** Ο πίνακας διασπάται με κυκλικό τρόπο σε n μικρότερους πίνακες.Αυτό σημαίνει ότι τοποθετείται διαδοχικά ενα στοιχείο του αρχικού πίνακα σε κάθε υποπίνακα πριν τοποθετηθεί το επόμενο.Η διαδικασία επαναλαμβάνεταται μέχρι όλα τα στοιχεία του αρχικού πίνακα να έχουν μοιραστεί στους n υποπίνακες.
- **complete:** Ο αρχικός πίνακας διασπάται σε ένα σύνολο καταχωρητών, το πληθος των οποίων είναι ίσο με το πλήθος των στοιχείων του πίνακα.

Η παράμετρος επιλέγεται ανάλογα με τον επιθυμητό βαθμό του παραλληλισμού και τους διαθέσιμους πόρους.Η zcu102 προσφέρει τους απαραίτητους πόρους για μέγιστο παραλληλισμό.Για να μεγιστοποιήσουμε τη δυνατότητα παράλληλων πράξεων χρησιμοποιούμε τη παράμετρο **complete**.Παράλληλα με τη χρήση του ARRAY_PARTITION με τη παράμετρο complete, συμπεριλαμβάνονται στον επιταχυντή 3 σύνολα καταχωρητών,όπου κάθε σύνολο αποτελείται από 16 καταχωρητές.Στα σύνολα αυτά αποθηκεύονται οι τιμές των διανυσμάτων πιθανότητας x1,x2 που λαμβάνονται από τη μνήμη και οι υπολογισμένες τιμές του x3.



Σχήμα 6.5: Σε αυτή την εικόνα παρουσιάζεται μια σύγκριση αρχιτεκτονικής που χρησιμοποιεί σύνολα καταχωρητών για την αποθήκευση δεδομένων στον επιταχυντή και μιας που χρησιμοποιεί μονάδες BRAM.Στη πρώτη περίπτωση που χρησιμοποιούνται σύνολα καταχωρητών το processing unit έχει πρόσβαση σε όλα τα δεδομένα σε ένα κύκλο ρολογιού.

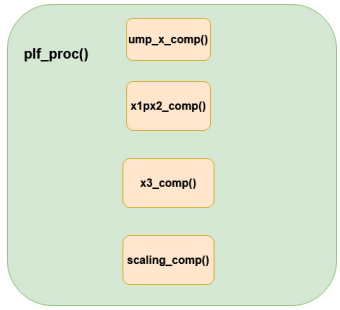
6.4.2 Υλοποίηση μονάδας επεξεργασίας του επιταχυντή

Οι υπολογισμοί της PLF στην plf_accel_block υλοποιούνται στο plf_block_loop.Κάθε επανάληψη στο plf_block_loop φέρνει από τη μνήμη τις 16 τιμές ενός site των διανυσμάτων εισόδου, πραγματοποιεί 4 κλήσεις της plf_proc για την εκτέλεση των PLF υπολογισμών και στέλνει στη μνήμη τις τιμές του διανύσματος εξόδου.Για κάθε στάδιο υπολογισμού της PLF, η plf_proc καλεί μια κατάλληλη υποσυνάρτηση για την υλοποίηση του όπως φαίνεται στο σχήμα 6.3.Οι υποσυναρτήσεις υλοποιούν τα διάφορα στάδια υπολογισμών της PLF όπως παρουσιάστηκαν στο κεφάλαιο 4.

Στο Vivado HLS, τα διαφορετικά επίπεδα συναρτήσεων δημιουργούν μια αντίστοιχη ιεραρχία και στην RTL μονάδα που προκύπτει.Στην αρχή κάθε υποσυνάρτησης της plf_proc προστίθεται η οδηγία **INLINE**.Η οδηγία αυτή καταργεί την ιεραρχία που δημιουργείται στην αρχιτεκτονική μεταξύ της βασικής συνάρτησης και των υποσυναρτήσεων που τη χρησιμοποιούν.Ταυτόχρονα το INLINE χρησιμοποιείται και στην plf_proc αλλά και στις συναρτήσεις readStream() και writeStream() που λαμβάνουν και στέλνουν δεδομένα από και προς τη μνήμη του PS.

Στην αρχή του plf_block_loop χρησιμοποιείται η οδηγία **PIPELINE**, η οποία παρέχεται από το Vivado HLS.Σε συνδιασμό με το γεγονός ότι οι συναρτήσεις που καλούνται μέσα στο loop χρησιμοποιούν την οδηγία INLINE, το PIPELINE έχει ως αποτέλεσμα οι υπολογισμοί της PLF να μην εκτελούνται σε ξεχωριστά υπολογιστικά blocks, με βάση τις διαφορετικές κλήσεις των συναρτήσεων, αλλά σε ένα ενιαίο block που περιλαμβάνει σύνολα καταχωρητών και pipelined αριθμητικές και λογικές υπολογιστικές μονάδες.Οι υπολογισμοί εκτελούνται με παράλληλο τρόπο, δηλαδή η αρχιτεκτονική εκτελεί τις πράξεις που δεν έχουν εξαρτήσεις μεταξύ τους ταυτόχρονα και όχι ακολουθιακά.Στην ουσία οι αρχιτεκτονικές για τα διαφορετικά στάδια υπολογισμού, όπως

παρουσιάστηκαν στο κεφάλαιο 4, ενοποιούνται σε ένα ενιαίο pipelined block.Με αυτό το τρόπο υλοποιείται fain-grain παραλληλισμός, δηλαδή παραλληλισμός στο επίπεδο των αριθμητικών και λογικών πράξεων.Κάθε πράξη εκτελείται μόλις τα δεδομένα που χρειάζεται είναι έτοιμα, χωρίς να αναμένει την ολοκλήρωση της εκτέλεσης των συναρτήσεων που προηγούνται.Για παράδειγμα, δε χρειάζεται να ολοκληρωθεί η λήψη όλων των δεδομένων από τα διανύσματα εισόδου για να ξεκινήσουν οι PLF υπολογισμοί.Αντίστοιχα κατά τη διάρκεια των PLF υπολογισμών, οι πράξεις για παράδειγμα της x1px2_comp() μπορούν να ξεκινήσουν πριν την ολοκλήρωση της ump_x_comp().



Σχήμα 6.6: Σε αυτό το σχήμα φαίνεται το σύνολο των υποσυναρτήσεων της plf_proc().

Το PIPELINE εισάγεται στο κώδικα μαζί με τη παράμετρο II(Initiation Interval).Το Initiation Interval καθορίζει τον αριθμό των κύκλων ρολογιού που απαιτείται να παρέλθουν για την επεξεργασία νέων δεδομένων από το pipeline.Η τιμή που μπορεί να πάρει η παράμετρος II εξαρτάται από το πλήθος των υπολογισμών που μπορούν να εκτλεστούν παράλληλα και από τις δυνατότητες μεταφοράς δεδομένων από τη μνήμη του PS στον επιταχυντή.Το πλήθος των παράλληλων υπολογισμών εξαρτάται από τους πόρους της FPGA που υλοποιεί την αρχιτεκτονική.Η τιμή του II παίρνει υπόψη τόσο το πλήθος των διαθέσιμων υπολογιστικών μονάδων της FPGA, όσο και τη δυνατότητα που παρέχει για αποθήκευση των δεδομένων του επιταχυντή με τέτοιο τρόπο ώστε να παρέχουν το μέγιστο δυνατό παραλληλισμό.Η ZCU102 προσφέρει τους κατάλληλους πόρους ώστε οι τιμές των left,right,EV όπως και των διανυσμάτων πιθανότητας να αποθηκευτούν σε σύνολα καταχωρητών και όχι σε BRAM αλλά και να σχηματιστεί ο απαραίτητος αριθμός από υπολογιστικές μονάδες που θα εκτελούν τους παράλληλους υπολογισμούς.Ωστόσο η τιμή του II εξαρτάται και από τις δυνατότητες της πλατφόρμας στη μεταφορά δεδομένων από και προς τον επιταχυντή.Τα δεδομένα στον επιταχυντή μεταφέρονται σε πακέτα των 128 bits και οι θύρες των διανυσμάτων πιθανότητας στη διεπαφή του επιταχυντή χρειάζονται 8 κύκλους ρολογιού για να λάβουν ή να μεταφέρουν τα 1024 bits που αντιστοιχούν στις 16 double τιμές.Συνεπώς, στο plf_loop_block,η παράμετρος II του PIPELINE έχει κάτω όριο τη τιμή 8. Ταυτόχρονα το SDSoC δίνει

τη δυνατότητα κλιμάκωση της συχνότητας λειτουργίας της αρχιτεκτονικής, το οποίο συμβάλει στη μεγαλύτερη βελτίωση της απόδοσης.Ο πίνακας που παρουσιάζεται στην εικόνα 6.7 δείχνει του πόρους που καταναλώνει ο επιταχυντής για συχνότητα λειτουργίας 100MHz.Το latency του plf_loop_block pipeline για τη συγκεκριμένη υλοποίηση είναι ίσο με 52 cc.Η χρήση πόρων που παρουσιάζεται στο SDSoC διαφέρει από αυτή που παρουσιάζεται στο Vivado HLS.Αυτό συμβαίνει γιατί το HLS παρουσιάζει μόνο τους πόρους που απαιτούνται για την υλοποίηση της συνάρτησης του επιταχυντή, ενώ το SDSoC περιλαμβάνει και τους πόρους που απαιτούνται για την επικοινωνία του επιταχυντή με το PS.

Resource	Used	Total	% Utilization
DSP	362	2520	14,37
BRAM	48	912	5,26
LUT	54804	274080	20
FF	85129	548160	15,53

Σχήμα 6.7: Ο πίνακας της εικόνας,τον όποιο παρέχει το SDSoC, παρουσιάζει τη χρήση πόρων της ZCU102 από τον επιταχυντή της συνάρτησης PLF με συχνότητα λειτουργίας 100MHz.

Η κλιμάκωση της συχνότητας για την αύξηση της απόδοσης έχει σαν αποτέλεσμα τη δημιουργία βαθύτερου pipeline με αντίστοιχη αύξηση πόρων.Ωστόσο η κλιμάκωση της συχνότητας για την αρχιτεκτονική του επιταχυντή της συνάρτησης PLF δεν γίνεται να φτάσει το όριο των 600MHz υπάρχει περιορισμός από τους διαθέσιμους πόρους της ZCU102.

6.4.3 Υλοποίηση επιταχυντή με πολλαπλούς πυρήνες

Η πλατφόρμα zcu102 παρέχει αρκετούς πόρους, οι οποίοι δίνουν τη δυνατότητα υλοποίησης επιταχυντή με περισσότερους του ενός πυρήνες.Το SDSoC παρέχει την οδηγία **SDS resource(ID)**, η οποία εισάγεται στο κώδικα ακριβώς πριν από μια κλήση της συνάρτησης του επιταχυντή.Η οδηγία αυτή συνδέει τη κλήση της συνάρτησης υλικού με μια συγκεκριμένη υλοποίηση(instance) της, στην οποία αποδίδεται ως αναγνωριστικό στοιχείο ο ακέραιος αριθμός ID.Για διαφορετικές τιμές του αναγνωριστικού ID δημιουργούνται διαφορετικές υλοποιήσεις του επιταχυντή στο PL.Με αυτό το τρόπο το SDSoC δίνει τη δυνατότητα να κατασκευαστούν πάνω στην FPGA πολλαπλά αντίγραφα του επιταχυντή που μπορούν να εκτελούν παράλληλα διαφορετικές κλήσεις.Με αυτό το τρόπο υλοποιούνται επιταχυντές με 2 και 4 πυρήνες.Θεωρητικά κάθε υλοποίηση του επιταχυντή δεν μοιράζεται DMA.Ωστόσο η zcu102 διαθέτει 4 DMA μονάδες.Αυτό σημαίνει ότι στη πράξη υπάρχει διαμοιρασμός των συνδέσεων μεταξύ των διαφορετικών πυρήνων του επιταχυντή, παρότι θεωρητικά κάθε πυρήνας παράγει τη δική του διασύνδεση με

τη μνήμη του PS.Στα διαγράμματα που ακολουθούν παρουσιάζονται οι αναφορές του SDSoC.

Resource	Used	Total	% Utilization
DSP	724	2520	28,73
BRAM	96	912	10,53
LUT	109608	274080	39,99
FF	170258	548160	31,06

Σχήμα 6.8: Ο πίνακας δείχνει τους πόρους που χρησιμοποιεί επιταχυντής με δύο πυρήνες σε συχνότητα λειτουργίας 100MHz.

Resource	Used	Total	% Utilization
DSP	1448	2520	57,46
BRAM	192	912	21,05
LUT	219216	274080	79,98
FF	340516	548160	62,12

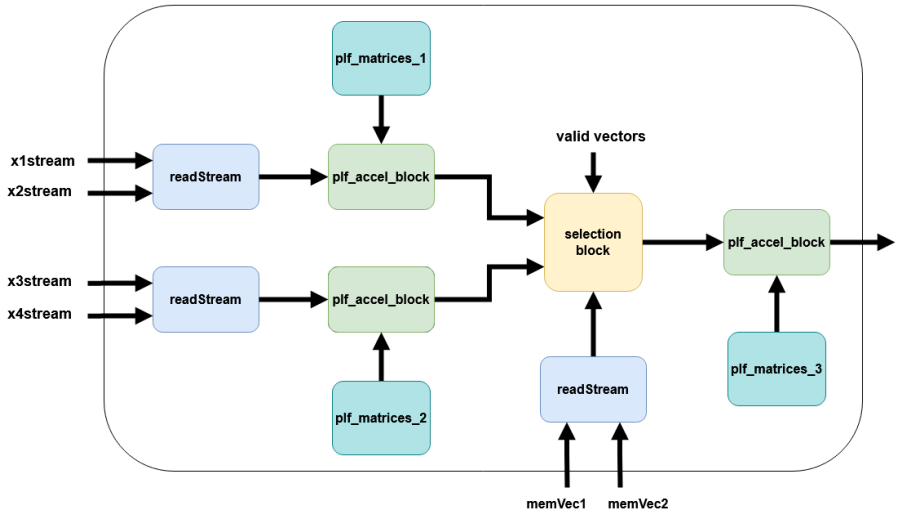
Σχήμα 6.9: Ο πίνακας δείχνει τους πόρους που χρησιμοποιεί επιταχυντής με τέσσερις πυρήνες σε συχνότητα λειτουργίας 100MHz.

6.4.4 Υλοποίηση της μονάδας **plf_accel_ts_block**

Ο επιταχυντής plf_accel_ts_block υλοποιεί την αρχιτεκτονική του plf_accel_ts_unit που παρουσιάστηκε στο κεφάλαιο 4.Το plf_accel_ts_unit υπολογίζει τη πιο απλή περίπτωση post-order διάσχισης.Με βάση τους πόρους της zcu102 επιλέχθηκε ότι η παράμετρος N του plf_accel_ts_block θα πάρει τη τιμή N=4.Αυτό σημαίνει ότι στην έξοδο δίνεται το διάνυσμα πιθανότητας του κόμβου-προγόνου από τους 4 πλησιέστερους κόμβους-απογόνους του.Στην εικόνα 6.10 φαίνονται οι πόροι που καταναλώνει η υλοποίηση για συχνότητα λειτουργίας του επιταχυντή και του data motion network στα 100MHz, ενώ η εικόνα 6.11 παρουσιάζει το διάγραμμα του επιταχυντή που υλοποιείται με τη βοήθεια του SDSoC.Το διάνυσμα valid vector αποτελείται από δύο τιμές.Κάθε τιμή δείχνει αν τα δεδομένα προς το plf_accel_block που παράγει τα δεδομένα εξόδου θα ληφθούν από τη μνήμη ή από τις plf_accel_block που προηγούνται.Αυτές οι plf_accel_block υπολογίζουν τα διανύσματα σε περίπτωση που αυτά δε βρίσκονται στη μνήμη.Η συνάρτηση υλικού προσθέτει τη συνάρτηση selection_block, η οποία με βάση τις τιμές του διανύσματος valid vector κάνει τη κατάλληλη επιλογή.

Resource	Used	Total	% Utilization
DSP	1086	2520	43,1
BRAM	78	912	8,55
LUT	126955	274080	46,32
FF	218002	548160	39,77

Σχήμα 6.10: Ο πίνακας δείχνει τους πόρους που καταναλώνει η μονάδα plf_accel_ts_block με συχνότητα λειτουργίας 100MHz.



Σχήμα 6.11: Το διάγραμμα παρουσιάζει την υλοποίηση του plf_accel_ts_block με τη βοήθεια του SDSoC.Το block υπολογίζει το κόμβο-πρόγονο από 4 κοντινότερους απογόνους του.

Κεφάλαιο 7

Αξιολόγηση απόδοσης επιταχυντών και εκτίμηση προτεινόμενου συστήματος

Στο παρόν κεφάλαιο παρουσιάζονται τα αποτελέσματα των πειραμάτων που πραγματοποιήθηκαν με τη χρήση των επιταχυντών που υλοποιήθηκαν στη πλατφόρμα zcu102 και του προτεινόμενου συστήματος διαχείρισης των PLF κλήσεων. Τα αποτελέσματα περιλαμβάνουν την αξιολόγηση της απόδοσης για τις αρχιτεκτονικές των επιταχυντών και την αξιολόγηση του συνολικού συστήματος με βάση το χρόνο εκτέλεσης μιας φυλογενετικής ανάλυσης καθώς αυξάνεται ο βαθμός συμπίεσης των δεδομένων.

7.1 Παράμετροι πειραμάτων αξιολόγησης

Για να υπολογιστεί η απόδοση των επιταχυντών και να αξιολογήσουμε τη συμπίεση μνήμης που επιτυγχάνεται με τη χρήση του συστήματος monitor και των επιταχυντών για τις εκτελέσεις της φυλογενετικής συνάρτησης, παράγουμε κατάλληλα σύνολα δεδομένων (datasets).Τα σύνολα αυτά προσομοιώνουν πολλαπλές ευθυγραμμισμένες ακολουθίες DNA που αντιστοιχούν σε ένα πλήθος από οργανισμούς(taxa).Ο παρακάτω πίνακας παρουσιάζει τις βασικές παραμέτρους που καθορίζουν τη κατασκευή των datasets που δίνονται ως είσοδοι στα πειράματα αξιολόγησης.

Evaluation Parameters

Sites	100	1.000	10.000	50.000	100.000	200.000				
tip nodes	100	1.000								
compress factor	10%	20%	30%	40%	50%	60%	70%	80%	90%	$\log_2(n)+2$

Η απόδοση των επιταχυντών υπολογίζεται για την εκτέλεση μιας κλήσης καθώς μεταβάλλεται το πλήθος των sites στα διανύσματα εισόδου/εξόδου.Η παράμετρος που κα-

θορίζει την απόδοση κάθε επιταχυντή είναι η διεκπεραιωτική ικανότητα καθώς μεταβάλλεται το μέγεθος των ακολουθιών.Το πλήθος των sites στα διανύσματα εισόδου/εξόδου παίρνει τις τιμές 100,1.000,10.000,50.000, 100.000 και 200.000.Τα παραπάνω νούμερα επιλέγονται γιατί είναι ενδεικτικά της αυξητικής πορείας που ακολουθεί το throughput των επιταχυντών καθώς η επιβάρυνση από το latency του pipeline γίνεται αμελητέα ποσότητα, αλλά και της σταθεροποίησης που ακολουθεί.

Στη συνέχεια χρειάζεται να αξιολογήσουμε το συνολικό σύστημα που διαχειρίζεται τις PLF κλήσεις μιας φυλογενετικής ανάλυσης.Αυτό αποτελείται από το λογισμικό του monitor και τους επιταχυντές που εκτελούν τις PLF κλήσεις.Η προσομοίωση μιας φυλογενετικής ανάλυσης μπορεί να πραγματοποιηθεί με την εξαγωγή των PLF κλήσεων που την υλοποιούν από το RAxML. Για το σκοπό αυτό δημιουργούμε σύνολα δεδομένων(datasets) που αποτελούνται από μεγάλο αριθμό taxa και μικρό μήκος ακολουθίας για κάθε οργανισμό.Χρησιμοποιούμε αυτά τα datasets ως είσοδο για την εκτέλεση φυλογενετικών αναλύσεων από το RAxML.Από τις αναλύσεις που εκτελούνται αποθηκεύονται τα σύνολα των PLF κλήσεων που παράγει το RAxML και στη συνέχεια δίνονται ως είσοδος στο monitor.Κάθε σύνολο PLF κλήσεων αντιστοιχίζεται στη κατασκευή ενός φυλογενετικού δέντρου και στην εφαρμογή σε αυτό του αλγορίθμου SPR.

Η αξιολόγηση του συστήματος γίνεται με μεταβολή της παραμέτρου $compress factor$.Η compress factor δίνει το ποσοστό των κόμβων που βγαίνουν εκτός μνήμη,δηλαδή τη συμπίεση που επιτυγχάνεται.Η σχέση $\log_2(n) + 2$ δίνει την ελάχιστη ποσότητα δεδομένων που απαιτείται να βρίσκονται στη μνήμη από τους εσωτερικούς κόμβους του φυλογενετικού δέντρου για την υλοποίηση των υπολογισμών της φυλογενετικής ανάλυσης [23].Το n δίνει το πλήθος των taxa στο dataset εισόδου.

Καθώς αυξάνεται το ποσοστό συμπίεσης και ελαττώνονται οι κόμβοι που βρίσκονται στη μνήμη μετριέται πως επηρεάζεται το πλήθος των PLF κλήσεων που παράγονται από το monitor για τις δυο περιπτώσεις υπολογισμού.Στη πρώτη περίπτωση οι υπολογισμοί εκτελούνται από επιταχυντή που σε κάθε κλήση του εκτελεί μια κλήση της PLF. Στη δεύτερη περίπτωση το σύστημα διαχείρισης μνήμης πραγματοποιεί κλήσεις προς τον επιταχυντή plf_accel_ts_unit που μπορεί σε μια κλήση του να εκτελέσει έως τρεις κλήσεις της PLF. Η ποσότητα των κλήσεων που παράγονται σε κάθε ενδεχόμενο από το monitor σε συνδυασμό με την απόδοση των επιταχυντών καθορίζουν τη συνολική απόδοση, η οποία μετριέται με το χρόνο ολοκλήρωσης μιας φυλογενετικής ανάλυσης.

Τα αποτελέσματα από τις διαφορετικές κλήσεις των επιταχυντών συγκρίνονται με την απόδοση διεκπεραίωσης των εκτελέσεων του προγράμματος RAxML, όταν αυτό εκτελείται σε πολυπύρηνο σύστημα επεξεργασίας.Τα πειράματα περιλαμβάνουν τη εκτέλεση της ακολουθιακής έκδοσης του RAxML και των εκδόσεων που χρησιμοποιούν AVX,AVX2 και Pthreads αντίστοιχα.Το σύστημα επεξεργασίας αποτελεί βάση σύγκρισης καθώς διαθέτει τους απαραίτητους πόρους για μια εκτέλεση φυλογενετικής

ανάλυσης μεγάλης έκτασης.

7.2 Πλατφόρμες πειραμάτων

Στην ενότητα αυτή παρουσιάζονται τα χαρακτηριστικά από το σύνολο των συσκευών που χρησιμοποιήθηκαν στην εργασία.

7.2.1 Πλατφόρμα **Zedboard**

Η πλατφόρμα zedboard χρησιμοποιήθηκε για την επικύρωση της σωστής λειτουργίας της βασικής μονάδας του επιταχυντή plf_accel_unit που υλοποιεί όλες τις απαραίτητες πράξεις για τον υπολογισμό ενός διανύσματος πιθανότητας και χρησιμοποιείται σαν βάση για όλες τις υπόλοιπες μονάδες.Η FPGA δίνει ακριβώς τα ίδια αποτελέσματα με την υλοποίηση του software.

Ωστόσο οι δυνατότητες του zedboard που είχαμε στη διάθεση μας, το οποίο παρέχει 53,280 LUTs, 220 DSP48′s, 280 BRAMs, 106,400 FF′s, δεν επαρκούν για την υλοποίηση αρχιτεκτονικής με αποδεκτή απόδοση.Για το λόγο αυτό, για την αξιολόγηση της απόδοσης χρησιμοποιήθηκε η πλατφόρμα ZCU102.

7.2.2 Πλατφόρμα **ZCU102**

Οι πόροι που παρέχει η ZCU102 παρουσιάστηκαν στο 6^o κεφάλαιο.Με τη βοήθεια της ZCU102 η εργασία αξιολογεί την απόδοση των διαφορετικών μονάδων των επιταχυντών, και του συστήματος που διαχειρίζεται τις κλήσεις των δεδομένων που δε βρίσκονται στη μνήμη.

Προεπιλεγμένα το περιβάλλον του SDSoC παράγει κατάλληλο αρχείο SD card image για την υλοποίηση της σχεδίασης πάνω σε πραγματική πλατφόρμα με αναδιατασσόμενη λογική.Με αυτό το αρχείο φορτώνουμε τη σχεδίαση στη ZCU102 μέσα από τη διεπαφή που παρέχει για κάρτα μνήμης SD.

Επιπλέον το περιβάλλον SDSoC παρέχει κατάλληλο API μέσα στη βιβλιοθήκη sds_lib για την μέτρηση της απόδοσης εφαρμογών.Με τη χρήση του API μπορούμε να καθορίσουμε χρονικές στιγμές και να υπολογίσουμε το χρονικό διάστημα που μεσολαβεί μέσα στο οποίο πραγματοποιείται η εκτέλεση του επιταχυντή.Οι μετρήσεις παρέχουν μεγάλη ακρίβεια καθώς δίνεται η δυνατότητα μέτρησης των κύκλων ρολογιού που χρειάστηκε η πλατφόρμα για να ολοκληρώσει μια εκτέλεση, καθώς και της συχνότητας λειτουργίας.Έτσι υπολογίζουμε εύκολα το χρόνο εκτέλεσης μέσα από τη σχέση:

$$time = \frac{clock\ cycles}{frequency} \tag{7.1}$$

7.2.3 Πολυπύρηνο Υπολογιστικό Σύστημα

Η πλατφόρμα που αξιολογούμε τις εκτελέσεις του RAxML είναι ένας server με 2 CPU's τύπου Intel(R) Xeon(R) CPU E5-2630v4 .Κάθε CPU αποτελείται από 10 cores και μπορεί να εκτελεί παράλληλα 20 threads ενώ συνοδεύεται από 25MB Cache.Οι CPU's είναι χρονισμένοι στα 2.20GHz με μέγιστη συχνότητα λειτουργίας στα 3.1GHz.Τέλος ο server παρέχει κύρια μνήμη 128GB DDR4 RAM.Μπορεί να υποστηρίξει την Pthread έκδοση του RAxML με παράλληλη λειτουργία 40 νημάτων.Ακόμα υποστηρίζει τη μικροαρχιτεκτονική AVX που επεκτείνει την x86 αρχιτεκτονική συνόλου εντολών.

Το RAxML εκτελέστηκε για τις διάφορες εκδόσεις του πάνω στο παραπάνω πολυπύρηνο επεξεργαστικό σύστημα.Η απόδοση αυτού του συστήματος αποτελεί μέτρο σύγκρισης για τους επιταχυντές.Για να μετρήσουμε της απόδοση του συστήματος για την έκδοση του RAxML με Pthreads, πήραμε μετρήσεις για 2,4,8,16 και 20 νήματα του RAxML τα οποία αντιστοιχίζονται στους πυρήνες του συστήματος.Η απόδοση του συστήματος επεξεργασίας όταν χρησιμοποιεί περισσότερους από 16 πυρήνες μειώνεται.

7.3 Αξιολόγηση Απόδοσης Επιταχυντών

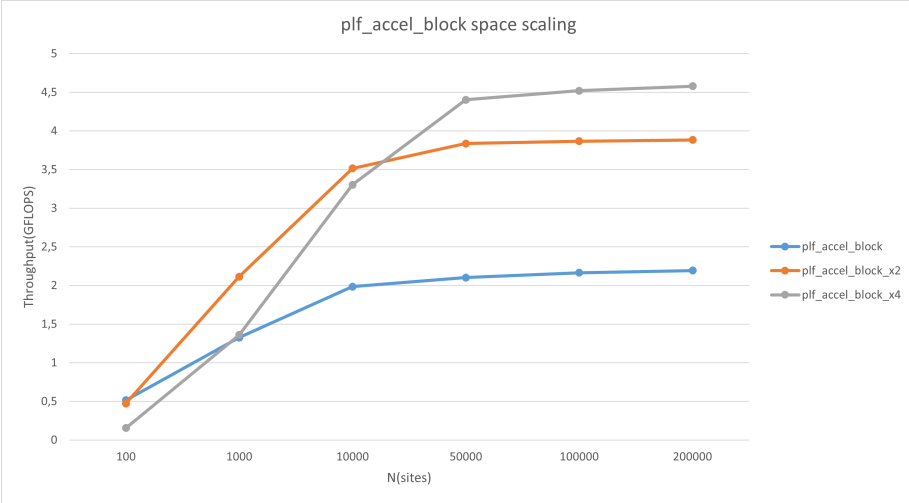
Στην ενότητα αυτή παρουσιάζουμε τα αποτελέσματα που δίνουν οι μετρήσεις των επιταχυντών στη ZCU102.Η απόδοση μετριέται με βάση τη διεκπεραιωτική ικανότητα του επιταχυντή, δηλαδή τη ποσότητα των διανυσμάτων των sites(site vectors) που παράγονται στη μονάδα του χρόνου.Ο χρόνος διεκπεραίωσης είναι ο μέγιστος ρυθμός παραγωγής αποτελεσμάτων που μπορεί να πετύχει η αρχιτεκτονική και υπολογίζεται από τη σχέση:

$$Throughput = \frac{N * FLOPV}{time}\ , \tag{7.2}$$

όπου N είναι το πλήθος των διανυσμάτων εισόδου, FLOPV(Floating-point operations per vector) είναι η ποσότητα των πράξεων διπλής ακρίβειας που απαιτούνται για τον υπολογισμό ενός διανύσματος θέσης(site vector) των 16 double τιμών, ενώ time είναι ο συνολικός χρόνος που απαιτείται για να ολοκληρωθούν οι υπολογισμοί της συνάρτησης.Ο υπολογισμός κάθε site vector απαιτεί να πραγματοποιηθούν 400 floating-point πράξεις.Με βάση τη σχέση 7.2, το *Throughput* υπολογίζεται με βάση τη ποσότητα των πράξεων διπλής ακρίβειας στη μονάδα του χρόνου(FLOPS(floating-point operations per sec)).Το *Throughput* μετριέται στη τάξη των *GFLOPS* $\left(10^9\ FLOPS\right)$.

7.3.1 Επεκτάσεις επιταχυντή με χρήση περισσότερων πυρήνων

Η αρχιτεκτονική που υλοποιήθηκε για μια απλή κλήση της PLF,όπως παρουσιάστηκε με στο κεφάλαιο 4 με την μονάδα plf_accel_unit, καταλαμβάνει μονάχα ένα ποσοστό των πόρων από την ZCU102. Το γεγονός αυτό δίνει τη δυνατότητα να κλιμακώσουμε την αρχιτεκτονική του επιταχυντή στους πόρους της συσκευής με περισσότερους πυρήνες της αρχικής υλοποίησης.Οι πυρήνες αυτοί εργάζονται ταυτόχρονα για τον υπολογισμό διαφορετικών sites της αλληλουχίας.Το διάγραμμα της εικόνας 7.1 συγκρίνει την απόδοση των διαφορετικών επεκτάσεων του επιταχυντή για διαφορετικό πλήθος πυρήνων που υλοποιούν τη μονάδα plf_accel_unit.

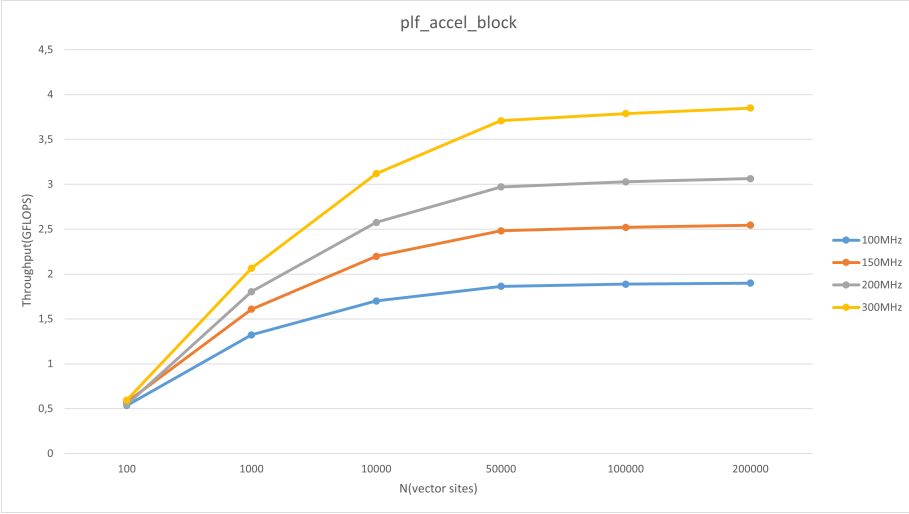


Σχήμα 7.1: Το διάγραμμα του σχήματος συγκρίνει την απόδοση για τον υπολογισμό μιας PLF κλήσης καθώς μεταβάλεται το πλήθος των sites όταν χρησιμοποιούνται ένας,δύο ή τέσσερις πυρήνες του επιταχυντή με συχνότητα λειτουργίας 100MHz.Στον άξονα x βρίσκεται το πλήθος των sites N και στον y το $Throughput(GFLOPS)$ των επιταχυντών.Η μπλε γραμμή δείχνει το $Throughput$ για ένα πυρήνα,η κόκκινη για 2 και η γκρι για 4.Παρατηρείται ότι για τάξη 10^3 sites το latency του pipeline θεωρείται αμελητέο.

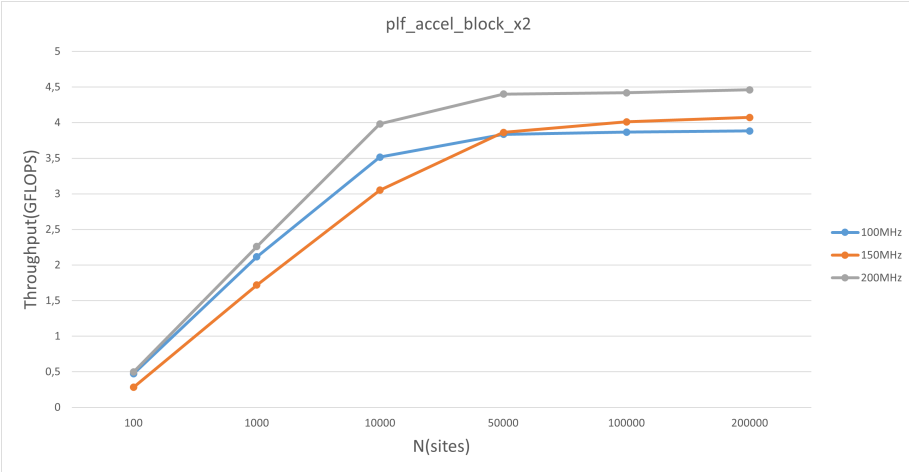
Στο διάγραμμα 7.1 συγκρίνεται η απόδοση των επιταχυντών καθώς μεταβάλλεται το πλήθος των sites με χρήση ενός,δύο ή τεσσάρων όμοιων πυρήνων πάνω στη συσκευή.Η χρήση περισσότερων πυρήνων δεν ήταν δυνατή στη ZCU102 λόγω του περιορισμού των διαθέσιμων πόρων.Παρατηρούμε ότι η εφαρμογή περισσότερων πυρήνων δεν σημαίνει αυτόματα ότι μετριέται καλύτερη απόδοση.Αυτό γίνεται καθώς αυξάνεται το πλήθος των sites.Για παράδειγμα η χρήση 4 πυρήνων του επιταχυντή δίνει τη καλύτερη απόδοση για datasets στα οποία κάθε taxa αποτελείται από ακολουθία μεγαλύτερη των 10000 sites.

7.3.2 Επεκτάσεις του επιταχυντή με κλιμάκωση στη συχνότητα

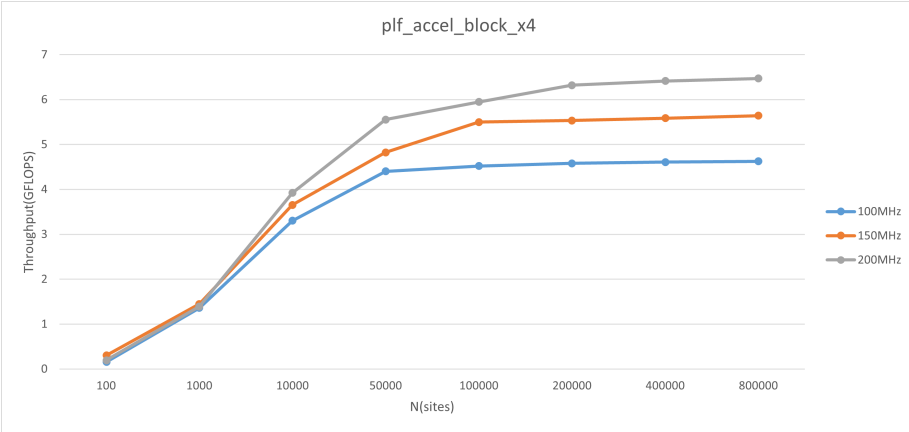
Παράλληλα με τη κλιμάκωση στους πόρους της συσκευής που παρουσιάστηκε στη προηγούμενη ενότητα,εφαρμόζεται κλιμάκωση και στη συχνότητα λειτουργίας του επιταχυντή.Τα διαγράμματα 7.2,7.3 και 7.4 που ακολουθούν παρουσιάζουν τη διεκπεραιωτική ικανότητα των επιταχυντών που υλοποιήθηκαν για ένα,δύο και τέσσερις πυρήνες όταν εφαρμόζονται διαφορετικές συχνότητες στη λειτουργία τους.



Σχήμα 7.2: Σε αυτό το διάγραμμα παρουσιάζεται η απόδοση του επιταχυντή με ένα πυρήνα της αρχιτεκτονικής για την εκτέλεση μιας PLF κλήση με συχνότητες λειτουργίας $100MHz$,$150MHz$,$200MHz$ και $300MHz$.Η απόδοση μετριέται καθώς μεταβάλλεται το πλήθος των sites.Η απόδοση για $300MHz$ γίνεται 1.75 φορές ή 75% μεγαλύτερη από εκείνη στα $100MHz$ λειτουργίας.



Σχήμα 7.3: Σε αντιστοιχία με το διάγραμμα 7.2 το διάγραμμα του σχήματος 7.3 παρουσιάζει την απόδοση επιταχυντή με δύο πυρήνες για $100MHz$,$150MHz$ και $200MHz$ καθώς μεταβάλλεται το πλήθος των sites.

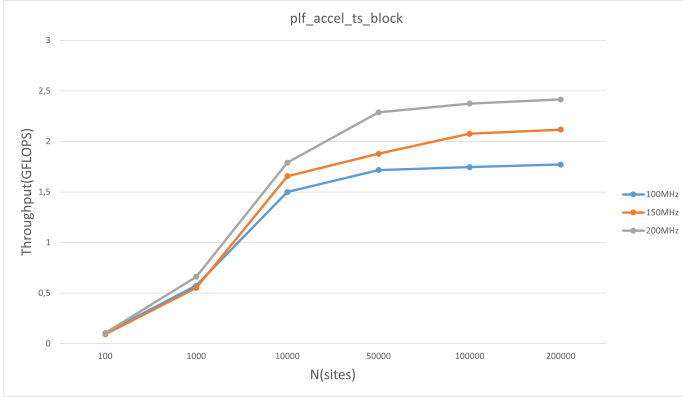


Σχήμα 7.4: Διάγραμμα απόδοσης επιταχυντή τεσσάρων πυρήνων για $100MHz$,$150MHz$ και $200MHz$ καθώς μεταβάλλεται το πλήθος των sites.

Στα παραπάνω διαγράμματα παρατηρείται ότι η απόδοση των επιταχυντών δεν αυξάνεται ανάλογα με την αύξηση της συχνότητας.Το ποσοστό αύξησης της απόδοσης του επιταχυντή είναι μικρότερο από το αντίστοιχο ποσοστό αύξησης της συχνότητας λειτουργίας.Αυτό οφείλεται στο περιορισμό που θέτει η μεταφορά δεδομένων από και προς τον επιταχυντή.

7.3.3 Απόδοση επιταχυντή **plf_accel_ts_unit**

Μέχρι στιγμής έχει αξιολογηθεί η απόδοση των επιταχυντών που υλοποιούν μια κλήση της PLF για τον υπολογισμό διανύσματος απογόνου με είσοδο τα δυο προγονικά διανύσματα.Στο παρακάτω διάγραμμα φαίνεται η διεκπεραιωτική ικανότητα της αρχιτεκτονικής του plf_accel_ts_unit καθώς αυτή κλιμακώνεται στη συσκευή με αύξηση της συχνότητα λειτουργίας.



Σχήμα 7.5: Διάγραμμα απόδοσης επιταχυντή δενδρικής δομής για $100MHz$,$150MHz$ και $200MHz$ καθώς μεταβάλλεται το πλήθος των sites.Η απόδοση μετριέται για μια κλήση του επιταχυντή.Κάθε κλήση του επιταχυντή υπολογίζει τρεις PLF κλήσεις.

.

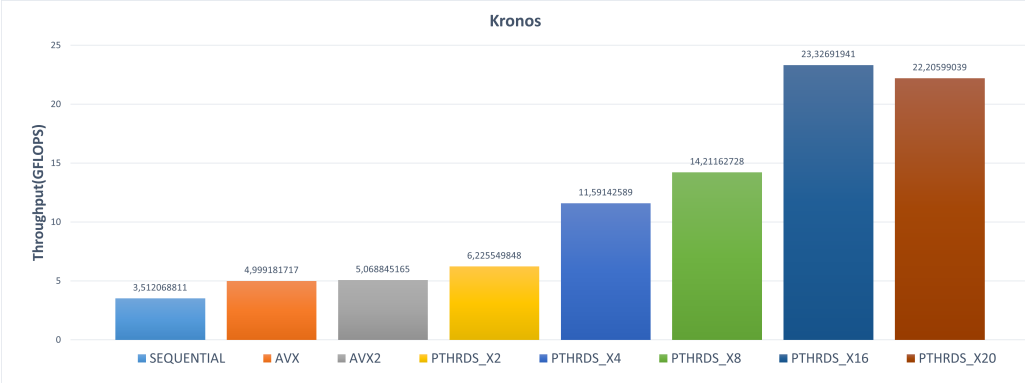
Από το διάγραμμα 7.5 παρατηρούμε ότι η απόδοση του plf_accel_ts_unit είναι αρκετά κοντά με την αντίστοιχη του plf_accel_block με ένα πυρήνα.Αυτό σημβαίνει γιατί χρησιμοποιεί ένα βαθύτερο pipeline.Είναι σημαντικό ότι η απόδοση των δύο επιταχυντών έχει μικρή απόκλιση γιατί δείχνει ότι ο χρόνος εκτέλεσης μιας κλήσης του plf_accel_ts_unit που υπολογίζει έως τρεις PLF κλήσεις είναι ανάλογος με το χρόνο που χρειάζεται το plf_accel_block να εκτελέσει μια και όχι τρεις κλήσεις.

7.3.4 Μέτρηση απόδοσης Πολυπύρηνου Συστήματος Επεξεργασίας

Το υπολογιστικό σύστημα που εξετάζεται υποστηρίζει την εκτέλεση του RAxML για όλες τις δυνατές εκδόσεις του.Οι διάφορες εκδόσεις του RAxML σχετίζονται με την αύξηση του παραλληλισμού στους υπολογισμούς και τη μείωση του εκτελέσιμου χρόνου.Το σχήμα 7.6 παρουσιάζει το throughput που επιτυγχάνεται από το σύστημα για τις διαφορετικές εκδόσεις του RAxML.

Το dataset που χρησιμοποιείται αποτελείται από 10 taxa με αλληλουχίες μήκους 650.000 sites.Το RAxML παράγει 57.000 PLF κλήσεις.Γνωρίζοντας το σχετικά μικρό

πλήθος των κλήσεων και το μεγάλο μέγεθος των αλληλουχιών μπορούμε να θεωρήσουμε ότι ο χρόνος εκτέλεσης των PLF κλήσεων είναι περίπου ίσος με το συνολικό χρόνο εκτέλεσης του RAxML. Συνεπώς για να υπολογίσουμε το **throughput** του συστήματος για την εκτέλεση PLF κλήσεων μπορούμε να χρησιμοποιήσουμε τη σχέση 7.2.



Σχήμα 7.6: Διάγραμμα απόδοσης για την εκτέλεση PLF κλήσεων για τις διαφορετικές εκδόσεις του RAxML.Στην Pthread έκδοση του RAxML χρησιμοποιούνται 2,4,8,16 και 20 νήματα αντίστοιχα.Παρατηρείται ότι η απόδοση για 20 νήματα ελαφρώς μειώνεται σε σχέση με αυτή που χρησιμοποιεί 16 νήματα.

.

7.3.5 Υπολογισμός **speedup** υπολογισμών της **FPGA**

Από τα διαγράμματα των προηγούμενων υποενοτήτων μπορούμε να υπολογίσουμε το **speedup** που επιτυγχάνεται για τον υπολογισμό PLF κλήσεων τόσο πολυπύρηνο σύστημα επεξεργασίας για τις εκδόσεις του RAxML σε σχέση με την ακολουθιακή έκδοση όσο και των επιταχυντών για τις επεκτάσεις της βασικής αρχιτεκτονικής του plf_accel_unit.H παράμετρος **speedup** είναι ένας αριθμός που μετράει τη σχέση της απόδοσης δύο συστημάτων για την επεξεργασία του ίδιου υπολογισμού.

$$speedup = \frac{S1}{S2}$$

, όπου $S1$ και $S2$ δίνουν το *troughput* για τα 2 συστήματα που συγκρίνονται.Ο πρώτος πίνακας που ακολουθεί παρουσιάζει το **speedup** που επιτυγχάνεται στο plf_accel_block με αύξηση της συχνότητας λειτουργίας ενώ ο δεύτερος παρουσιάζει το **speedup** που επιτυγχάνεται με χρήση περισσότερων πυρήνων.

Cores	Frequency	Max Throughput(GFLOPS)	Speedup
1	100	1.89	x1
1	150	2.54	x1.34
1	200	3.06	x1.62
1	300	3.87	x2.04

Σχήμα 7.7: Ο πίνακας παρουσιάζει το speedup που επιτυγχάνεται στην απόδοση του επιταχυντή plf_accel_block καθώς η υλοποίηση του κλιμακώνεται στη συχνότητα.

.

Cores	Frequency	Max Throughput(GFLOPS)	Speedup
1	300	3.87	x1
2	200	4.46	x1.15
4	200	6.47	x1.67

Σχήμα 7.8: Ο πίνακας παρουσιάζει το speedup που επιτυγχάνεται στο χρόνο εκτέλεσης μιας PLF κλήσης όταν η αρχιτεκτονική του plf_accel_block χρησιμοποιεί περισσότερους πυρήνες.Η απόδοση των επιταχυντών δε συγκρίνεται για ίδια συχνότητα λειτουργία αλλά για τη μέγιστη που επιτυγχάνεται από τη κάθε υλοποίηση πάνω στη πλακέτα ZCU102.

.

Η ποσότητα των δεδομένων που μπορεί να σταλεί από το σύστημα επεξεργασίας της εφαρμογής και να δεχθεί η FPGA αποτελεί καθοριστικό παράγοντα στην απόδοση των επιταχυντών.Η εφαρμογή των πράξεων με το μέγιστο παραλληλισμό φράζεται από τη ποσότητα δεδομένων που μπορεί να δεχθεί ο επιταχυντής σε κάθε κύκλο ρολογιού.Αυτός είναι ο βασικός λόγος που η αύξηση της συχνότητας λειτουργίας ή η χρήση πολλαπλών πυρήνων δεν οδηγεί σε ανάλογη αύξηση του throughput.

Ο plf_accel_block σε συχνότητα 300MHz καθώς και ο διπύρηνος επιταχυντής για οποιαδήποτε συχνότητα λειτουργίας μπορεί να δώσει καλύτερη απόδοση από την εκτέλεση της ακολουθιακής έκδοσης του RAxML.Ακόμα με τη χρήση του τετραπύρηνου επιταχυντή στα 150MHz επιτυγχάνεται καλύτερη απόδοση σε σχέση με την εκτέλεση της έκδοσης του RAxML που χρησιμοποιεί την επέκταση μικροαρχιτεκτονικής AVX και AVX2 ενώ αυξάνοντας της συχνότητα στα 200MHz επιτυγχάνεται ελαφρώς καλύτερη απόδοση σε σχέση με την εκτέλεση της έκδοσης του RAxML με Pthreads όταν χρησιμοποιούνται 2 πυρήνες.Ο πίνακας που ακολουθεί παρουσιάζει το speedup που επιτυγχάνεται από τον τετραπύρηνο επιταχυντή σε σχέση με το σύστημα επεξεργασίας όπως περιγράφηκε παραπάνω.

RAxML sw version	hw core	hw frequency(MHz)	RAxML sw throughput(GFLOPS)	hw throughput(GFLOPS)	Speedup
sequential	1	300	3,51	3,87	x1,1025641025641
sequential	2	200	3,51	4,46	x1,27065527065527
AVX	2	200	4,99	4,46	x0,893787575150301
AVX2	2	200	5,06	4,46	x0,881422924901186
sequential	4	200	3,51	6,46	x1,84045584045584
AVX	4	200	4,99	6,46	x1,29458917835671
AVX2	4	200	5,06	6,46	x1,27667984189723
Pthreads_x2	4	200	6,22	6,46	x1,03858520900322

Σχήμα 7.9: Ο πίνακας παρουσιάζει το speedup που επιτυγχάνεται από τις διαφορετικές εκδόσεις του επιταχυντή για την εκτέλεση μιας PLF κλήσης.Η σύγκριση γίνεται με την εκτέλεση κάποιων εκδόσεων του RAxML από πολυπύρηνο σύστημα επεξεργασίας.

.

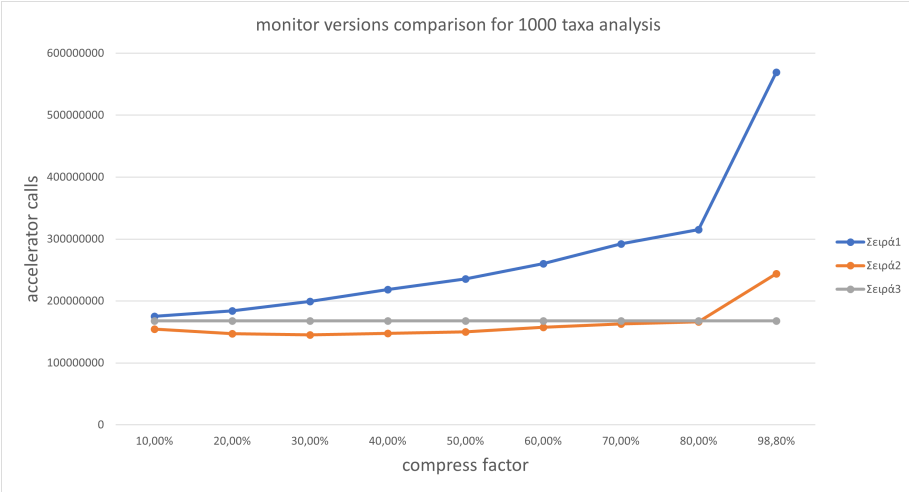
7.4 Συνολική αξιολόγηση συστήματος

Η ενότητα αυτή παρουσιάζει την αξιολόγηση της εκτέλεσης φυλογενετικών αναλύσεων χρησιμοποιώντας τα δύο λογισμικά διαχείρισης των PLF κλήσεων, **monitor** και **monitor_ts**,καθώς και τις διαφορετικές επεκτάσεις του επιταχυντή για την εκτέλεση αυτών των κλήσεων.Ο επιταχυντής **plf_accel_block** όπως και η διπύρηνη και τετραπύρηνη εκδοχή του χρησιμοποιούνται μαζί με το **monitor**.Ο **plf_accel_ts_block** χρησιμοποιείται μαζί με το **monitor_ts**. Υπενθυμίζεται ότι κάθε κλήση που παράγει το **monitor** αντιστοιχίζεται σε μια PLF κλήση, ενώ μια κλήση του **monitor_ts** αντιστοιχίζεται από μια έως τρεις PLF κλήσεις.Οι δύο εκδόσεις συγκρίνονται με βάση τη ποσότητα των κλήσεων που παράγουν όταν δέχονται στην είσοδο το σύνολο των PLF κλήσεων από μια φυλογενετική ανάλυση του RAxML.Το μέτρο της αξιολόγησης του συνολικού συστήματος είναι η επιτάχυνση της εκτέλεσης μιας φυλογενετικής ανάλυσης ενώ εφαρμόζεται συμπίεση των δεδομένων που αποθηκεύονται στη μνήμη.

Για τα πειράματα παράγουμε 2 σύνολα δεδομένων με 100 και 1000 **taxa** αντίστοιχα.Οι κλήσεις που παράγονται από το RAxML δίνονται ως είσοδος στις 2 εκδόσεις του **monitor**.Ακολούθως αυτά παράγουν νέα σύνολα PLF κλήσεων, το πλήθος των οποίων μεταβάλλεται καθώς μεταβάλλεται και η τιμή του **compress factor**.Τα διαγράμματα στις εικόνες 7.10,7.11 συγκρίνουν το πλήθος των κλήσεων που παράγονται από τις δυο εκδόσεις του **monitor** για τις δύο φυλογενετικές αναλύσεις.Ταυτόχρονα τα διαγράμματα συγκρίνουν τις κλήσεις που παράγουν τα **monitor** και **monitor_ts** σε σχέση με τις κλήσεις που παράγονται από το RAxML.

Ως κατώτατο όριο συμπίεσης στο σύστημα διαχείρισης της μνήμης δοκιμάζεται το 10% των δεδομένων και ως ανώτατο η ελάχιστη ποσότητα που απαιτείται για να μπορούν να πραγματοποιηθούν οι υπολογισμοί της συνάρτησης, η οποία είναι ίση με $Q = \log_2 n + 2$.Η ποσότητα Q προκύπτει για 100 **taxa** περίπου ίση με 9 και για 1000 **taxa** περίπου ίση με 12.Με βάση αυτές τις τιμές προκύπτει ότι ανώτατο ποσοστό συμπίεσης για 100 **taxa** είναι το 91 % και για 1000 **taxa** το 98,8 %.

Σχήμα 7.10: Το διάγραμμα παρουσιάζει τη μεταβολή του πλήθους των κλήσεων που παράγονται από τα monitor,monitor_ts καθώς μεταβάλλεται ο compressFactor.Οι κλήσεις υπολογίζουν φυλογενετική ανάλυση για 100 taxa.Ταυτόχρονα οι κλήσεις που παράγονται από τα δυο λογισμικά συγκρίνονται με το πλήθος των κλήσεων που παράγει το RAxML χωρίς την εφαρμογή κάποιας συμπίεσης δεδομένων στη μνήμη.



Σχήμα 7.11: Σύγκριση κλήσεων που παράγονται για την υλοποίηση φυλογενετικής ανάλυσης για 1000 taxa από RAxML,monitor,monitor_ts.Το διάγραμμα υπολογίζεται όπως και το 7.7.Αυτό που διαφοροποιείται είναι το μέγεθος της φυλογενετικής ανάλυσης.

Από τα παραπάνω διαγράμματα παρατηρούμε ότι οι κλήσεις που παράγονται από το monitor_ts είναι αισθητά λιγότερες από αυτές που παράγονται από το monitor.Ακόμα

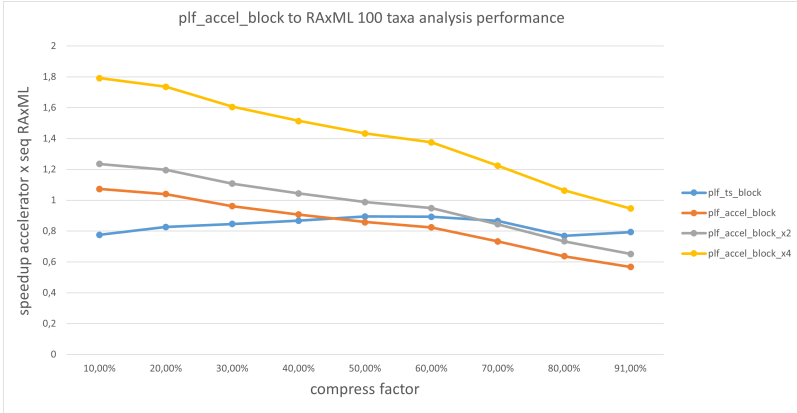
είναι φανερό ότι το πλήθος των κλήσεων του monitor_ts είναι μικρότερα ακόμα και σε σχέση με το πλήθος των κλήσεων που παράγει το RAxML ακόμα και για υψηλά ποσοστά συμπίεσης.Οι κλήσεις του υπερβαίνουν αυτές του RAxML μονάχα για ποσοστά συμπίεση που είναι ίσα με το θεωρητικό όριο Q ή κοντά σε αυτό.

Από τα διαγράμματα που παρουσιάζουν την απόδοση των επιταχυντών φάνηκε ότι η διεκπεραιωτική ικανότητα του plf_accel_ts_unit είναι αρκετά κοντά με αυτή του plf_accel_unit για την ίδια συχνότητα λειτουργίας.Παράλληλα είναι συγκρίσιμη με την αντίστοιχη απόδοση του πολυπύρηνου συστήματος επεξεργασίας για την εκτέλεση της sequential έκδοσης του RAxML. Γνωρίζοντας ότι οι κλήσεις που παράγει το monitor_ts είναι λιγότερες από αυτές που παράγει το RAxML ακόμα και για μεγάλα ποσοστά συμπίεσης της μνήμης, ενώ ταυτόχρονα ο επιταχυντής plf_accel_ts_unit έχει ικανοποιητική απόδοση μπορούμε να υποθέσουμε ότι το σύστημα που παρουσιάζεται αποτελεί μια κατάλληλη λύση για μείωση των απαιτήσεων μνήμης με μικρή επιβάρυνση στο χρόνο εκτέλεσης ή ακόμα και με μείωση του χρόνου εκτέλεσης της φυλογενετικής ανάλυσης.

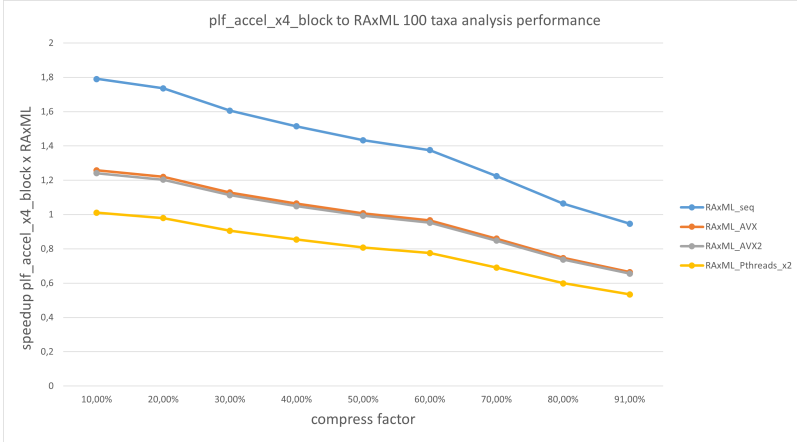
Τα επόμενα διαγράμματα στηρίζονται σε 2 φυλογενετικές αναλύσεις που εκτελούνται για dataset με 100 και 1.000 taxa αντίστοιχα και 5.000.000 sites per taxa. Υπολογίζεται ο χρόνος εκτέλεσης των αναλύσεων από το RAxML στο πολυπύρηνο σύστημα καθώς και ο χρόνος όταν για την εκτέλεση των αναλύσεων εφαρμόζεται συμπίεση των δεδομένων στη μνήμη και παράλληλα χρησιμοποιούνται τα monitor και monitor_ts και οι διαφορετικές επεκτάσεις του επιταχυντή.Ακολούθως υπολογίζεται, ο χρόνος εκτέλεσης των φυλογενετικών αναλύσεων, για τα εξής διαφορετικά ζεύγη monitor-επιταχυντή: {monitor , plf_accel_block},{monitor, plf_accel_block_x2}, {monitor, plf_accel_block_x4},{monitor_ts , plf_accel_ts_block}.

Στο τέλος υπολογίζεται ο λόγος του χρόνου που χρειάζεται καθένα από τα παραπάνω ζεύγη για την εκτέλεση της φυλογενετικής ανάλυσης προς τον αντίστοιχο χρόνο του πολυπύρηνου για την εκτέλεση του sequential RAxML.Τα διαγράμματα 7.12 και 7.14 παρουσιάζουν πως μεταβάλλεται το ratio των χρόνων εκτέλεσης καθώς αυξάνεται η συμπίεση των δεδομένων στη μνήμη.

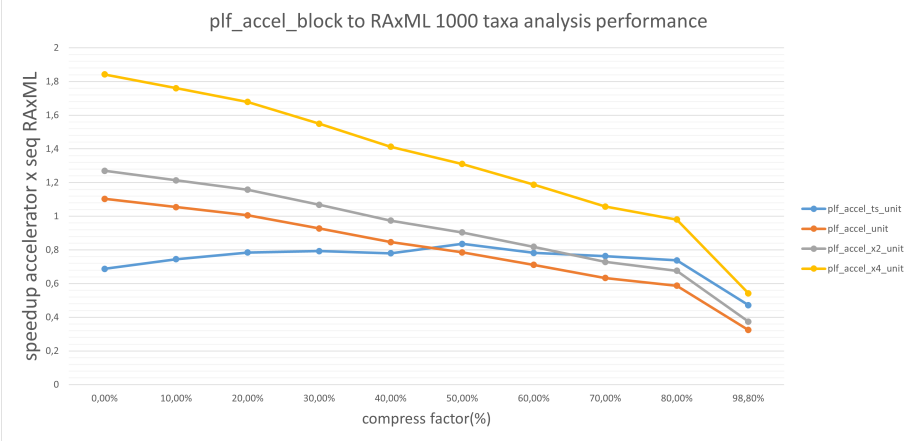
Στα διαγράμματα 7.13 και 7.15 υπολογίζεται το ratio του χρόνου του συστήματος που χρησιμοποιεί την τετραπύρηνη έκδοση του επιταχυντή προς τους αντίστοιχους χρόνους για τις sequential,AVX,AVX2,Pthreads εκδόσεις του RAxML ενώ αυξάνεται ο compressFactor.



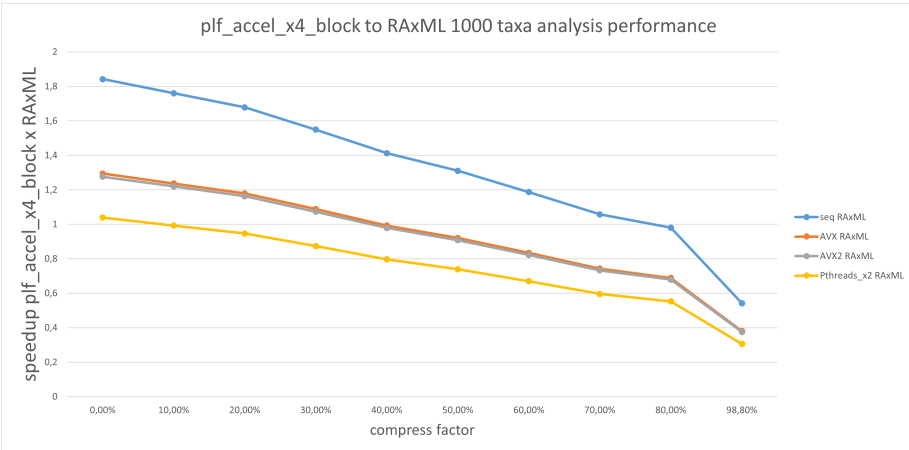
Σχήμα 7.12: Το διάγραμμα παρουσιάζει την απόδοση του συστήματος σε σχέση με το sequential RAxML, για την εκτέλεση φυλογενετικής ανάλυσης με 100 taxa και 5.000.000 sites per probability vector.Ο κάθετος άξονας δείχνει το λόγο του χρόνου εκτέλεσης της ανάλυσης από μια έκδοση του συστήματος προς αντίστοιχο χρόνο του sequential RAxML.Τρεις από τις 4 γραμμές (κόκκινη,γρι,κίτρινη) χρησιμοποιούν το λογισμικό monitor και τις επεκτάσεις του επιταχυντή με 1,2,4 πυρήνες αντίστοιχα για την εκτέλεση των PLF κλήσεων που παράγει το monitor.Η κόκκινη γραμμή δείχνει τη σχέση του χρόνου εκτέλεσης του συστήματος προς το sequential RAxML όταν χρησιμοποιούνται το monitor_ts και ο επιταχυντής plf_accel_ts_block.



Σχήμα 7.13: Το διάγραμμα παρουσιάζει την απόδοση του συστήματος που χρησιμοποιεί το λογισμικό monitor και τον επιταχυντή plf_accel_block_x4 σε σχέση με την απόδοση του πολυπύρηνου συστήματος για την εκτέλεση του sequential,AVX,AVX2,Pthreads RAxML.Η φυλογενετική ανάλυση που υπολογίζεται αποτελείται από 100 taxa και μέγεθος DNA ακολουθίας για κάθε taxa ίση με 5.000.000 sites.



Σχήμα 7.14: Το διάγραμμα παρουσιάζει την απόδοση του συστήματος σε σχέση με το sequential RAxML, για την εκτέλεση φυλογενετικής ανάλυσης με 1000 taxa και μέγεθος DNA ακολουθίας για κάθε taxa ίση με 5.000.000 sites.Το διάγραμμα υπολογίζεται όπως και το 7.9.



Σχήμα 7.15: Το διάγραμμα παρουσιάζει την απόδοση του συστήματος που χρησιμοποιεί το λογισμικό monitor και τον επιταχυντή plf_accel_block_x4 σε σχέση με την απόδοση για την εκτέλεση του sequential,AVX,AVX2,Pthreads RAxML.Η φυλογενετική ανάλυση που υπολογίζεται αποτελείται από 1000 taxa και μέγεθος DNA ακολουθίας για κάθε taxa ίση με 5.000.000 sites.

Από τα διαγράμματα 7.9 και 7.11 μπορεί κανείς να παρατηρήσει ότι ο χρόνος εκτέλεσης του προτεινόμενου συστήματος,για οποιοδήποτε ζεύγος monitor-επιταχυντή, δεν αυξάνεται με τον ίδιο αλλά με πολύ μικρότερο ρυθμό από εκείνο που αυξάνεται ο compressfactor. Παρατηρείται ότι για συμπίεση δεδομένων μεγαλύτερη του 50%,το προτεινόμενο σύστημα που αποτελείται από το monitor_ts και το plf_accel_ts_block επιτυγχάνει μικρότερο ratio από το ζεύγος {monitor,plf_accel_block}.Για συμπίεση μεγαλύτερη του 70% το ratio γίνεται μικρότερο και από εκείνο του monitor,plf_accel_block_x2

παρότι το throughput του plf_accel_block_x2 είναι 1,8 φορές μεγαλύτερο.

Επιπλέον από τα διαγράμματα 7.10 και 7.12 παρατηρείται ότι το σύστημα με χρήση του plf_accel_block_x4 μπορεί να επιτύχει μεγάλη συμπίεση των δεδομένων με μικρή αύξηση του ratio ακόμα και όταν το σύστημα επεξεργασίας εκτελεί τις βελτιωμένες εκδόσεις AVX,AVX2,Pthreads του RAxML.Συνεπώς η χρήση περισσότερων πόρων για επέκταση του plf_accel_ts_unit με περισσότερους πυρήνες και για λειτουργία σε μεγαλύτερες συχνότητες μπορεί να καταστήσει το συνολικό σύστημα ιδιαίτερα αποδοτικό για την εκτέλεση φυλογενετικών αναλύσεων μεγάλης έκτασης.

Κεφάλαιο 8

Επίλογος

Στο τελευταίο κεφάλαιο παρουσιάζεται συνοπτικά οι λύσεις που προτάθηκαν στα πλαίσια της εργασίας και πιθανές μελλοντικές επεκτάσεις.

8.1 Συμπεράσματα

Η εργασία παρουσιάζει το πρόβλημα του μεγάλου κόστους σε πόρους μνήμης σε εκτελέσεις φυλογενετικών αναλύσεων και προτείνει λύση που βασίζεται στη συμφέρουσα ανταλλαγή πόρων μνήμης με κάποια ανεκτή επιβάρυνση στο χρόνο εκτέλεσης.Συγκεκριμένα εξετάζεται το κόστος σε πόρους μνήμης προγράμματος φυλογενετικών αναλύσεων που χρησιμοποιεί τον αλγόριθμο SPR για τη κατασκευή φυλογενετικών δέντρων και το κριτήριο της μέγιστης πιθανοφάνειας για την αξιολόγηση τους.Η εργασία παρουσιάζει τόσο τον SPR αλγόριθμο καθώς και το τρόπο που εφαρμόζεται στο ευρέως διαδεδομένο πρόγραμμα RAxML όσο και τα μαθηματικά μοντέλα που προκύπτουν από το κριτήριο της μέγιστης πιθανοφάνειας.

Η εξοικονόμηση πόρων μνήμης επιτυγχάνεται με το σύστημα λογισμικού **monitor**.Το **monitor** λαμβάνει τις κλήσεις της φυλογενετικής συνάρτησης που παράγονται από το RAxML.Όταν τα δεδομένα από ένα σύνολο κόμβων του φυλογενετικού δέντρου απομακρύνονται από τη μνήμη, το **monitor** παράγει ένα σύνολο από επιπρόσθετες PLF κλήσεις ώστε να πραγματοποιηθεί η αξιολόγηση του φυλογενετικού δέντρου.Οι επιπρόσθετες PLF κλήσεις κάποιο επιπλέον κόστος στο χρόνο εκτέλεσης της ανάλυσης.Για το λόγο αυτό σχεδιάζονται και υλοποιούνται σε αναδιατασσόμενη λογική μονάδες επιταχυντών που εκτελούν PLF κλήσεις.Το **monitor** δέχεται ένα σύνολο κλήσεων από το πρόγραμμα φυλογενετικών αναλύσεων και παράγει ένα δεύτερο σύνολο κλήσεων που εκτελούνται από τον χρησιμοποιούμενο επιταχυντή.Ακόμα υλοποιείται επιταχυντής που δίνει τη δυνατότητα να ενσωματωθούν σε μια κλήση του επιταχυντή έως τρεις PLF κλήσεις.Αυτό είναι ωφέλιμο γιατί δίνει τη δυνατότητα για ακόμα μεγαλύτερη επιτάχυν-

ση στο χρόνο εκτέλεσης για το ίδιο ποσοστό συμπίεσης δεδομένων στη μνήμη.Με βάση αυτό σχεδιάζεται μια δεύτερη έκδοση του **monitor** που παράγει κλήσεις προς τον επιταχυντή οι οποίες ενσωματώνουν εως τρεις PLF κλήσεις.

Χρησιμοποιώντας αυτές τις υλοποιήσεις το προτεινόμενο σύστημα μπορεί να εκτελεί φυλογενετικές αναλύσεις όπου μεγάλο ποσοστό των εσωτερικών κόμβων δεν αποθηκεύεται στη μνήμη και ταυτόχρονα ο χρόνος εκτέλεσης μιας ανάλυσης συγκρινόμενος με τον αντίστοιχο χρόνο που χρειάζεται ένα πολυπύρηνο επεξεργαστικό σύστημα μπορεί ακόμα και να μειωθεί για μικρό ποσοστό συμπίεσης των δεδομένων της μνήμης ενώ για μεγάλα ποσοστά συμπίεσης η επιβάρυνση στο χρόνο εκτέλεσης είναι αναλογικά μικρότερη.

8.2 Μελλοντική Δουλειά

Η εργασία δείχνει ότι μπορεί να κατασκευαστεί σύστημα εκτέλεσης φυλογενετικών αναλύσεων που να εξοικονομεί πόρους μνήμης με μια δυσανάλογα μικρότερη επιβάρυνση στο χρόνο εκτέλεσης.Επεκτάσεις του προτεινόμενου συστήματος μπορεί να είναι:

- επέκταση της αρχιτεκτονικής του επιταχυντή δενδρικής δομής ώστε να υλοποιεί σε μια κλήση **post-order** διάσχιση μεγαλύτερου βάθους και συνεπώς περισσότερες PLF κλήσεις.Με αυτό το τρόπο μπορεί να δωθεί η δυνατότητα για επιτάχυνση του χρόνου εκτέλεσης μια ανάλυσης για μεγάλα ποσοστά συμπίεσης των δεδομένων στη μνήμη.
- επέκταση της αρχιτεκτονικής των προτεινόμενων επιταχυντών με περισσότερους πυρήνες σε μεγαλύτερη πλατφόρμα για μεγαλύτερη απόδοση στην εκτέλεση PLF κλήσεων.
- προσαρμογή της αρχιτεκτονικής των επιταχυντών ώστε να υποστηρίζουν την εκτέλεση PLF κλήσεων από μοριακά δεδομένων πρωτεϊνών.
- ενσωμάτωση του λογισμικού **monitor** με δημιουργία κατάλληλης διεπαφής στο RAxML.

Bibliography

[1] Vivado high level synthesis tutorial. url:https://www.xilinx.com/support/documentation $vivado-high-level-synthesis-tutorial.pdf$.

[2] Xilinx sdsoc tutorial. url:https://www.xilinx.com/support/documentation/sw$_m$$anuals/xilinx$ $sdsoc-user-guide.pdf$.

[3] Nikolaos Alachiotis, Euripides Sotiriades, Apostolos Dollas, and Alexandros Stamatakis. Exploring fpgas for accelerating the phylogenetic likelihood function. In *2009 IEEE International Symposium on Parallel & Distributed Processing*, pages 1–8. IEEE, 2009.

[4] Nikolaos Alachiotis and Alexandros Stamatakis. A generic and versatile architecture for inference of evolutionary trees under maximum likelihood. In *2010 Conference Record of the Forty Fourth Asilomar Conference on Signals, Systems and Computers*, pages 829–835. IEEE, 2010.

[5] Nikolaos Alachiotis, Alexandros Stamatakis, Euripides Sotiriades, and Apostolos Dollas. A reconfigurable architecture for the phylogenetic likelihood function. In *2009 International Conference on Field Programmable Logic and Applications*, pages 674–678. IEEE, 2009.

[6] Amos Bairoch and Brigitte Boeckmann. The swiss-prot protein sequence data bank. *Nucleic acids research*, 19(Suppl):2247, 1991.

[7] Dennis A Benson, Ilene Karsch-Mizrachi, David J Lipman, James Ostell, and Eric W Sayers. Genbank. *Nucleic acids research*, 37(suppl_1):D26–D31, 2008.

[8] S Berger, Nikolaos Alachiotis, and Alexandros Stamatakis. An optimized reconfigurable system for computing the phylogenetic likelihood on dna data. In *Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 2009 IEEE International Symposium on*, pages 1–8, 2012.

[9] Simon A Berger and Alexandros Stamatakis. Accuracy and performance of single versus double precision arithmetics for maximum likelihood phylogeny reconstruction. In *International Conference on Parallel Processing and Applied Mathematics*, pages 270–279. Springer, 2009.

[10] Benny Chor and Tamir Tuller. Maximum likelihood of evolutionary trees: Hardness and approximation. *Bioinformatics (Oxford, England)*, 21 Suppl 1:i97–106, 07 2005.

[11] Pavol Duris and Zvi Galil. A time-space tradeoff for language recognition. volume 17, pages 53–57, 01 1981.

[12] Robert C Edgar. Muscle: multiple sequence alignment with high accuracy and high throughput. *Nucleic acids research*, 32(5):1792–1797, 2004.

[13] Robert C Edgar and Serafim Batzoglou. Multiple sequence alignment. *Current opinion in structural biology*, 16(3):368–373, 2006.

[14] Joseph Felsenstein. Evolutionary trees from dna sequences: A maximum likelihood approach. *Journal of molecular evolution*, 17:368–76, 02 1981.

[15] Walter M Fitch. On the problem of discovering the most parsimonious tree. *The American Naturalist*, 111(978):223–257, 1977.

[16] Stéphane Guindon, Frédéric Delsuc, Jean-François Dufayard, and Olivier Gascuel. Estimating maximum likelihood phylogenies with phyml. In *Bioinformatics for DNA sequence analysis*, pages 113–137. Springer, 2009.

[17] Masami Hasegawa, Hirohisa Kishino, and Taka aki Yano. Dating of the human-ape splitting by a molecular clock of mitochondrial dna. *Journal of Molecular Evolution*, 22:160–174, 1985.

[18] Masami Hasegawa, Hirohisa Kishino, and Taka-aki Yano. Dating of the human-ape splitting by a molecular clock of mitochondrial dna. *Journal of molecular evolution*, 22(2):160–174, 1985.

[19] Andrew P Hendry, Michael T Kinnison, Mikko Heino, Troy Day, Thomas B Smith, Gary Fitt, Carl T Bergstrom, John Oakeshott, Peter S Jørgensen, Myron P Zalucki, et al. Evolutionary principles and their practical application. *Evolutionary Applications*, 4(2):159–183, 2011.

[20] Mark Holder and Paul O Lewis. Phylogeny estimation: traditional and bayesian approaches. *Nature reviews genetics*, 4(4):275, 2003.

[21] John P Huelsenbeck and Fredrik Ronquist. Mrbayes: Bayesian inference of phylogenetic trees. *Bioinformatics*, 17(8):754–755, 2001.

[22] Fernando Izquierdo-Carrasco. Inference of many-taxon phylogenies. 2014.

[23] Fernando Izquierdo-Carrasco, Julien Gagneur, and Alexandros Stamatakis. Trading memory for running time in phylogenetic likelihood computations. *Heidelberg Institute for Theoretical Studies*, 01 2011.

[24] Fernando Izquierdo-Carrasco, Julien Gagneur, and Alexandros Stamatakis. Trading memory for running time in phylogenetic likelihood computations. *Heidelberg Institute for Theoretical Studies*, 01 2011.

[25] Fernando Izquierdo-Carrasco and Alexandros Stamatakis. Computing the phylogenetic likelihood function out-of-core. pages 444 – 451, 06 2011.

[26] TH Jukes and CR Cantor. Evolution of protein molecules. mammalian protein metabolism. edited by: Munro hn. 1969.

[27] Server Kasap and Khaled Benkrid. High performance phylogenetic analysis with maximum parsimony on reconfigurable hardware. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 19(5):796–808, 2010.

[28] Kazutaka Katoh, George Asimenos, and Hiroyuki Toh. Multiple alignment of dna sequences with mafft. In *Bioinformatics for DNA sequence analysis*, pages 39–64. Springer, 2009.

[29] Motoo Kimura et al. Evolutionary rate at the molecular level. *Nature*, 217(5129):624–626, 1968.

[30] Oleksii Kozlov. Models, optimizations, and tools for large-scale phylogenetic inference, handling sequence uncertainty, and taxonomic validation.

[31] Εμμανουήλ Λαδουκάκης. Η *εξέλιξη ως επιστήμη*. 2015.

[32] Cheng Ling, Chunbao Zhou, Arong Luo, Guoguang Zhao, Tsuyoshi Hamada, and Xiaoyan Zhu. Optimizing the bayesian inference of phylogeny on graphic processors. In *2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 333–342. IEEE, 2015.

[33] Pietro Liò and Nick Goldman. Models of molecular evolution and phylogeny. *Genome research*, 8(12):1233–1244, 1998.

[34] Eman M Mohamed. Enhanced probcons for multiple sequence alignment in cloud computing. 2019.

[35] Masatoshi Nei, Fumio Tajima, and Yoshio Tateno. Accuracy of estimated phylogenetic trees from molecular data. *Journal of molecular evolution*, 19(2):153–170, 1983.

[36] Shane O' Reilly. *Organic carbon cycling in marine sediments and seabed seepage features in Irish waters*. PhD thesis, 11 2013.

[37] Wayne Pfeiffer and Alexandros Stamatakis. Hybrid mpi/pthreads parallelization of the raxml phylogenetics code. pages 1 – 8, 05 2010.

[38] Frederico Pratas, Pedro Trancoso, Alexandros Stamatakis, and Leonel Sousa. Fine-grain parallelism using multi-core, cell/be, and gpu systems: accelerating the phylogenetic likelihood function. In *2009 International Conference on Parallel Processing*, pages 9–17. IEEE, 2009.

[39] Sandun Rajapaksa, Wageesha Rasanjana, Indika Perera, and Dulani Meedeniya. Gpu accelerated maximum likelihood analysis for phylogenetic inference. In *Proceedings of the 2019 8th International Conference on Software and Computer Applications*, pages 6–10, 2019.

[40] Naruya Saitou and Masatoshi Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular biology and evolution*, 4(4):406–425, 1987.

[41] Martin Simonsen, Thomas Mailund, and Christian NS Pedersen. Rapid neighbour-joining. In *International Workshop on Algorithms in Bioinformatics*, pages 113–122. Springer, 2008.

[42] Martin Simonsen, Thomas Mailund, and Christian NS Pedersen. Building very large neighbour-joining trees. In *BIOINFORMATICS*, pages 26–32, 2010.

[43] Alexandros Stamatakis. The raxml v8. 2. x manual. *Heidelberg Institute for Theoretical Studies. Available from: http://sco. h-its. org/exelixis/web/software/raxml/# documentation (accessed 20 July 2016)*, 2016.

[44] Alexandros Stamatakis, Thomas Ludwig, and Harald Meier. Raxml-iii: A fast program for maximum likelihood-based inference of large phylogenetic trees. *Bioinformatics (Oxford, England)*, 21:456–63, 03 2005.

[45] Jean-Francois Taly, Cedrik Magis, Giovanni Bussotti, Jia-Ming Chang, Paolo Di Tommaso, Ionas Erb, Jose Espinosa-Carrasco, Carsten Kemena, and Cedric Notredame. Using the t-coffee package to build multiple sequence alignments of protein, rna, dna sequences and 3d structures. *Nature protocols*, 6(11):1669–1682, 2011.

[46] Simon Tavaré. Some probabilistic and statistical problems in the analysis of dna sequences. *Lectures on mathematics in the life sciences*, 17(2):57–86, 1986.

[47] Julie D Thompson, Toby J Gibson, and Des G Higgins. Multiple sequence alignment using clustalw and clustalx. *Current protocols in bioinformatics*, (1):2–3, 2003.

[48] Jeffrey L. Thorne, Hirohisa Kishino, and Joseph Felsenstein. An evolutionary model for maximum likelihood alignment of dna sequences. *Journal of Molecular Evolution*, 33(2):114–124, Aug 1991.

[49] Jeffrey Vitter. External memory algorithms and data structures. 05 1999.

[50] Simon Whelan and David A Morrison. Inferring trees. In *Bioinformatics*, pages 349–377. Springer, 2017.

[51] Jun Xu and Richard J Lipton. On fundamental tradeoffs between delay bounds and computational complexity in packet scheduling algorithms. *IEEE/ACM Transactions on Networking*, 13(1):15–28, 2005.

[52] Ziheng Yang. Maximum likelihood phylogenetic estimation from dna sequences with variable rates over sites: approximate methods. *Journal of Molecular evolution*, 39(3):306–314, 1994.

[53] Ziheng Yang. Among-site rate variation and its impact on phylogenetic analyses. *Trends in Ecology & Evolution*, 11(9):367–372, 1996.

[54] Ziheng Yang and Bruce Rannala. Molecular phylogenetics: Principles and practice. *Nature reviews. Genetics*, 13:303–14, 03 2012.

[55] Xiaofan Zhou, Xing-Xing Shen, Chris Todd Hittinger, and Antonis Rokas. Evaluating fast maximum likelihood-based phylogenetic programs using empirical phylogenomic data sets. *Molecular biology and evolution*, 35(2):486–503, 2017.

[56] Stephanie Zierke and Jason D Bakos. Fpga acceleration of the phylogenetic likelihood function for bayesian mcmc inference methods. *BMC bioinformatics*, 11(1):1–12, 2010.