ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΕΡΓΑΣΤΗΡΙΟ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Υλοποίηση και Βελτιστοποίηση του Αλγορίθμου Smith - Waterman σε Πολυπύρηνους Επεξεργαστές και Πολυνηματικούς Επεξεργαστές Γραφικών

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Νικόλαος Σ. Κούτσικος

Επιβλέπων : Νεκτάριος Κοζύρης

Αναπληρωτής Καθηγητής Ε.Μ.Π.

Αθήνα, Σεπτέμβριος 2012

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΕΡΓΑΣΤΗΡΙΟ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Υλοποίηση και Βελτιστοποίηση του Αλγορίθμου Smith - Waterman σε Πολυπύρηνους Επεξεργαστές και Πολυνηματικούς Επεξεργαστές Γραφικών

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Νικόλαος Σ. Κούτσικος

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

Εγκρίθηκε από την τριμερή εξεταστική επιτροπή την 5η Σεπτεμβρίου 2012.

.....	.....	.....
Ν. Κοζύρης	Π. Τσανάκας	Δ.Φωτάκης
Αν. Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Λέκτορας Ε.Μ.Π.

Αθήνα, Σεπτέμβριος 2012

.....

Νικόλαος Σ. Κούτσικος
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Ο αλγόριθμος Smith-Waterman αποτελεί έναν από τους πιο σημαντικούς αλγόριθμους που χρησιμοποιείται στην Βιοπληροφορική. Πραγματοποιεί τοπική στοίχιση βιολογικών ακολουθιών, η οποία δίνει την δυνατότητα κατανόησης σε βάθος των βιολογικών λειτουργιών, αναγνώρισης των μεταλλάξεων και προσδιορισμό της γενεολογίας. Σκοπός αυτής της διπλωματικής εργασίας, είναι η περιγραφή και ανάλυση της λειτουργίας του αλγορίθμου Smith-Waterman, εύρεση των σημείων που μπορεί να παραλληλοποιηθεί και τελικά υλοποίηση του τόσο σε πολυπύρηνους επεξεργαστές, όσο και σε επεξεργαστές γραφικών. Αναλύεται σε βάθος η αρχιτεκτονική των επεξεργαστών γραφικών, που αποτελούν ανερχόμενη λύση στο χώρο των συστημάτων υψηλής επίδοσης και παρουσιάζονται οι βασικές τεχνικές αποδοτικής εκμετάλλευσής τους. Τέλος εφαρμόζονται διάφορες βελτιστοποιήσεις που έχουν ως σκοπό την αύξηση της επίδοσης του αλγορίθμου αλλά και την διευκρίνηση των χαρακτηριστικών που αποτελούν όριο για την επίδοση. Αναλύονται τα αποτελέσματα και εξάγονται σημαντικά συμπεράσματα για την συμπεριφορά του αλγορίθμου, κάτι που μας δίνει την δυνατότητα να προτείνουμε ιδέες και μελλοντικές κατευθύνσεις για έρευνα.

Λέξεις Κλειδιά

Αλγόριθμος Smith-Waterman, τοπική στοίχιση ακολουθιών, Βιοπληροφορική, Δυναμικός προγραμματισμός, παράλληλη αρχιτεκτονική, πολυπύρηνος επεξεργαστής, επεξεργαστής γραφικών, GPGPU, Nvidia, CUDA.

Abstract

The Smith-Waterman algorithm is one of the most important algorithms in Bioinformatics. This algorithm performs local alignment of biological sequences, which enables an in-depth understanding of the biological functions, mutation recognition and genealogy specification. The goal of this diploma thesis is to exploit the parallelization opportunities of the Smith-Waterman algorithm for implementing high performance versions for emerging manycore processors, namely GPUs. Graphics processors are the upcoming solution for high performance systems; in this thesis, we perform an analysis of their architecture and present techniques for their efficient utilization. We have determined the performance-limiting factors of the algorithm and implemented a variety of optimizations. The analysis of the experimental results has let us to draw significant conclusions and propose directions for future research toward the optimization of the Smith-Waterman algorithm.

Keywords

Smith-Waterman algorithm, local sequence alignment, Bioinformatics, dynamic programming, parallel architecture, multicore processor, graphics processor, GPGPU, Nvidia, CUDA.

Ευχαριστίες

Η διπλωματική εργασία αυτή πραγματοποιήθηκε στο Εργαστήριο Υπολογιστικών συστημάτων της Σχολής Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Εθνικού Μετσόβιου Πολυτεχνείου, υπό την επίβλεψη του Αναπληρωτή Καθηγητή Νεκτάριου Κοζύρη.

Καταρχήν θα ήθελα να ευχαριστήσω τον καθηγητή μου κ. Νεκτάριο Κοζύρη, για την εποπτεία του κατά την εκπόνηση της εργασίας μου, αλλά και την συμβολή του στην διαμόρφωσή μου ως μηχανικού από το μάθημά του και τις πολύτιμες συμβουλές του.

Επίσης θα ήθελα να εκφράσω τις ευχαριστίες μου σε όλα τα μέλη του εργαστηρίου, και ιδιαίτερα στον Λέκτορα Γιώργο Γκούμα για την συνεχή καθοδήγησή.

Ακόμα οφείλω ένα πολύ μεγάλο ευχαριστώ στον υποψήφιο διδάκτορα Βασίλη Καρακάση, για όλες τις γνώσεις και την καθοδήγησή που μου προσέφερε, την αστείρευτη υπομονή του και γενικότερα τη σημαντική συμβολή του στην ολοκλήρωση αυτής της εργασίας.

Θα ήθελα να ευχαριστήσω το οικογενειακό μου περιβάλλον και ιδιαίτερα τους γονείς μου, Σπύρο Κούτσικο και Δήμητρα Κούτσικου για την συνολική υποστήριξη τους καθ' όλη τη διάρκεια της ζωής μου.

Τέλος θα ήθελα να ευχαριστήσω τους φίλους μου και ιδιαίτερα την κοπέλα μου Νόρα Καρατσικάκη για την ψυχολογική υποστήριξη που μου προσέφεραν.

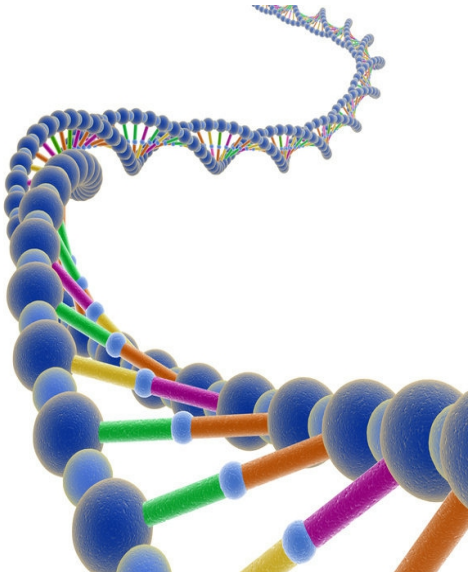
Περιεχόμενα

ΚΕΦΑΛΑΙΟ 1

ΕΙΣΑΓΩΓΗ

1.1 Ανακεφαλαίωση της μοριακής βιολογίας

Όλοι οι ζώντες οργανισμοί, από την πιο απλή μορφή τους, όπως είναι τα βακτήρια, ως και την πιο πολύπλοκη, όπως είναι ο άνθρωπος, έχουν σαν κύριο κοινό χαρακτηριστικό των κυττάρων τους το γενετικό κώδικα (DNA). Το DNA είναι ένα νουκλεϊκό οξύ που περιέχει τις γενετικές πληροφορίες που καθορίζουν τη βιολογική ανάπτυξη όλων των κυτταρικών μορφών ζωής. Το DNA έχει τη μορφή της διπλής έλικας, που η ανακάλυψή της πραγματοποιήθηκε το 1953 από τους James Watson και Francis Crick. Από πολλούς η διπλή έλικα του DNA θεωρείται η πιο σημαντική βιολογική ανακάλυψη του 20ου αιώνα.



Η δομή του DNA

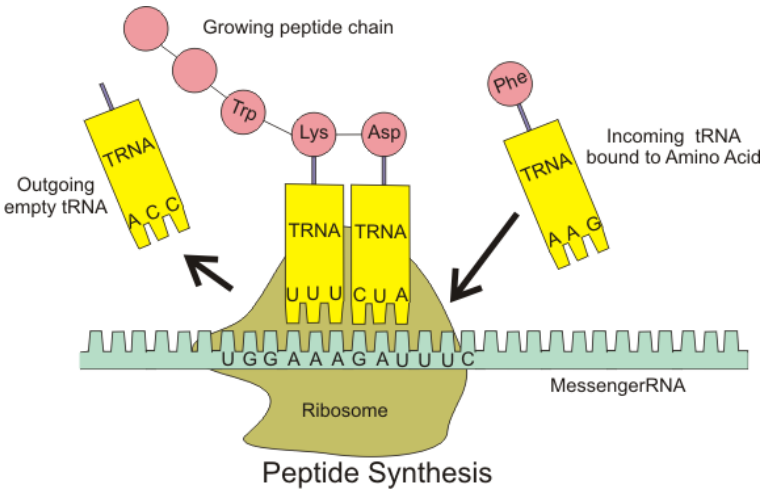
Πρόκειται για μια μεγαλομοριακή ένωση που συγκροτείται από αζωτούχες-πρωτεϊνικές βάσεις, φωσφορικές ρίζες και ένα σάκχαρο με πέντε άτομα άνθρακα (πεντόζη), την δε(σ)οξυριβόζη. Το σύνολο των μορίων DNA που υπάρχουν σε ένα κύτταρο αποτελούν το γενετικό υλικό του. Το DNA είναι ο φορέας των γενετικών πληροφοριών του κυττάρου, όχι

μόνον με την έννοια της μεταβίβασης χαρακτηριστικών, αναλλοίωτων από γενεά σε γενεά, αλλά και της ρύθμισης της φυσιογνωμίας εξειδίκευσης κάθε κυττάρου για την επιτέλεση των ιδιαίτερων λειτουργιών του.

Η διαμόρφωση των μεγάλων μορίων του DNA στο χώρο έχει τη μορφή δύο επιμήκων αλυσίδων, οι οποίες συστρέφονται ελικοειδώς μεταξύ τους. Οι αζωτούχες βάσεις στο DNA είναι τέσσερις:

- κυτοσίνη C,
- γουανίνη G,
- θυμίνη T,
- αδενίνη A.

Οι αζωτούχες βάσεις, ανάλογα με την σειρά αλληλουχίας τους σε τριάδες, κωδικοποιούν το μήνυμα για τη σύνθεση των αμινοξέων του κυττάρου στα ριβοσώματα. Εκεί τα αμινοξέα συνδυάζονται, με τη σειρά κατά την οποία μεταφέρθηκαν στο ριβόσωμα και συντίθενται έτσι οι διαφορετικές πρωτεΐνες.



Η σύνθεση πρωτεϊνών στα ριβοσώματα

Οι πρωτεΐνες αποτελούν τα πιο διαδεδομένα και πολυδιάστατα τόσο στη μορφή όσο και στη λειτουργία τους μακρομόρια. Ακόμη και σ΄ ένα απλό κύτταρο των βακτηρίων εντοπίζονται εκατοντάδες διαφορετικές πρωτεΐνες που κάθε μια εξ' αυτών έχει ιδιαίτερο ρόλο. Οι πρωτεΐνες αποτελούν είτε δομικά συστατικά των μεμβρανών του κυττάρου είτε συνεργούν σε κάποια συγκεκριμένη λειτουργία, όπως η δημιουργία πρωτεϊνικών συμπλόκων. Η ακολουθία αμινοξέων σε μια πρωτεΐνη καθορίζεται από ένα γονίδιο και κωδικοποιείται κατά τον

γενετικό κώδικα DNA. Ο γενετικός κώδικας κωδικοποιεί 20 αμινοξέα τα οποία φαίνονται στον παρακάτω πίνακα:

Διεθνής σύντμηση	Ελληνική ονομασία	Διεθνής σύντμηση	Ελληνική ονομασία
Ala	Αλανίνη	Leu	Λευκίνη
Arg	Αργινίνη	Lys	Λυσίνη
Asn	Ασπαραγίνη	Met	Μεθειονίνη
Asp	Ασπαραγινικό οξύ	Phe	Φαινυλαλανίνη
Cys	Κυστείνη	Pro	Προλίνη
Gln	Γλουταμίνη	Ser	Σερίνη
Glu	Γλουταμινικό οξύ	Thr	Θρεονίνη
Gly	Γλυκίνη	Trp	Θρυπτοφάνη
His	Ιστιδίνη	Tyr	Τυροσίνη
Ile	Ισολευκίνη	Val	Βαλίνη

Τα 20 αμινοξέα

Οι διάφορες λειτουργίες που παρατηρούνται στους οργανισμούς γίνονται χάρη στις πρωτεΐνες. Ο βιολογικός τους ρόλος καθορίζεται κάθε φορά από την τρισδιάστατη δομή τους που είναι συνέπεια της αλληλουχίας των αμινοξέων, η οποία και ξεκινά από την πρωτοταγή δομή, για την οποία καθοριστικοί παράγοντες είναι τα νουκλεϊκά οξέα (DNA, RNA). Η γνώση της πρωτοταγούς δομής μιας πρωτεΐνης, δηλαδή η αλληλουχία των αμινοξέων, παρέχει χρήσιμα στοιχεία για τη λειτουργία και την εξέλιξη μιας πρωτεΐνης, και κατά συνέπεια του οργανισμού.

1.2 Ο τομέας της Βιοπληροφορικής

Η Βιοπληροφορική είναι ένας επιστημονικός κλάδος που προέκυψε από την συνεργασία των επιστημών της μοριακής βιολογίας και της πληροφορικής. Ασχολείται με την μελέτη των μεθόδων για ανάκτηση, ανάλυση και αποθήκευση βιολογικών δεδομένων, όπως είναι τα νουκλεϊκά οξέα (DNA/RNA) και οι ακολουθίες πρωτεϊνών. Με τις μεθόδους που εφαρμόζει

παράγει νέα γνώση, η οποία είναι πολύ σημαντική σε άλλους τομείς όπως η Ιατρική και η Φαρμακολογία.

Ο πρωταρχικός στόχος της Βιοπληροφορικής είναι να βοηθήσει στην κατανόηση των βιολογικών διαδικασιών. Τα βιολογικά δεδομένα μπορούν εύκολα να απεικονιστούν σαν ψηφιακή πληροφορία. Για παράδειγμα, τα βιολογικά μακρομόρια που αναφέρονται πιο πάνω, μπορούν να αναπαρασταθούν ως μεγάλες ακολουθίες συμβόλων. Με αυτόν τον τρόπο δίνεται η δυνατότητα να εφαρμοστούν αλγόριθμοι, οι οποίοι χρησιμοποιώντας την υπολογιστική ισχύ που είναι πλέον ευρέως διαθέσιμη, επεξεργάζονται τα δεδομένα αυτά και παράγουν χρήσιμα αποτελέσματα. Επιστήμονες από το κλάδο της πληροφορικής σε συνεργασία με επιστήμονες Βιολόγους, συνεργάζονται μέσω του προγραμματισμού δημιουργώντας νέες δομές και διαδικασίες από τις οποίες οι εξαγόμενες πληροφορίες χρησιμοποιούνται στο πεδίο της Βιοπληροφορικής

Τα βιολογικά δεδομένα πλέον αποθηκεύονται σε βάσεις δεδομένων, των οποίων το μέγεθος αυξάνεται ραγδαία ημέρα με την ημέρα. Όλος αυτός ο όγκος δεδομένων πρέπει να αναλύεται καθημερινά. Μια από τις κυριότερες εκφάνσεις της Βιοπληροφορικής είναι η επεξεργασία αυτών των βάσεων δεδομένων, με την χρήση προγραμμάτων που - όπως αναφέραμε και πιο πάνω - έχουν δημιουργηθεί από πληροφορικούς και βιολόγους. Έτσι καθημερινά αναλύονται βιολογικές ακολουθίες πάνω από 260.000 οργανισμών, αριθμός που συνολικά αντιστοιχεί σε πάνω από 190 δισεκατομμύρια νουκλεοτίδια Ο ρόλος τους είναι να ψάχνουν για μεταλλάξεις του γενετικού υλικού, να αναγνωρίζουν ακολουθίες που μοιάζουν με απώτερο σκοπό την αναγνώριση κοινών χαρακτηριστικών που πιθανώς να καταλήξει στην εξιχνίαση αγνώστων βιολογικών λειτουργιών.

Ο αλγόριθμος Smith-Waterman που εξετάζεται σε αυτή την διπλωματική εργασία έχει αποδειχτεί βασικό εργαλείο για την ανάλυση ακολουθιών και συγκεκριμένα για την στοίχιση τους. Η στοίχιση των βιολογικών ακολουθιών δίνει την δυνατότητα αναγνώρισης των όμοιων περιοχών τους, κάτι που έχει ιδιαίτερη σημασία αφού οι κοινές αυτές περιοχές μπορεί να μοιράζονται λειτουργικά χαρακτηριστικά. Επίσης, η ομοιότητα που παρατηρείται είναι ένδειξη της πιθανότητας ύπαρξης κοινού προγόνου και οι σημερινές τους διαφορές να είναι αποτέλεσμα μεταλλάξεων που έλαβαν τόπο λόγω της εξέλιξης. Το να βρεθεί κοινός πρόγονος μεταξύ δύο οργανισμών δίνει την δυνατότητα να εφαρμοστεί η γνώση που έχουμε για τον έναν, στον άλλον οργανισμό. Ακόμα, η στοίχιση ακολουθιών βοηθάει να καθοριστεί η

καταγωγή άγνωστων έως τώρα βιολογικών ακολουθιών. Τέλος, ο συνδυασμός όλης της παραπάνω πληροφορίας καθιστά δυνατή την κατασκευή εξελικτικών δέντρων, που ξεδιαλύνουν τις σχέσεις μεταξύ των διαφόρων ειδών και την πορεία της εξέλιξης.

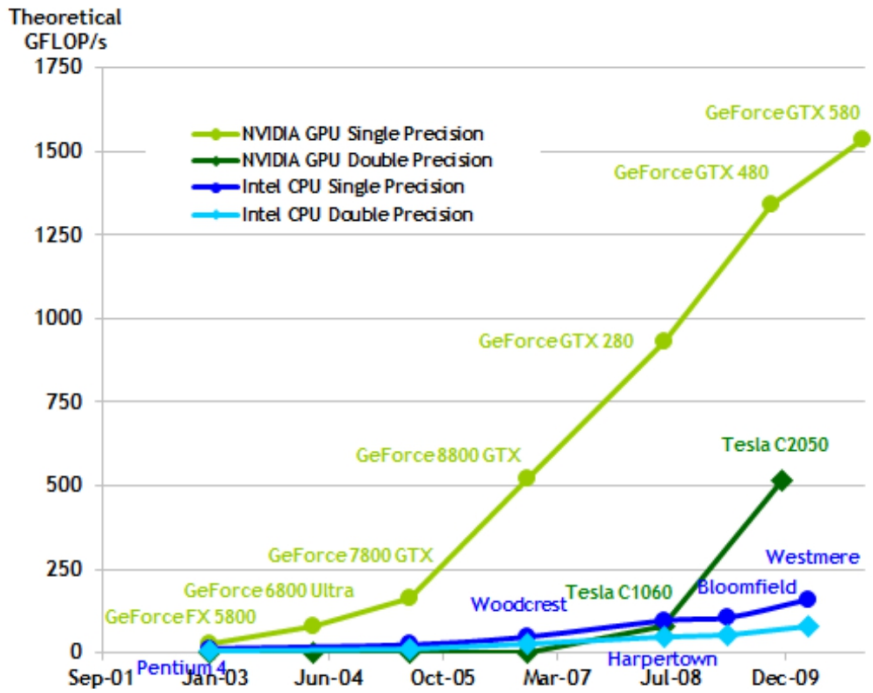
Όλα τα παραπάνω στοιχεία δίνουν το έναυσμα για περαιτέρω μελέτη και ανάλυση του αλγορίθμου Smith-Waterman με απώτερο σκοπό την βελτιστοποίηση ενός σημαντικού εργαλείου.

1.3 Ο ρόλος των επεξεργαστών γραφικών

Οι απαιτήσεις επεξεργαστικής ισχύος αυξάνονται καθημερινά. Όχι για τον μέσο χρήστη, που η υπάρχουσα επεξεργαστική ισχύς καλύπτει τις ανάγκες του, αλλά για την επιστημονική κοινότητα, που στηρίζεται ολοένα και περισσότερο στους υπολογιστές. Τα τελευταία χρόνια παρατηρείται μια προσπάθεια από εταιρίες εκτός του "κλασσικού" χώρου των επεξεργαστών (CPU) να εισέλθουν στον χώρο του High Performance Computing – HPC, δηλαδή του τομέα που χρησιμοποιεί υπερυπολογιστές (supercomputers) και συστοιχίες υπολογιστών (computer clusters) για την επίλυση επιστημονικών υπολογιστικών προβλημάτων. Από την μία, οι εκπρόσωποι των εμπορικών CPU έχουν καταφέρει να ενσωματώσουν πολλούς επεξεργαστικούς πυρήνες (cores) υψηλής συχνότητας σε έναν επεξεργαστή (6 cores με clock μεγαλύτερου των 3 Ghz). Αυτοί είναι οι λίγο πολύ γνωστοί σε όλους multi-cores CPU και γίνεται προσπάθεια να γίνει μετάβαση σε many-cores. Από την άλλη, έχουν εμφανιστεί κάποιες υβριδικές προτάσεις, όπως ο επεξεργαστής Cell της IBM ο οποίος έχει 1 Power Processor Element - PPE, ο οποιος είναι ουσιαστικά ένας multi-thread 64-bit PowerPC επεξεργαστής και 8 Synergistic Processing Elements – SPE, τα οποία ειναι 128-bit SIMD RISC επεξεργαστικές μονάδες.

Η εξέλιξη που έχει συγκεντρώσει το ενδιαφέρον αρκετά μεγάλου μέρους του επιστημονικού κόσμου είναι η χρησιμοποίηση επεξεργαστών γραφικών - GPU για την εκτέλεση υπολογισμών τους οποίους παραδοσιακά αναλαμβάνουν οι CPU. Αυτό είναι γνωστό ως

General Purpose computation on Graphics Processing Units και εν συντομία GPGPU ή GPGP ή GP[2]. Ο λόγος που δημιουργήθηκε και εξελίχθηκε η τάση αυτή, είναι ότι η επεξεργαστική ισχύς των GPU έχει αυξηθεί σε τόσο μεγάλο βαθμό, που έχουν την δυνατότητα να αποδώσουν ρεαλιστικά, υψηλής ανάλυσης 3D γραφικά σε πραγματικό χρόνο. Για να δώσουμε μια ίδεα για την διαφορά που υπάρχει σε υπολογιστική δύναμη, ο γρηγορότερος επεξεργαστής της Intel, ο hexacore Hyper-threaded i7 3930K έχει μέγιστη επεξεργαστική απόδοση 182 GFLOPS, ενώ η γρηγορότερη κάρτα γραφικών της Nvidia (με έναν επεξεργαστή), η GTX 680 έχει μέγιστη επεξεργαστική απόδοση 3090 GFLOPS. Η εικόνα του χάσματος επεξεργαστικής ισχύος, απεικονίζεται επιτυχώς στο επόμενο διάγραμμα.



Μέγιστη θεωρητική επίδοση σε CPUs και GPUs

Όμως, οι επεξεργαστές γραφικών δεν μπορούσαν να χρησιμοποιηθούν για υπολογισμούς γενικού σκοπού, με τις παραδοσιακές δυνατότητες που είχαν μέχρι πριν από λίγα χρόνια, οι οποίες ήταν αρκετά περιορισμένες. Το κύριο βήμα που έπρεπε να γίνει προς αυτήν την κατεύθυνση είναι οι προγραμματιζόμενοι Vertex Shaders, κάτι το οποίο έγινε για 1η φορά στον επεξεργαστή γραφικών G80 της Nvidia, η οποία αποτελεί το πρώτο δείγμα της αρχιτεκτονικής ενοποιημένων Shaders – Unified Shader Architecture. Την ίδια πορεία ακολούθησε και η ATI (πλέον AMD).

Έχοντας εκπληρώσει τις προϋποθέσεις από πλευράς hardware, αυτό που έμενε ήταν να

υλοποιηθούν τα κατάλληλα προγραμματιστικά εργαλεία, έτσι ώστε να δωθεί στους προγραμματιστές η δυνατότητα να εκμεταλλευτούν τις δυνατότητες των GPU. Έχουν γίνει αρκετές προσπάθειες, αυτές όμως που επικρατούν, φαίνεται να είναι το CUDA SDK της Nvidia και το Stream SDK της AMD, το οποίο παλιότερα (επί ATI εποχής) λεγόταν CTM – Close To Metal και αποτελούσε έναν low-level τρόπο προγραμματισμού των GPU.

Κάπου έδω έρχεται να προστεθεί η Open Computing Language – OpenCL, η οποία είναι μια γλώσσα, ή καλύτερα ένα framework για την δημιουργία προγραμμάτων που εκτελούνται σε ετερογενείς αρχιτεκτονικες. Συγκεκριμένα, η OpenCL περιλαμβάνει μια γλώσσα βασισμένη πάνω στην C99 για την συγγραφή πυρήνων / kernels που εκτελούνται σε OpenCL συσκευές (στην περίπτωσή μας, κάρτες γραφικών). Επίσης περιλαμβάνει APIs που χρησιμοποιούνται για τον ορισμό και τον έλεγχο της ετερογενής πλατφόρμας. Παρέχει δυνατότητες παράλληλων υπολογισμών είτε task-based είτε data-based. Οι κύριοι ανταγωνιστές της OpenCL είναι το CUDA της Nvidia και το DirectCompute της Microsoft.

Η OpenCL βρίσκεται υπό την διαχείρηση του Khronos Group, που αποτελεί μια μη κερδοσκοπική κοινοπραξία. Αρχικά αναπτύχθηκε απο την Apple, η οποία πρότεινε και την συνεργασία με AMD, IBM, Intel και Nvidia, οι οποίοι συγκρότησαν το Khronos Group. Λογικό είναι να αναρωτίεται κάποιος για τον λόγο που ανταγωνιστικές εταιρίες, όπως η Nvidia και η AMD, ενώ είχαν τα δικά τους APIs, συνεργάστηκαν (και μάλιστα η AMD εγκατέλειψε το δικό της CTM) για την εξέλιξη της OpenCL. Ο λόγος είναι ότι αν η κάθε εταιρία του χώρου έχει το δικό της framework και στηρίζει μόνο αυτό, τότε οι προγραμματιστές μοιράζονται, όπως και οι εμπορικές και επιστημονικές εφαρμογές κάτι το οποίο δεν βοηθάει την εξέλιξη. Χρειάζεται λοιπόν, όχι μόνο στο χώρο του GPU computing αλλά και γενικότερα να υπάρχει κάποιο standard, μέσω του οποίου θα μπορούν να μιλάνε όλοι.

ΚΕΦΑΛΑΙΟ 2

Ο ΑΛΓΟΡΙΘΜΟΣ SMITH - WATERMAN

2.1 Στοίχιση Ακολουθιών

Στη Βιοπληροφορική, η στοίχιση ακολουθιών (sequence alignment) είναι μία διαδικασία κατά την οποία δύο ακολουθίες DNA, RNA ή πρωτεϊνών ταξινομούνται με σκοπό την αναγνώριση περιοχών ομοιότητας. Οι ομοιότητες μεταξύ βιολογικών ακολουθιών είναι συνέπεια λειτουργικών, δομικών ή εξελικτικών σχέσεων και η αναγνώρισή τους είναι μια εργασία υψηλής σημασίας, καθώς μπορεί να οδηγήσει στην πλήρη κατανόηση των διαφόρων βιολογικών μηχανισμών. Συνήθως οι στοιχισμένες ακολουθίες νουκλεοτιδίων ή αμινοξέων αναπαριστώνται σαν γραμμές ενός πίνακα με κενά μεταξύ των στοιχείων, έτσι ώστε ίδια ή παραπλήσια στοιχεία να είναι στην ίδια στήλη.

Σκοπός της στοίχισης ακολουθιών είναι η τοποθέτηση των δύο υπό εξέταση ακολουθιών η μία κάτω από την άλλη με τέτοιον τρόπο έτσι ώστε να ταιριάζουν όσο τον δυνατόν περισσότερο. Για να γίνει αυτό, χρησιμοποιείται κάποιο μοτίβο βαθμολογίας, και η βέλτιστη στοίχιση είναι αυτή που συγκεντρώνει την υψηλότερη βαθμολογία. Για παράδειγμα, κάθε ζεύγος συμβόλων που ταιριάζει, βαθμολογείται με κάποιο θετικό σκορ (π.χ. +3), ενώ κάθε ζεύγος που διαφέρει βαθμολογείται με κάποιο αρνητικό σκορ (π.χ. -2). Αν απαιτείται να εισαχθεί κάποιο κενό έτσι ώστε οι ακολουθίες να ταιριάζουν σε άλλα σημεία, το κάθε κενό βαθμολογείται πάλι με κάποιο αρνητικό σκορ (π.χ. -1). Το άθροισμα όλων των επιμέρους

σκορ, μας δίνει την συνολική βαθμολογία της ταξινόμησης. Ελέγχοντας όλες τις δυνατές ταξινομήσεις και επιλέγοντας αυτήν με την μεγαλύτερη βαθμολογία, παίρνουμε την βέλτιστη στοίχιση.

Η στοίχιση νουκλεοτιδίων μπορεί να δώσει πολύ σημαντικές πληροφορίες για τις αιτίες και τα αποτελέσματα των μεταλλάξεων. Για παράδειγμα, όταν δύο στοιχισμένες ακολουθίες DNA έχουν κοινό πρόγονο, οι διαφορές ερμηνεύονται σαν σημειακές μεταλλάξεις (αλλάζει μόνο μία βάση νεοκλεοτιδίου), ενώ τα κενά σαν μεταλλάξεις προσθήκης ή διαγραφής. Οι σημειακές μεταλλάξεις συμβαίνουν κατά την αντιγραφή του DNA και επηρεάζονται από τις περιβαλλοντικές συνθήκες. Ενώ παλιότερα υπήρχε η πεποίθηση ότι αυτές οι μεταλλάξεις γίνονται τυχαία, πρόσφατες έρευνες έχουν δείξει ότι οι σημειακές μεταλλάξεις συμβαίνουν για να ανταποκριθεί ο οργανισμός στις περιβαλλοντικές αλλαγές. Αντίθετα, οι μεταλλάξεις προσθήκης ή διαγραφής συμβαίνουν στην κωδική περιοχή του mRNA και έχουν σαν πιθανό αποτέλεσμα την τελείως διαφορετική μετάφρασή του, πράγμα που είναι μια συνήθης αιτία γενετικών ασθενειών.

Αντίστοιχα, στη στοίχιση πρωτεϊνών ο υψηλός βαθμός ομοιότητας συγκεκριμένων περιοχών αμινοξέων (ακολουθιακών μοτίβων) είναι μέτρο για την δομική και λειτουργική σημασία των περιοχών αυτών. Αυτό συμβαίνει, γιατί η διατήρηση της περιοχής αυτής στο βάθος της εξέλιξης των διαφόρων ειδών, με καθόλου ή ελάχιστες αλλαγές (αλλαγή λίγων αμινοξέων με άλλα παραπλήσιων βιοχημικών ιδιοτήτων) υποδεικνύει ότι τα τμήματα αυτά έχουν υψηλή βιολογική σημασία, και μεταλλάξεις σε μια τέτοια περιοχή μπορούν να οδηγήσουν σε μία μη βιώσιμη μορφή ζωής.

Πολύ μικρές ή πολύ παρόμοιες ακολουθίες μπορούν να στοιχιστούν με το χέρι. Όμως στην πράξη δεν υπάρχουν τέτοιες ακολουθίες. Τα σημαντικά προβλήματα απαιτούν στοίχιση μεγάλου αριθμού ακολουθιών, οι οποίες έχουν μεγάλο μήκος και μεγάλη μεταβλητότητα. Έτσι, η ανθρώπινη γνώση έχει χρησιμοποιηθεί για να κατασκευαστούν αλγόριθμοι οι οποίοι παράγουν στοιχίσεις υψηλής ακρίβειας.

Εκτός της Βιοπληροφορικής, στοίχιση ακολουθιών χρησιμοποιείται και σε άλλους τομείς, όπως για παράδειγμα στην επεξεργασία φυσικής γλώσσας και κατ' επέκταση στην μηχανική μετάφραση. Συγκεκριμένα, τεχνικές στοίχισης ακολουθιών χρησιμοποιούνται για να δημιουργήσουν τα σύνολα από τα οποία οι αλγόριθμοι που παράγουν φυσική γλώσσα θα

επιλέξουν τις λέξεις που θα χρησιμοποιήσουν. Επίσης, στοίχιση ακολουθιών αρκετές φορές χρησιμοποιείται και στην επεξεργασία οικονομικών δεδομένων

```
AAB24882        TYHMCQFHCRYVNNHSGEKLYECNERSKAFSCPSHLQCHKRRQIGEKTHEHNQCGKAFPT 60
AAB24881        --------------------YECNQCGKAFAQHSSLKCHYRTHIGEKPYECNQCGKAFSK 40
                                    ****: .***:  * *:** * :****.:* *******..

AAB24882        PSHLQYHERTHTGEKPYECHQCGQAFKKCSLLQRHKRTHTGEKPYE-CNQCGKAFAQ- 116
AAB24881        HSHLQCHKRTHTGEKPYECNQCGKAFSQHGLLQRHKRTHTGEKPYMNVINMVKPLHNS 98
                 **** *:***********:***:**.: .***************    :  *.: :
```

Παραδείγματα στοίχισης ακολουθιών

2.2 Είδη στοίχισης ακολουθιών

Οι υπολογιστικές μέθοδοι στοίχισης ακολουθιών συνήθως χωρίζονται σε δύο κατηγορίες, ανάλογα με το είδος στοίχισης που χρησιμοποιούν. Συγκεκριμένα, υπάρχει η ολική στοίχιση (global alignment) και η τοπική στοίχιση (local alignment). Στη συνέχεια παρουσιάζουμε τις διαφορές των δύο αυτών ειδών στοίχισης.

2.2.1 Ολική στοίχιση ακολουθιών

Στην ολική στοίχιση ακολουθιών (global sequence alignment) οι δύο ακολουθίες στοιχίζονται σε όλο το μήκος με τον καλύτερο δυνατό τρόπο. Θα μπορούσαμε να πούμε ότι γίνεται προσπάθεια να εξαπλωθεί η αντιστοίχιση σε όλο το μήκος των ακολουθιών. Κάθε σύμβολο της κάθε ακολουθίας αντιστοιχίζεται είτε σε ένα σύμβολο της άλλης, είτε σε ένα κενό. Η μέθοδος αυτή είναι χρήσιμη όταν οι ακολουθίες που στοιχίζονται είναι παραπλήσιες και έχουν σχεδόν ίδιο μήκος. Ο πιο γνωστός αλγόριθμος που εφαρμόζει ολική στοίχιση ακολουθιών είναι ο αλγόριθμος Needleman – Wunsh, ο οποίος βασίζεται στην τεχνική του δυναμικού προγραμματισμού.

2.2.2 Τοπική στοίχιση ακολουθιών

Η τοπική στοίχιση ακολουθιών (local sequence alignment) βρίσκει περιοχές που παρουσιάζουν αυξημένη ομοιότητα ή ακόμα και παρόμοια ακολουθιακά μοτίβα μέσα σε μεγάλες ακολουθίες που πιθανώς διαφέρουν κατά πολύ. Οι περιοχές που διαφέρουν πολύ μπορούν να μείνουν εκτός της στοίχισης και να συμπληρωθούν με κενά. Η τοπική στοίχιση συνήθως προτιμάται, είναι όμως αρκετά πιο δύσκολη να υπολογιστεί λόγω της επιπλέον προσπάθειας που γίνεται για να βρεθούν οι περιοχές ομοιότητας. Για τους λόγους αυτούς η τοπική στοίχιση έχει κεντρίσει το ενδιαφέρον της επιστημονικής κοινότητας και έχουν εφαρμοστεί πλήθος υπολογιστικών αλγορίθμων για την υλοποίησή της, συμπεριλαμβανομένων ευριστικών ή πιθανοτικών αλγορίθμων που όμως δεν μπορούν να εγγυηθούν ότι το αποτέλεσμα τους θα είναι το σωστό. Ο πιο γνωστός αλγόριθμος που εφαρμόζει τοπική στοίχιση ακολουθιών είναι ο αλγόριθμος Smith - Waterman, ο οποίος επίσης βασίζεται στην τεχνική του δυναμικού προγραμματισμού.

`FTFTALILLAVAV`	`FTFTALILL-AVAV`
`F--TAL-LLA-AV`	`--FTAL-LLAAV--`
Ολική στοίχιση	Τοπική στοίχιση

Παραδείγματα ολικής και τοπικής στοίχισης ακολουθιών

2.3 Ο αλγόριθμος Smith - Waterman

Ο αλγόριθμος των Smith-Waterman είναι πιο γνωστός αλγόριθμος που εφαρμόζει τοπική στοίχιση ακολουθιών (local sequence alignment) για να εντοπίσει παρόμοιες περιοχές μεταξύ νουκλεοτιδίων ή πρωτεϊνών. Για να το επιτύχει αυτό συγκρίνει τμήματα όλων των πιθανών μηκών και βελτιστοποιεί την βαθμολογία στοίχισης, που αποτελεί μέτρο για την ομοιότητα των ακολουθιών. Ο αλγόριθμος αυτός προτάθηκε από τους Temple F. Smith και Michael S. Waterman το 1981 και βελτιστοποιήθηκε από τον Gotoh το 1982. Είναι παραλλαγή του αλγορίθμου Needleman-Wunch με την διαφορά ότι δεν χρησιμοποιεί αρνητικές βαθμολογίες

(όταν ένα ζεύγος έχει αρνητική βαθμολογία, τότε αυτή τίθεται ίση με μηδέν). Με τον τρόπο αυτό, οι τοπικές στοιχίσεις μπορούν να παρατηρηθούν. Ο αλγόριθμος Smith-Waterman χρησιμοποιεί την τεχνική του δυναμικού προγραμματισμού (dynamic programming). Για να υπολογίσει την βέλτιστη τοπική στοίχιση κάνει χρήση ενός πίνακα αντικατάστασης (substitution matrix) και ενός συστήματος βαθμολογίας κενών (gap-scoring scheme).

2.3.1 Δυναμικός προγραμματισμός

Ο δυναμικός προγραμματισμός (dynamic programming) είναι μια μέθοδος επίλυσης περίπλοκων προβλημάτων χωρίζοντας το πρόβλημα σε μικρότερα και απλούστερα υποπροβλήματα τα οποία είναι επικαλυπτόμενα. Αυτό σημαίνει ότι αν συνδυαστούν οι λύσεις των μικρών υποπροβλημάτων δίνουν την λύση μεγαλύτερων υποπροβλημάτων, κάτι που αν συνεχιστεί μας δίνει την λύση συνολικού προβλήματος. Η ιδέα του δυναμικού προγραμματισμού είναι απλή: προσπαθεί να λύσει το κάθε υποπρόβλημα μόνο μία φορά και να αποθηκεύσει το αποτέλεσμα. Κάθε φορά που θα χρειαστεί το αποτέλεσμα αυτού του υποπροβλήματος, τότε απλά θα το αναζητήσει και δεν θα χρειαστεί να το υπολογίσει ξανά. Η τεχνική αυτή είναι ιδιαίτερα χρήσιμη όταν ο αριθμός των επαναλαμβανόμενων υποπροβλημάτων αυξάνεται εκθετικά σε σχέση με το μέγεθος του προβλήματος.

Για να μπορεί να εφαρμοστεί η τεχνική του δυναμικού προγραμματισμού, πρέπει να ισχύει η αρχή της βελτιστότητας. Αυτό σημαίνει ότι κάθε τμήμα της βέλτιστης λύσης αποτελεί την βέλτιστη λύση για το αντίστοιχο υποπρόβλημα. Με τον τρόπο αυτό, υπολογίζονται αναδρομικά οι επιμέρους λύσεις και συνδυάζονται για να δώσουν την βέλτιστη λύση για το συνολικό πρόβλημα.

Ο δυναμικός προγραμματισμός χρησιμοποιείται στον αλγόριθμο Smith-Waterman. Όπως αναφέρθηκε και πιο πάνω, ο αλγόριθμος αυτός συγκρίνει τμήματα όλων των πιθανών μηκών από τις ακολουθίες που προσπαθεί να στοιχίσει. Έτσι οι στοιχίσεις των μικρών επιμέρους τμημάτων χρησιμοποιούνται για να υπολογιστούν στοιχίσεις μεγαλύτερων τμημάτων, μέχρις ότου να καταλήξει στο πλήρες μήκος των ακολουθιών.

2.3.2 Πίνακας αντικατάστασης

Λόγω του φαινομένου της εξέλιξης των ειδών, οι αλληλουχίες αμινοξέων των πρωτεϊνών αλλάζουν σταδιακά από γενιά σε γενιά, λόγω των μεταλλάξεων που υφίσταται το DNA. Κάθε αμινοξύ έχει περισσότερες ή λιγότερες πιθανότητες να μετατραπεί σε κάποιο άλλο. Για παράδειγμα, τα υδρόφιλα αμινοξέα έχουν περισσότερες πιθανότητες να μετατραπούν σε κάποιο άλλο υδρόφιλο αμινοξύ, παρά σε κάποιο υδρόφοβο.

Οι πίνακες αντικατάστασης (substitution matrices) περιγράφουν τη πιθανότητα (ή τον ρυθμό) με την οποία ο χαρακτήρας μιας ακολουθίας μπορεί να μετατραπεί σε έναν άλλον, με την πάροδο του χρόνου. Έτσι για τα αμινοξέα κατασκευάζεται ένας πίνακας 20 x 20 του οποίου το (i, j) στοιχείο εκφάζει την πιθανότητα του i-οστου αμινοξέος να μετατραπεί στον j-οστό ανιμοξύ. Ο πίνακας αυτός ονομάζεται πίνακας αντικατάστασης. Υπάρχουν διάφορα είδη τέτοιων πινάκων, με τους πιο διαδεδομένους να είναι οι PAM και BLOSUM.

Οι πίνακες PAM (Point Accepted Mutation) προτάθηκαν από την Margaret Dayhoff το 1978 και βασίζονται σε 1572 μεταλλάξεις σε 71 οικογένειες σχετικών μεταξύ τους πρωτεϊνών. Υπάρχει μεγάλος αριθμός πινάκων PAM, όμως στην πράξη συνήθως χρησιμοποιούνται οι πίνακες PAM30 και PAM70. Ο αριθμός που συνοδεύει κάθε πίνακα PAM αναφέρεται στον αριθμό των παρατηρούμενων μεταλλάξεων. Για παράδειγμα ο πίνακας PAM1 δίνει τις πιθανότητες αντικατάστασης για ακολουθίες που παρουσιάζουν 1 μετάλλαξη κάθε 100 αμινοξέα. Οι μεταλλάξεις μπορούν και να επικαλύπτονται και για αυτό υπάρχει μέχρι και πίνακας PAM250, που σημαίνει ότι αναφέρεται σε ακολουθίες που παρουσιάζουν 250 μεταλλάξεις κάθε 100 αμινοξέα. Ο κάθε πίνακας PAM υπολογίζεται από τον ακόλουθο τύπο:

$$\mathrm{PAM}_n(i,j) = log\frac{f(j)M^n(i,j)}{f(i)f(j)} = log\frac{M^n(i,j)}{f(i)}$$

όπου M(i, j) είναι η πιθανότητα του αμινοξέος i να μετατραπεί στο αμινοξύ j κατά τη διάρκεια μιας μετάλαξης, ενώ f(i) είναι η συχνότητα των του αμινοξέος i.

Η μεθοδολογία που ακολουθήθηκε για τους πίνακες PAM δεν λειτούργησε πολύ καλά στην πράξη, ιδιαίτερα όταν γίνεται στοίχιση ακολουθιών με μεγάλες διαφορές. Για το λόγο οι Henikoff και Henikoff πρότειναν το 1992 τους πίνακες BLOSUM (BLOcks of amino acid SUbstitution Matrix). Για την κατασκευή των πινάκων αυτών χρησιμοποιήθηκαν στοιχίσεις πρωτεϊνών που εξελικτικά διαφέρουν πολύ. Οι πιθανότητες που χρησιμοποιήθηκαν για τον

δημιουργία των πινάκων αυτών υπολογίστηκαν σε διάφορα μπλοκ ακολουθιών διαφόρων στοιχίσεων πρωτεϊνών. Τα μπλοκ αυτά ορίζονται από ένα όριο που χρησιμοποιείται για την κατασκευή του κάθε πίνακα και εκφράζει το ποσοστό ομοιότητας των πρωτεϊνών. Για τον BLOSUM62 το όριο αυτό είναι το 62% και πρακτικά αυτό σημαίνει ότι όταν κάποιες πρωτεΐνες είναι κατά το 62% ίδιες, συμβάλουν στην ίδια θέση του πίνακα. Αυτό το όριο χρησιμοποιείται για να μειωθεί η επιρροή παραπλήσιων πρωτεϊνών. Έτσι, ο πίνακας BLOSUM45 έχει χαμηλότερο όριο ομοιότητας (45%) και για αυτό το λόγο χρησιμοποιείται για την στοίχιση ακολουθιών που διαφέρουν πολύ, ενώ ο BLOSUM80 για παραπλήσιες ακολουθίες. Για τον υπολογισμό του κάθε πίνακα BLOSUM χρησιμοποιείται η ακόλουθη εξίσωση:

$$S_{ij} = \left(\frac{1}{\lambda}\right) \log\left(\frac{p_{ij}}{q_i * q_j}\right)$$

όπου p_{ij} είναι η πιθανότητα δύο αμινοξέων i και j να αντικαταστήσουν το ένα το άλλο σε μια ακολουθία, ενώ q_i και q_j είναι η πιθανότητα τα αμινοξέα i και j να βρεθούν στην ίδια πρωτεϊνή τυχαία.

Όπως έχει φανεί από την πράξη, ο πίκανας BLOSUM62 λειτουργεί εξαιρετικά για την εύρεση ομοιοτήτων μέσω της στοίχισης ακολουθιών και για το λόγο αυτό είναι ο πιο ευρέως χρησιμοποιούμενος πίνακας αντικατάστασης. Πρόσφατα παρατηρήθηκε ότι ο BLOSUM62 που χρησιμοποιείται εκτενώς τα τελευταία χρόνια, δεν ακολουθεί πιστά τον αλγόριθμο που πρότειναν οι Henikoff και Henikoff, αλλά έχει γίνει κάποιο υπολογιστικό λάθος. Παρόλα αυτά, ο "εσφαλμένα υπολογισμένος" BLOSUM62 λειτουργεί άψογα προσφέροντας την καλύτερη απόδοση στις συγκρίσεις ακολουθιών.

Κάθε κελί του πίνακα αντικατάστασης είναι ένα σκορ της στοίχισης των δύο στοιχείων των ακολουθιών που στοιχίζονται. Όταν το σκορ αυτό είναι θετικό τότε υπάρχει ταίριασμα μεταξύ των δύο στοιχείων και το σκορ αυτό αναφέρεται και ως ανταμοιβή. Όσο καλύτερο είναι το ταίριασμα τόσο μεγαλύτερη είναι και η ανταμοιβή. Όταν το σκορ είναι αρνητικό τότε τα 2 στοιχεία που συγκρίνονται δεν ταιριάζουν, και αυτό αναφέρεται και ως ποινή. Και πάλι, όσο χειρότερο είναι το ταίριασμα, τόσο μεγαλύτερη θα είναι η ποινή.

Ala	4																			
Arg	-1	5																		
Asn	-2	0	6																	
Asp	-2	-2	1	6																
Cys	0	-3	-3	-3	9															
Gln	-1	1	0	0	-3	5														
Glu	-1	0	0	2	-4	2	5													
Gly	0	-2	0	-1	-3	-2	-2	6												
His	-2	0	1	-1	-3	0	0	-2	8											
Ile	-1	-3	-3	-3	-1	-3	-3	-4	-3	4										
Leu	-1	-2	-3	-4	-1	-2	-3	-4	-3	2	4									
Lys	-1	2	0	-1	-3	1	1	-2	-1	-3	-2	5								
Met	-1	-1	-2	-3	-1	0	-2	-3	-2	1	2	-1	5							
Phe	-2	-3	-3	-3	-2	-3	-3	-3	-1	0	0	-3	0	6						
Pro	-1	-2	-2	-1	-3	-1	-1	-2	-2	-3	-3	-1	-2	-4	7					
Ser	1	-1	1	0	-1	0	0	0	-1	-2	-2	0	-1	-2	-1	4				
Thr	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	1	5			
Trp	-3	-3	-4	-4	-2	-2	-3	-2	-2	-3	-2	-3	-1	1	-4	-3	-2	11		
Tyr	-2	-2	-2	-3	-2	-1	-2	-3	2	-1	-1	-2	-1	3	-3	-2	-2	2	7	
Val	0	-3	-3	-3	-1	-2	-2	-3	-3	3	1	-2	1	-1	-2	-2	0	-3	-1	4
	Ala	Arg	Asn	Asp	Cys	Gln	Glu	Gly	His	Ile	Leu	Lys	Met	Phe	Pro	Ser	Thr	Trp	Tyr	Val

Ο πίνακας BLOSUM62

Για να μπορούν οι υπολογιστικοί αλγόριθμοι που υλοποιούν στοίχιση ακολουθιών να λειτουργούν τόσο με πρωτεΐνες όσο και με ακολουθίες DNA, οι πίνακες αντικατάστασης επεκτείνονται, και προστίθονται σε αυτούς 4 γραμμές και 4 στήλες, καθεμιά για τις 4 αζωτούχες βάσεις του DNA.

2.3.3 Βαθμολογία κενών

Κατά την στοίχιση ακολουθιών σχεδόν πάντα επιβάλλεται η προσθήκη κάποιων κενών στις ακολουθίες, έτσι ώστε αυτές να στοιχιστούν με τον βέλτιστο τρόπο. Τα κενά αυτά βαθμολογούνται με κάποιο αρνητικό σκορ, κάτι το οποίο συνήθως αναφέρεται ως ποινή κενού (gap penalty). Η ποινή κενού συμβάλει στην τελική βαθμολογία της στοίχισης και για αυτό το λόγο η τιμή της θα πρέπει να είναι σε συμφωνία με τις τιμές του πίνακα αντικατάστασης που θα επιλεχθεί. Τελικά ο πίνακας αποκατάστασης και η βαθμολογία των κενών, είναι τα δύο στοιχεία που επηρεάζουν την βέλτιστη στοίχιση. Διαλέγοντας υψηλή

ποινή κενού έχει ως αποτέλεσμα την στοίχιση στοιχείων με μεγάλες διαφορές, έτσι ώστε να αποφευχθεί η δημιουργιά κάποιου κενού. Υπάρχουν διάφοροι τύποι ποινών κενού, οι οποίοι παρουσιάζονται στη συνέχεια:

- *Σταθερή ποινή κενών* (constant gap penalty). Αποτελεί τον πιο απλό τρόπο βαθμολόγησης κενών. Υπάρχει μία μόνο παράμετρος, έστω p, η οποία προστίθεται στη βαθμολογία της στοίχισης όταν ένα κενό δημιουργείται. Αυτό σημαίνει ότι κενά οσοδήποτε μήκους έχουν σταθερή ποινή p.
- *Γραμμική ποινή κενών* (linear gap penalty). Σε αυτή την περίπτωση υπάρχει πάλι μία μόνο παράμετρος, έστω p, όμως αυτή συμβολίζει την ποινή ανά μονάδα μήκους του κενού. Άρα για κενό μήκους m η συνολική ποινή του κενού που θα προστεθεί στη βαθμολογία της στοίχισης είναι pm. Έτσι, η συνολική ποινή ενός μεγάλου κενού είναι η ίδια με την ποινή πολλών μικρών κενών (που το άθροισμα των μηκών τους ισούται με το μήκος του μεγάλου κενού).
- *Συσχετισμένη ποινή κενών* (affine gap penalty). Είναι η περισσότερο χρησιμοποιούμενη μέθοδος βαθμολόγησης κενών, καθώς ανταποκρίνεται στις βιολογικές ιδιότητες των ακολουθιών. Χρησιμοποιούνται δύο παράμετροι, μία για την ποινή όταν αρχίζει ένα κενό, έστω o και μία για την ποινή όταν επεκτείνεται ένα κενό, έστω e. Έτσι η συνολική ποινή ενός κενού μήκους m ισούται με o+(m-1)e. Οι βιολογικές ακολουθίες είναι πιο πιθανό να έχουν ένα μεγάλο κενό παρά πολλά μικρά. Αυτό συμβαίνει γιατί για παράδειγμα ένα κενό μήκους 10 είναι πιθανώς να έχει προέλθει από μία εισαγωγή ή μία διαγραφή μέρους της ακολουθίας. Έτσι οι τιμές των παραμέτρων επιλέγονται ως εξής: το o (gap opening) πρέπει να είναι μια αρνητική τιμή, έτσι ώστε να αποθαρρύνεται η δημιουργία κενών. Η παράμετρος e (gap extention) θα πρέπει να έχει επίσης αρνητική τιμή, μικρότερη όμως κατ' απόλυτη τιμή, έτσι ώστε να ενθαρρύνεται η επέκταση και όχι η δημιουργία πολλών μικρών κενών.

2.3.4 Περιγραφή του αλγορίθμου Smith - Waterman

Ο αλγόριθμος Smith-Waterman χρησιμοποιείται για να αναγνωρίσει τις ομοιότητες μεταξύ δύο ακολουθιών. Για να το επιτύχει αυτό δρα σε 2 στάδια. Αρχικά υπολογίζει έναν πίνακα βαθμολογίας, που περιέχει όλες τις πιθανές συγκρίσεις. Έπειτα, ξεκινώντας από το στοιχείο του πίνακα με την υψηλότερη βαθμολογία κάνει οπισθοδρόμηση για να βρει την ακριβής

στοίχιση των δύο ακολουθιών που συγκρίνει. Στη συνέχεια θα περιγραφεί το κάθε στάδιο του αλγορίθμου όσο γίνεται εκτενέστερα με σκοπό την πλήρη κατανόησή του.

Έστω ότι έχουμε δύο ακολουθίες προς σύγκριση, την $A = a_1\ a_2\ a_3\ ...\ a_m$ και την $B = b_1\ b_2\ b_3\ ...\ b_n$ μήκους m και n αντίστοιχα. Ο πίνακας βαθμολογίας H που θα προκύψει, έχει διαστάσεις (m+1)(n+1). Η ποινή ανοίγματος κενού είναι G_s, η ποινή επέκτασης κενού είναι G_e και ο πίνακας αντικατάστασης είναι S. Η πρώτη στήλη και η πρώτη γραμμή του πίνακα H αρχικοποιούνται με μηδέν, δηλαδή ισχύει ότι $H(i, 0) = 0$ για $0 \le i \le m$ και $H(0, j) = 0$ για $0 \le j \le n$. Ο λόγος που γίνεται αυτό είναι για να έχουν αρχικές τιμές σύγκρισης τα οριακά στοιχεία του πίνακα. Τα υπόλοιπα στοιχεία του πίνακα, δηλαδή για $1 \le i \le m$ και $1 \le j \le n$ ισχύει:

$$H_{i,j} = Max \begin{Bmatrix} Max(H_{i-1,j-1} + S_{i,j}, 0) \\ Max_{0<k<j}(H_{i,j-k} - (G_s + kG_e)) \\ Max_{0<k<i}(H_{i-k,j} - (G_s + kG_e)) \end{Bmatrix}$$

Αυτό σημαίνει ότι το κάθε κελί του πίνακα είναι το μέγιστο από: α. το διαγώνια πάνω και αριστερά κελί αυξημένο κατά το σκορ που προκύπτει από τον πίνακα αντικατάστασης για τα στοιχεία της γραμμής και της στήλης, β. το μέγιστο όλων τον κελιών που βρίσκονται αριστερά από το εξεταζόμενο κελί στην ίδια γραμμή, μειωμένα κατά την ποινή κενού, γ. το μέγιστο όλων τον κελιών που βρίσκονται πάνω από το εξεταζόμενο κελί στην ίδια στήλη, μειωμένα κατά την ποινή κενού και δ. το μηδέν. Για παράδειγμα στο παρακάτω παράδειγμα χρησιμοποιούμε σκορ ομοιότητας 5, σκορ διαφοράς -3, ποινή ανοίγματος κενού 8 και ποινή επέκτασης κενού 1.

	0	**C**	**A**	**G**	**C**	**C**	**U**	**C**	**G**	**C**	**U**	**U**	**A**	**G**
0	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*
A	*0*	0	5	0	0	0	0	0	0					
A	*0*	0	5	2	0	0	0	0	0					
U	*0*	0	0	2	0	0	5	0	0					
G	*0*	0	0	5	0	0	0	2	5					
C	*0*	5	0	0	10	5	0	5	0					
C	*0*	5	2	0	5	15	6	5	**4**					
A	*0*													
U	*0*													
U	*0*													
G	*0*													
C	*0*													
C	*0*													
G	*0*													
G	*0*													

Υπολογισμός του πίνακα βαθμολογίας και οι εξαρτήσεις δεδομένων

Το εξεταζόμενο κελί παίρνει την τιμή 4 γιατί είναι το μέγιστο από την πάνω αριστερά τιμή 5 – 3 = 2, την πάνω τιμή 5 – 8 – 2 = -5 (που είναι το μέγιστο των 0 – 8 – 1 = -9, 5 – 8 – 2 = -5, 0 – 8 – 3 = -11 κ.ο.κ), την αριστερή τιμή 15 – 8 – 3 = 4 (που είναι το μέγιστο των 5 – 8 – 1 = -4, 6 – 8 – 2 = - 4, 15 – 8 – 3 = 4, 5 – 8 – 4 = -7 κ.ο.κ.) και του μηδενός. Η γραμμοσκιασμένη περιοχή του πίνακα δείχνει τις εξαρτήσεις που υπάρχουν.

Αν ο αλγόριθμος Smith-Waterman υλοποιηθεί με αυτόν τον τρόπο, τότε το μέγιστο της κάθε γραμμής και της κάθε στήλης (ο δεύτερος και ο τρίτος όρος της προηγούμενης εξίσωσης) υπολογίζονται ξανά και ξανά αυξάνοντας το υπολογιστική δουλειά του αλγορίθμου κατά O(nm(n+m)). Στο πρόβλημα αυτό δίνει την λύση ο δυναμικός προγραμματισμός. Το μέγιστο της στήλης που υπολογίζεται στο παραπάνω παράδειγμα, μπορεί να χρησιμοποιηθούν και στο κάτω κελί, ενώ το μέγιστο της γραμμής μπορεί να χρησιμοποιηθεί και στο δεξιά κελί. Έτσι αν αποθηκεύονται τα μέχρι εκείνη την στιγμή μέγιστα, τα επόμενα κελιά δεν θα χρειαζόταν να κάνουν συγκρίσεις για όλα τα πάνω και αριστερά κελιά, παρά μόνο μία σύγκριση για πάνω και μία για αριστερά. Για να επιτευχθεί αυτό, χρησιμοποιούνται άλλοι δύο πίνακες, οι E και F που αποθηκεύουν τα μέγιστα των γραμμών και στηλών αντίστοιχα. Εϊναι εμφανές ότι για τους πίνακες E και F ισχύει η αρχή της βελτιστότητας, αφού το κάθε κελί τους περιέχει την μέγιστη τιμή (άρα και βέλτιστη λύση) του αντίστοιχου υποπροβλήματος, δηλαδή κάθε κελί του πίνακα E περιέχει την μέγιστη τιμή των αριστερά κελιών της ίδιας γραμμής ενώ κάθε κελί του πίνακα F περιέχει την μέγιστη τιμή των πάνω κελιών της ίδιας στήλης.

Χρησιμοποιώντας την τεχνική του δυναμικού προγραμματισμού, για τον υπολογισμό του πίνακα H, χρησιμοποιούνται άλλοι δύο πίνακες, οι E και F διαστάσεων (m+1)(n+1) επίσης. Η πρώτη στήλη και η πρώτη γραμμή των πινάκων E, F και H ισούται με το μηδέν, δηλαδή έχουμε ότι E(i, 0) = F(i, 0) = H(i, 0) = 0 για $0 \le i \le m$ και E(0, j) = F(0, j) = H(0, j) = 0 για $0 \le j \le n$. Ο λόγος που γίνεται αυτό είναι για να έχουν αρχικές τιμές σύγκρισης τα οριακά στοιχεία των πινάκων. Για τα υπόλοιπα στοιχεία των πινάκων αυτών, δηλαδή για $1 \le i \le m$ και $1 \le j \le n$ ισχύουν τα εξής:

$$E_{i,j} = Max\left(E_{i,j-1} - G_e, H_{i,j-1} - G_s\right)$$
$$F_{i,j} = Max\left(F_{i-1,j} - G_e, H_{i-1,j} - G_s\right)$$
$$H_{i,j} = max\left(E_{i,j}, F_{i,j}, H_{i-1,j-1} + S_{A[i],B[j]}, 0\right)$$

Όπως είναι φανερό από τις παραπάνω εξισώσεις, ο Smith – Waterman είναι πολύ απαιτητικός αλγόριθμος. Για τον υπολογισμό του κάθε κελιού του πίνακα βαθμολογίας, απαιτούνται οι εξής ενέργειες :

- Ανάγνωση από 5 θέσεις μνήμης.
- Εγγραφή σε 3 θέσεις μνήμης
- 5 προσθέσεις – αφαιρέσεις
- 5 conditional branches για να υπολογιστούν τα 3 μέγιστα

Επίσης, ο υπολογισμός του κάθε κελιού του πίνακα εξαρτάται από το αριστερό, το πάνω και το πάνω-αριστερά κελί. Αυτό πρακτικά σημαίνει ότι για να υπολογιστούν τα στοιχεία μιας αντιδιαγωνίου, απαιτούνται οι 2 προηγούμενες αντιδιαγώνιοι, άρα υπάρχει εξάρτηση της αντιδιαγωνίου i από τις αντιδιαγώνιες i-1 και i-2, κάτι που επηρεάζει σημαντικά τον τρόπο με τον οποίο υπολογίζονται τα κελιά του πίνακα.

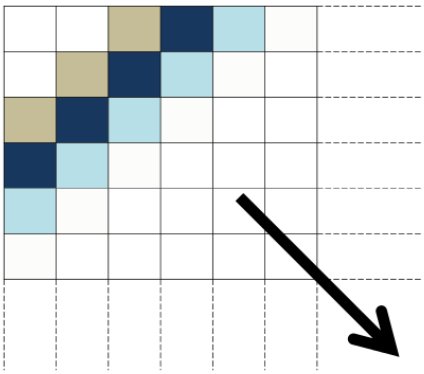
	0	C	A	G	C	C	U	C	G	C	U	U	A	G
0	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*
A	*0*	0	5	0	0	0	0	0	0	0	?			
A	*0*	0	5	2	0	0	0	0	0	?				
U	*0*	0	0	2	0	0	5	0	?					
G	*0*	0	0	5	0	0	0	?						
C	*0*	5	0	0	10	5	?							
C	*0*	5	2	0	5									
A	*0*	0	10	1	?									
U	*0*	0	1	?										
U	*0*	0	?											
G	*0*	?												
C	*0*													
C	*0*													
G	*0*													
G	*0*													

Υπολογισμός του πίνακα βαθμολογίας κατά αντιδιαγωνίους

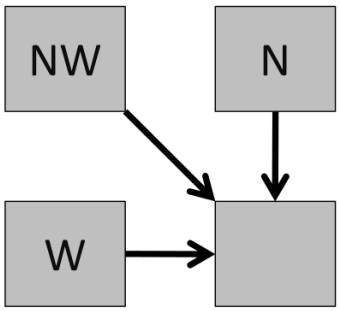
Εφαρμόζοντας δυναμικό προγραμματισμό στον αλγόριθμο Smith-Waterman, τριπλασιάζουμε τις απαιτήσεις σε μνήμη, όμως μειώνουμε πολύ την υπολογιστική δουλειά που πρέπει να γίνει, κερδίζοντας σε ταχύτητα. Η χρονική πολυπλοκότητα του αλγορίθμου για την στοίχιση δύο ακολουθιών είναι O(nm), όπου n και m τα μήκη των ακολουθιών. Στην πράξη όμως, ο αλγόριθμος δεν χρησιμοποιείται για να στοιχίσει μόνο δύο ακολουθίες, αλλά για q ακολουθίες ενός query πάνω σε μία database που περιέχει d ακολουθίες. Άρα συνολικά γίνονται qd στοιχίσεις κόστους O(nm).

Οι εξαρτήσεις που έχουν τα στοιχεία της κάθε αντιδιαγωνίου, δηλαδή οι 2 προηγούμενες αντιδιαγώνιοι υποδεικνύει την βέλτιστη σειρά υπολογισμού των κελιών του πίνακα βαθμολογίας. Ο πίνακας αυτός υπολογίζεται κατά αντιδιαγωνίους, κάτι που χαρακτηρίζει τον

αλγόριθμο Smith-Waterman ως ισοφασικό ή κυματικό αλγόριθμο (wavefront algorithm). Ο χαρακτηρισμός αυτός έγκειται στο γεγονός ότι τα υπολογιζόμενα στοιχεία της κάθε αντιδιαγωνίου μαζί με τις εξαρτήσεις τους μοιάζουν με ένα κύμα που διατρέχει τον πίνακα από πάνω-αριστερά έως κάτω-δεξιά.



Το "κύμα" που διατρέχει τον πίνακα



Οι εξαρτήσεις του κάθε κελιού

Ο υπολογισμός του πίνακα βαθμολογίας είναι το πιο επεξεργαστικά απαιτητικό κομμάτι του αλγορίθμου. Όταν αυτό έχει ολοκληρωθεί, η βέλτιστη στοίχιση των δύο ακολουθιών υπολογίζεται κάνοντας μία οπισθοδρόμιση στον πίνακα βαθμολογίας. Η οπισθοδρόμηση ξεκινάει από το στοιχείο του πίνακα με την υψηλότερη βαθμολογία το οποίο δείχνει το τελευταίο στοιχείο της βέλτιστης στοίχισης. Από το κελί αυτό ο αλγόριθμος κινείται προς τα πάνω και προς τα αριστερά, ψάχνοντας από ποιο κελί προήλθε αυτή η τιμή. Όταν η κίνηση είναι προς τα πάνω, προστίθεται ένα κενό στην πρώτη ακολουθία, ενώ όταν η κίνηση είναι προς τα αριστερα, προστίθεται ένα κενό στη δευτερη ακολουθία. Όταν η κίνηση είναι διαγώνια πάνω-αριστερά, δεν εισάγεται κανένα κενό. Η οπισθοδρόμηση σταματάει όταν το επόμενο κελί έχει βαθμολογία μηδέν. Όταν κάποιες από τις 3 επιλογές έχουν την ίδια βαθμολογία, προτιμάται η διαγώνια κίνηση, δηλαδή η κίνηση που δεν εισάγει κενό. Αν αυτό δεν είναι εφικτό (τα πάνω και αριστερά έχουν υψηλότερη βαθμολογία) τότε επιλέγεται η κίνηση που εισάγει κενό στην πρώτη ακολουθία. Η επιλογή αυτή έχει βιολογικούς λόγους, καθώς στην θέση της πρώτης ακολουθίας συνήθως τοποθετείται η βάση δεδομένων των διαθέσιμων ακολουθιών και οι βιολόγοι βγάζουν σωστότερα συμπεράσματα όταν τα κενά βρίσκονται σε αυτές τις ακολουθίες.

	0	C	A	G	C	C	U	C	G	C	U	U	A	G
0	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*
A	*0*	0	5	0	0	0	0	0	0	0	0	0	5	0
A	*0*	0	5	2	0	0	0	0	0	0	0	0	5	2
U	*0*	0	0	2	0	0	5	0	0	0	5	5	0	2
G	*0*	0	0	***5***	0	0	0	2	5	0	0	2	2	5
C	*0*	5	0	0	***10***	5	0	5	0	10	1	0	0	0
C	*0*	5	2	0	5	***15***	6	5	4	5	7	1	0	0
A	*0*	0	10	1	0	***6***	12	3	2	1	2	4	6	0
U	*0*	0	1	7	0	5	***11***	9	1	0	6	7	1	3
U	*0*	0	0	0	4	4	10	***8***	6	0	5	11	4	1
G	*0*	0	0	5	0	3	1	7	***13***	4	3	2	8	9
C	*0*	5	0	0	10	5	0	6	4	***18***	9	8	7	6
C	*0*	5	2	0	5	15	6	5	4	9	15	6	5	4
G	*0*	0	2	7	0	6	12	3	10	8	6	12	3	10
G	*0*	0	0	7	4	5	3	9	8	7	5	3	9	8

Η διαδικασία της οπισθοδρόμησης στον συμπληρωμένο πίνακα βαθμολογίας

ΚΕΦΑΛΑΙΟ 3

ΥΠΟΛΟΓΙΣΜΟΙ ΓΕΝΙΚΟΥ ΣΚΟΠΟΥ ΣΕ ΕΠΕΞΕΡΓΑΣΤΕΣ ΓΡΑΦΙΚΩΝ

3.1 Οι επιστημονικοί αλγόριθμοι και οι επεξεργαστές γραφικών

Οι επιστημονικοί αλγόριθμοι συνήθως έχουν να επεξεργαστούν μεγάλο όγκο συνεχόμενων δεδομένων (streaming data). Το γεγονός αυτό κάνει τους κλασσικούς επεξεργαστές όχι και τόσο κατάλληλους για να τέτοιου είδους υπολογισμούς, αφού βασίζονται σε ιεραρχίες μνήμης που δεν είναι ικανές να χειριστούν αποδοτικά τόσο μεγάλο όγκο δεδομένων. Έτσι, η απόδοση σχεδόν όλων των επιστημονικών αλγορίθμων πλέον περιορίζεται από την ταχύτητα της μνήμης (είναι memory bound) και οι επεξεργαστές δεν μπορούν να εκμεταλλευτούν μεγάλος μέρος της επεξεργαστικής ισχύς τους.

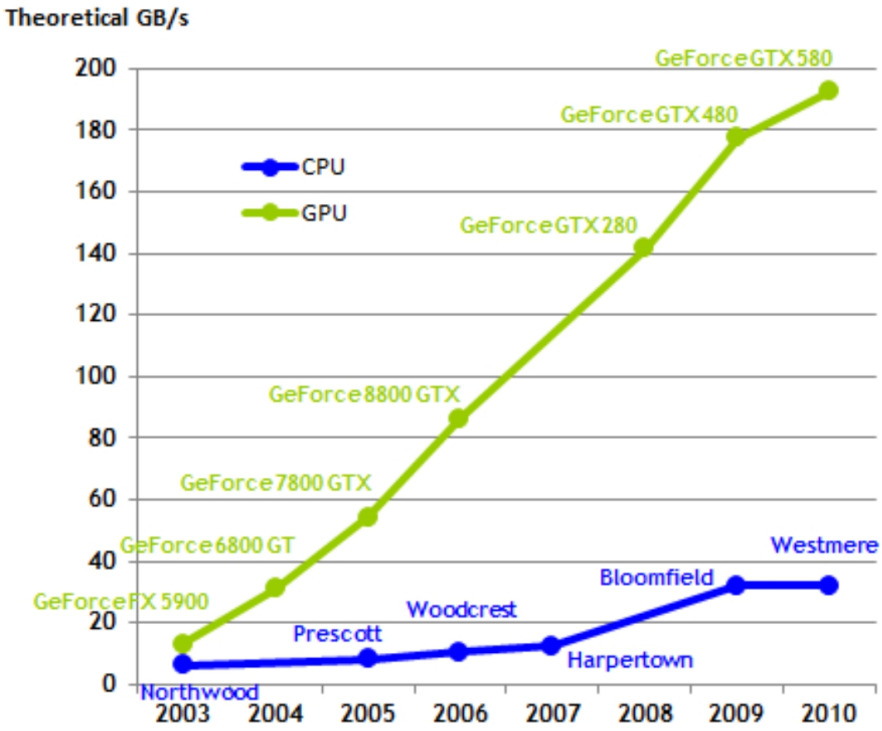
Η λύση σε αυτό το πρόβλημα δεν ήρθε από τους κατασκευαστές επεξεργαστών, αλλά από έναν διαφορετικό χώρο που αρκετοί δυσπιστούσαν ότι μπορεί να βοηθήσει, τον χώρο των επεξεργαστών γραφικών. Η αλήθεια είναι ότι οι επεξεργαστές γραφικών δεν είναι πανάκεια, καθώς σε αντίθεση με τους κλασσικούς επεξεργαστές, είναι σχεδιασμένοι για υπολογισμούς ειδικού σκοπού, και όχι γενικού. Παρόλα αυτά, για συγκεκριμένες εφαρμογές οι

επεξεργαστές γραφικών δείχνουν την δύναμη τους, προσφέροντας επιδόσεις πολλαπλάσιας τάξης μεγέθους. Για να γίνει κάτι τέτοιο, πρέπει ο προγραμματιστής να έχει βαθιά γνώση της αρχιτεκτονικής των επεξεργαστών γραφικών, έτσι ώστε να μπορέσει να εκμεταλλευτεί την ιδιαίτερη αρχιτεκτονική τους και να "ξεκλειδώσει" τις επιδόσεις.

Μέχρι πρόσφατα αρκετοί διαφωνούσαν με την άποψη ότι οι μελλοντικοί υπολογιστές θα ενσωματώνουν χιλιάδες επεξεργαστικούς πυρήνες. Πλέον το όραμα αυτό έχει ήδη εκπληρωθεί, αφού η ρυθμός αύξησης επεξεργαστικής ισχύος των επεξεργαστών γραφικών ξεπέρασε κατά πολύ τους παραδοσιακούς επεξεργαστές, με αποτέλεσμα οι σημερινές κάρτες γραφικών να έχουν χιλιάδες πυρήνες διαθέσιμους για επεξεργασία δεδομένων. Η εικόνα αυτή έχει γίνει εμφανής και στον τομέα των υπερυπολογιστών (supercomputers), αφού οι επεξεργαστές γραφικών έχουν εξαπλωθεί ευρέως και οι υπερυπολογιστές που τους ενσωματώνουν ανεβαίνουν ολοένα και πιο ψηλά στην γνωστή λίστα των Top500 supercomputers.

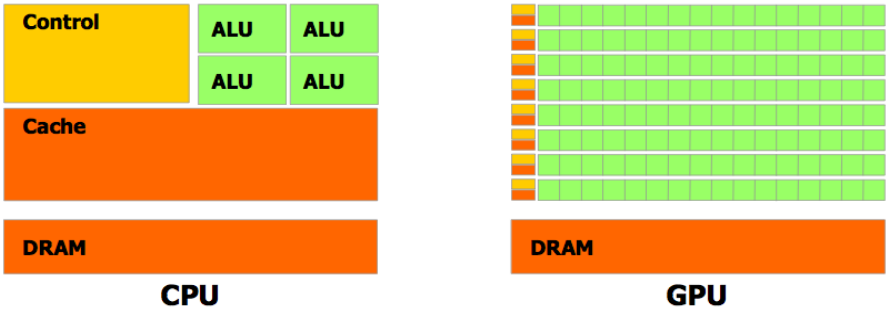
3.2 Διαφορές αρχιτεκτονικής CPU - GPU

Στην σύγχρονη αρχιτεκτονική υπολογιστών, η τάση για την αύξηση της συχνότητας λειτουργίας των επεξεργαστικών μονάδων έχει πλέον εγκαταλειφθεί. Την θέση της, πήρε η προσπάθεια αύξησης του αριθμού των επεξεργαστικών μονάδων. Όλοι αυτοί οι πυρήνες για να πάρουν δεδομένα προς επεξεργασία, πρέπει να "μιλήσουν" με την κύρια μνήμη του συστήματος, η ταχύτητα της οποίας όπως είναι γνωστό εδώ και χρόνια, αυξάνεται με πολύ μικρότερο ρυθμό απ' ότι η ταχύτητα και το πλήθος των πυρήνων. Έτσι έχουμε σαν αποτέλεσμα, το εύρος της μνήμης να αποτελεί περιορισμό (bottleneck) για την επίδοση, δηλαδή η διαθέσιμη επεξεργαστικής ισχύς μένει ανεκμετάλλευτη, κάτι το οποίο οι σύγχρονοι αρχιτέκτονες υπολογιστών καλούνται να λύσουν, εξισορροπώντας τις δυνατότητες των cores και της μνήμης.



Μέγιστο θεωρητικό έυρος μνήμης σε CPUs και GPUs

Ο τρόπος που χρησιμοποιεί η κάθε αρχιτεκτονική για να αντιμετωπίσει το πρόβλημα που ακούει στο όνομα memory bandwidth bottleneck, φαίνεται ακόμα και σε μια λιθογραφική φωτογραφία του κάθε πυρήνα. Έτσι, σε έναν επεξεργαστή γενικού σκοπού (CPU), το μεγαλύτερο μέρος του die καλύπτεται από κρυφές μνήμες (caches). Πράγματι, όλες οι σημερινοί CPU μειώνουν το πρόβλημα του εύρους μνήμης με την χρήση ιεραρχίας πολλών επιπέδων κρυφών μνημών (multi-level cache hierarchy). Αντίθετα, το μεγαλύτερο μέρος του die ενός επεξεργαστή γραφικών (GPU) καλύπτεται από αριθμητικές μονάδες (ALUs). Και όσο και αν δεν ακούγεται λογικό, οι GPUs αντιμετωπίζουν το θέμα της μνήμης με τις τραγικά περισσότερες ALUs. Αυτό γίνεται μέσω του τεράστιου αριθμού των νημάτων (threads) που μπορούν να διαχειριστούν, πίσω από τα οποία μπορούν να "κρυφτούν" οι προσβάσεις μνήμης (memory accesses), εξαλείφοντας τη μεγάλη καθυστέρηση (latency) που υπάρχει.



Σύγκριση της χρήσης του die μεταξύ CPU και GPU

Οι σχεδιαστικές αυτές διαφορές μεταξύ CPU και GPU, διαφοροποιούν και τον τρόπο επεξεργασίας των δεδομένων. Έτσι, οι CPU επιτυγχάνουν υψηλή επίδοση σε κάθε διαφορετικό thread εκτέλεσης και εκτελούν παράλληλα διαφορετικά threads / tasks, υλοποιώντας task parallelism επεξεργασία. Από την άλλη, οι GPU πετυχαίνουν υψηλή επίδοση όταν έχουμε μεγάλο σύνολο δεδομένων στα οποία πρέπει να εκτελεστεί η ίδια λειτουργία υλοποιώντας data parallelism επεξεργασία. Επειδή λοιπόν εκτελείται η ίδια λειτουργία σε όλα τα δεδομένα, υπάρχουν πολύ μικρότερες απαιτήσεις για υλικό που ελέγχει την ροή του προγράμματος (π.χ. branch control), άρα ακόμα μεγαλύτερος χώρος στο die για επεξεργαστικές μονάδες. Με λίγα λόγια, οι GPUs αποτελούν SIMD (Single Instruction Multiple Data) ή καλύτερα SIMT (Single Instruction Multiple Thread) επεξεργαστικές μονάδες, εξειδικευμένες για υπολογισμούς υψηλού παραλληλισμού και πολλών threads.

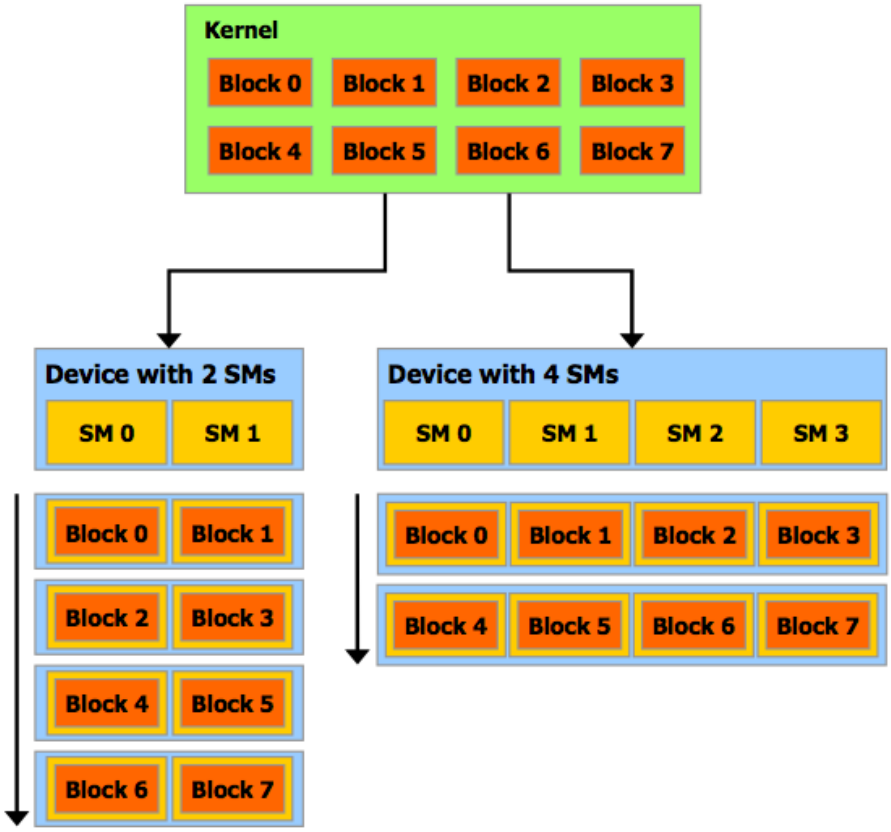
3.3 Η αρχιτεκτονική CUDA της Nvidia

3.3.1 Οι στόχοι

Τον Νοέμβρη του 2006, η Nvidia παρουσίασε την Compute Unified Device Architecture, μία γενικού σκοπού παράλληλη υπολογιστική αρχιτεκτονική, που εκμεταλλεύεται τις GPUs της εταιρίας για να αντιμετωπιστούν πολλά πολύπλοκα προβλήματα με πιο αποδοτικό τρόπο απ' ότι σε ένα CPU. Οι γλώσσες τις οποίες υποστηρίζει η αρχιτεκτονική CUDA, είναι η CUDA C, η OpenCL, η DirectCompute και η CUDA Fortran.

Ο νόμος του Moore συνεχίζει να ισχύει και στις μέρες μας, και πλέον η αύξηση του αριθμού των transistors μεταφράζεται σε αύξηση του αριθμού των επεξεργαστικών πυρήνων. Έτσι, οι επεξεργαστές παρέχουν όλο και περισσότερο την δυνατότητα παράλληλης εκτέλεσης κώδικα. Σκοπός της αρχιτεκτονικής CUDA είναι να παρέχει ένα επεκτάσιμο προγραμματιστικό μοντέλο, με την έννοια ότι ο κώδικας που γράφεται θα συνεχίσει να τρέχει αποδοτικά και στο

μέλλον, καθώς οι διαθέσιμοι πυρήνες αυξάνονται. Η επεκτασιμότητα αυτή επιτυγχάνεται μέσω τριών βασικών χαρακτηριστικών: α. ιεραρχία ομάδων των threads, β. ιεραρχία διαμοιραζόμενης μνήμης και γ. συγχρονισμός.

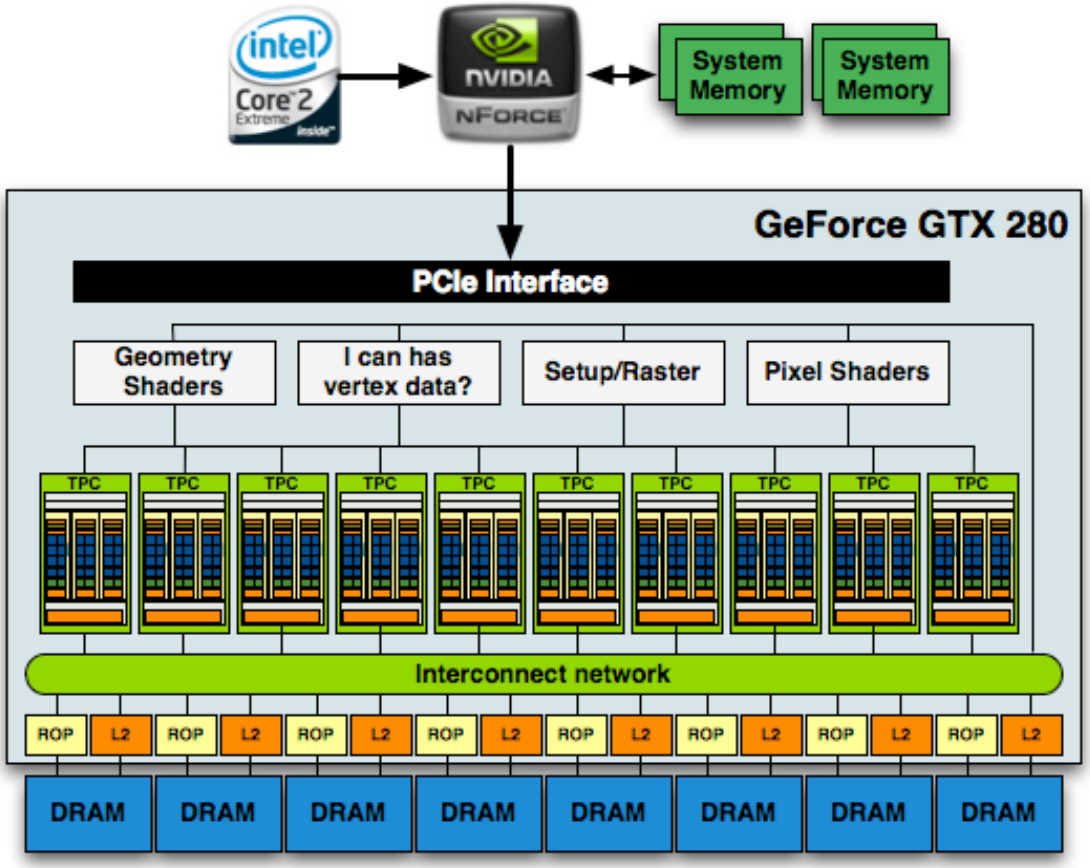


Ο διαμοιρασμός των thread blocks ανάλογα με τα τους διαθέσιμους πολυεπεξεργαστές

Ο προγραμματιστής πρέπει να χωρίσει το προς επίλυση πρόβλημα σε υποπροβλήματα τα οποία δεν εμφανίζουν αλληλοεξαρτήσεις και μπορούν να λυθούν ανεξάρτητα σε παράλληλα thread blocks, και το κάθε υποπρόβλημα σε επιμέρους τμήματα που μπορούν να λυθούν συνεργατικά από τα threads του block. Έτσι τα thread blocks μπορούν να δρομολογηθούν σε οποιοδήποτε από τους διαθέσιμους πολυεπεξεργαστές (SM), με οποιαδήποτε σειρά, ταυτόχρονα ή σειριακά, γεγονός που επιτρέπει στο πρόγραμμα να εκτελεστεί σε οποιοδήποτε αριθμό πυρήνων υπάρχουν διαθέσιμοι κατά το runtime. Για παράδειγμα, όπως φαίνεται και στο προηγούμενο σχήμα, αν υπάρχουν 2 διαθέσιμοι πολυεπεξεργαστές, τα thread blocks εκτελούνται ανά 2 παράλληλα, ενώ αν υπάρχουν 4 εκτελούνται ανά 4 παράλληλα. Έτσι, το επεκτάσιμο προγραμματιστικό μοντέλο της CUDA αρχιτεκτονικής, επιτρέπει στον παράλληλο κώδικα να τρέχει είτε σε τελευταίας γενίας, υψηλής επίδοσης επεξεργαστές γραφικών με πολλούς πολυεπεξεργαστές, είτε σε απλούς επεξεργαστές γραφικών που διαθέτουν μικρότερο αριθμό πολυεπεξεργαστών.

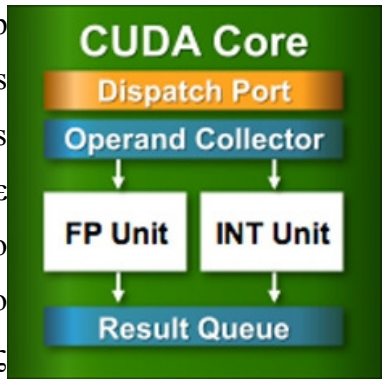
3.3.2 Περιγραφή της αρχιτεκτονικής CUDA

.



Διάρθρωση ενός επεξεργαστή γραφικών CUDA

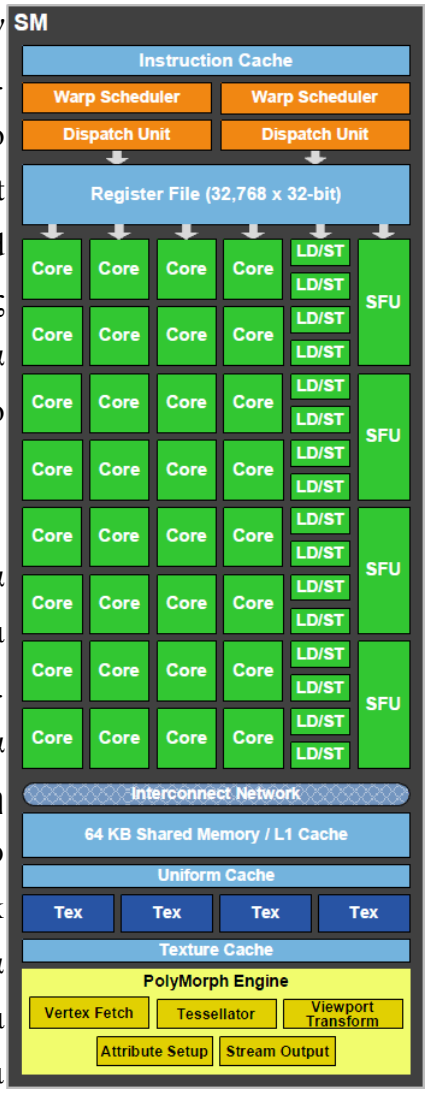
Το κύριο δομικό στοιχείο των συσκευών CUDA είναι τα multithreaded Streaming Multiprocessors (SMs) τα οποία εκτελούν τα thread blocks (work-group). Ο υπολογιστικός πυρήνας (kernel) εκτελείται μέσω ενός πλέγματος από thread blocks. Κάθε SM αποτελείται από 32 Scalar Processors (SPs), 4 ειδικές μονάδες για άρρητους - transcedentals, 2 μονάδες διαχείρισης των multithreaded εντολών και on-chip διαμοιραζόμενη μνήμη (shared memory). Επίσης, 4 SMs αποτελούν ένα Graphics Processing Cluster (GPC). Τα threads εκτελούνται στα SPs. Κάθε thread block αναγνωρίζεται με μοναδικό τρόπο από το work-group ID και κάθε thread από το global ID ή από τον συνδυασμό του work-group ID στο οποίο ανήκει και το local ID. Τα 3 αυτά IDs συνδέονται με την εξής σχέση:



Διάρθρωση ενός SP

*global_id = local_id + group_id * local_size*

Τα SMs δημιουργούν, διαχειρίζονται και εκτελούν ταυτόχρονα threads με μηδενικό overhead δρομολόγησης. Συγχρονίζουν τα work-groups με μία μόνο εντολή. Έτσι, ο ταχύτατος συγχρονισμός σε συνδυασμό με τη lightweight δημιουργία των threads και το μηδενικό overhead δρομολόγησης επιτρέπουν παραλληλισμό υψηλής απόδοσης καθώς και την διάσπαση των προβλημάτων σε πολύ μικρά μέρη, ακόμα και να αντιστοιχεί ένα thread σε κάθε στοιχείο δεδομένων (data element).



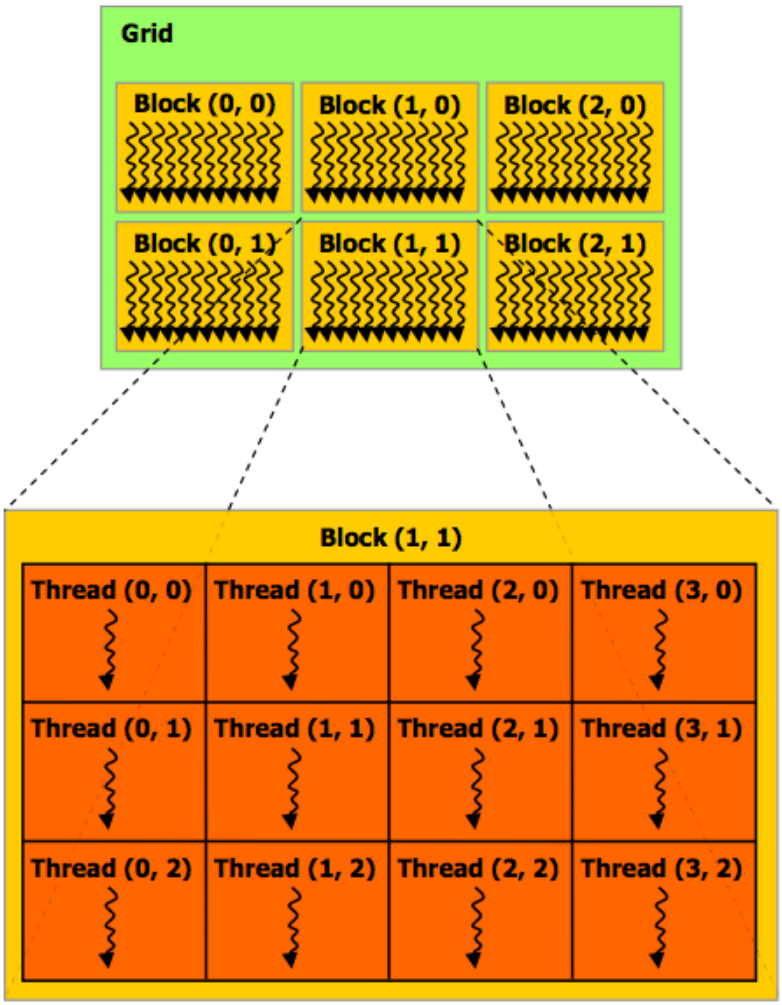
Διάρθρωση ενός SM

Όταν ο υπολογιστικός πυρήνας (kernel) ξεκινήσει να εκτελείται, τα work-groups απαριθμούνται και μοιράζονται σαν thread blocks στους διαθέσιμους multiprocessors (SMs). Τα thread blocks εκτελούνται ανεξάρτητα. Μπορούν να εκτελεστούν είτε παράλληλα είτε το ένα μετά το άλλο. Αυτή η ανεξαρτησία είναι αυτή που επιτρέπει το scalability που αναφέραμε πιο πάνω. Τα threads του κάθε thread block εκτελούνται ταυτόχρονα στα SPs του SM. Όταν όλα τα threads ενός thread block ολοκληρωθούν, ολοκληρώνεται και το thread block και νέα blocks ξεκινούν να εκτελούνται στα ελεύθερα (idle) SMs.

Τα SMs δημιουργούν, διαχειρίζονται και εκτελούν παράλληλα εκαντοντάδες threads. Τα threads αυτά χωρίζονται σε ομάδες των 32, που ονομάζονται warps. Τα ανεξάρτητα threads που αποτελούν ένα warp ξεκινούν μαζί έχοντας την ίδια programm address, αλλά το καθένα έχει τον δικό του address counter και register state, και έτσι μπορεί να εκτελεστεί ανεξάρτητα. Ο όρος warp προέρχεται από το weaving την πρώτη τεχνολογία παράλληλων threads.

Όταν στέλνονται σε ένα SM ένα ή περισσότερα thread blocks για εκτέλεση, το SM τα χωρίζει σε warps τα οποία χρονοδρομολογούνται προς εκτέλεση από έναν warp scheduler. Ο τρόπος με τον οποίο τα thread blocks χωρίζονται σε warps είναι πάντα ο ίδιος: Κάθε warp περιλαμβάνει threads συνεχόμενου και αυξανόμενου thread ID, και το πρώτο warp περιλαμβάνει το thread 0. Το warp εκτελεί μία εντολή κάθε φορά, έτσι ώστε όταν και τα 32 threads του warp έχουν ίδιο execution path, παρατηρείται μέγιστη απόδοση. Αν τα threads

ενός warp αποκλίνουν, έχοντας διαφορετικό execution path λόγω ενός data-dependent conditional branch, το warp εκτελεί σειριακά το κάθε path, απενεργοποιόντας τα threads που δεν συμβαδίζουν με το path αυτό, και όταν όλα τα paths ολοκληρωθούν, τότε τα threads συγκλίνουν και πάλι σε κοινό execution path. Τα warps των οποίων τα threads ακολουθούν διαφορετική ροή ονομάζονται divergent warps. Στην πραγματικότητα, σε αντίθεση με τους κλασσικούς επεξεργαστές, στις GPUs δεν υπάρχει πρόβλεψη διακλάδωσης (branch prediction) και υποθετική εκτέλεση (speculative execution). Από την άλλη, διαφορετικά warps εκτελούνται ανεξάρτητα, ασχέτως αν έχουν κοινό ή διαφορετικό execution path.



Ο διαμοιρασμός των threads se thread blocks

3.3.3 Το προγραμματιστικό μοντέλο SIMT

Το προγραμματιστικό μοντέλο SIMT (Single Instrustion Multiple Thread) έχει παραπλήσια χαρακτηριστικά με την αρχιτεκτονική SIMD (Single Instruction Multiple Data). Η κύρια διαφορά είναι ότι η SIMD παραχωρεί την διαχείριση του SIMD width στο software, ενώ στο SIMT οι εντολές καθορίζουν την εκτέλεση και τις διακλαδώσεις (branching). Έτσι στο SIMT, οι προγραμματιστές γράφουν thread-level parallel κώδικα για τα ανεξάρτητα threads και data-parallel κώδικα για τα threads με ίδιο execution path. Για λόγους ορθότητας, οι προγραμματιστές μπορούν να μην λάβουν υπ' όψην την SIMT συμπεριφορά. Σε αυτή την περίπτωση όμως, δεν θα υπάρχει κάποια ιδιαίτερα υψηλή απόδοση. Αυτό είναι ανάλογο με την εκμετάλλευση των cache lines σε έναν κλασσικό κώδικα γραμμένο για CPU. Ο προγραμματιστής μπορεί να αγνοήσει την ύπαρξή τους, αλλά μόνο χρησιμοποιώντας σωστά τα cache lines μπορεί να επιτύχει υψηλή απόδοση. Στην πραγματικότητα, ο κώδικας θα πρέπει σπάνια να αποκλίνει από το warp.

Το "πλαίσιο" εκτέλεσης – execution context (programm counters, registers κλπ) κάθε warp συντηρείται on-chip, καθ' όλη την διάρκεια εκτέλεσης του. Έτσι, η εναλλαγή μεταξύ των contexts (context switching) έχει μηδενικό κόστος και κατά το issuing των εντολών, ο warp scheduler επιλέγει ένα warp που έχει όλα τα threads του έτοιμα προς εκτέλεση (άρα όλα τα source data έτοιμα) και κάνει issue επόμενη εντολή των threads αυτών. Τα έτοιμα προς εκτέλεση warps ονομάζονται ενεργά – active warps.

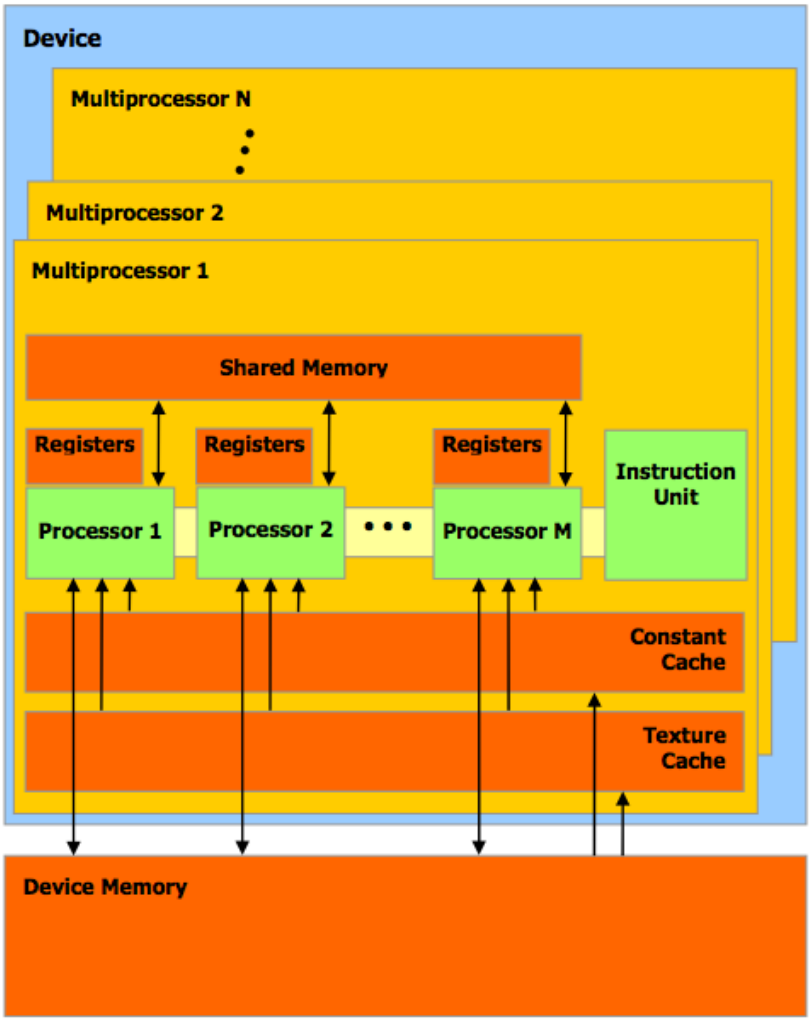
Κάθε SM έχει μια ομάδα από 32-bit registers και μία παράλληλη data cache - shared memory που μοιράζεται σε όλα τα νήματα ενός block. Ο αριθμός των blocks και των warps που μπορούν να συνυπάρχουν στο κάθε SM, εξαρτάται από το πλήθος των registers και το μέγεθος της shared memory, που διαθέτουν τα SMs. Αν δεν υπάρχουν αρκετοί registers ή shared memory διαθέσιμα έτσι ώστε το κάθε SM να μπορεί να επεξεργαστεί τουλάχιστον ένα thread block, ο kernel δεν θα μπορεί να ξεκινήσει. Τα μεγέθη αυτά καθώς και άλλα ειδικά τεχνικά χαρακτηριστικά διαφοροποιούν τις διάφορες γενίες των GPUs και περιγράφονται από την υπολογιστική ικανότητα (compute capability) του κάθε CUDA device. Το compute capability ορίζεται από ένα κύριο αριθμό έκδοσης και ένα αριθμό υπο-έκδοσης στην μορφή <major-number>.<minor-number>. CUDA devices με ίδιο κύριο αριθμό έκδοσης έχουν την ίδια αρχιτεκτονική πυρήνα και ο αριθμός υπο-έκδοσης αναφέρεται μόνο σε μικρές βελτιώσεις στην αρχιτεκτονική πυρήνα, συμπεριλαμβάνοντας πιθανώς κάποιες νέες δυνατότητες.

3.3.4 Το μοντέλο μνήμης

Αναπόσπαστο μέρος της πολύπλοκης αρχιτεκτονικής των επεξεργαστών γραφικών, είναι και η διάρθρωση της μνήμης. Υπάρχουν διάφορα είδη μνήμης, διαφορετικού μεγέθους και τύπου, που όπως είναι λογικό προσφέρουν διαφορετική απόδοση. Λόγω των ιδιαίτερων αυτών χαρακτηριστικών, η κάθε μνήμη είναι κατάλληλη για διαφορετικά δεδομένα. Η σωστή χρήση των διαθέσιμων μνημών αποτελεί βασικό κλειδί για την επίτευξη υψηλής απόδοσης. Έτσι κρίνεται αναγκαία η παρουσίαση των διαθέσιμων ειδών μνήμης. Συγκεκριμένα υπάρχουν:

- 32k τοπικοί 32-bit καταχωρητές (registers).
- L1 κρυφή μνήμη (cache) μεγέθους 16KB ή 48KB (εξαρτάται από το μέγεθος της shared memory) με cache line 128 bytes για κάθε SM.
- L2 κρυφή μνήμη (cache) μεγέθους 768KB για κάθε GPC με cache line 128 bytes που μοιράζεται σε όλα τα SMs.
- Παράλληλη data cache (shared memory) μεγέθους 16KB ή 48KB που διαμοιράζεται σε όλα τα SPs ενός SM και έχει μηδενική καθυστέρηση. Όπως είναι φανερό, η L1 cache και η shared memory έχουν μεταβλητό μέγεθος. Συνολικά αυτές οι δύο caches έχουν μέγεθος 64KB που μπορεί να χωριστεί είτε σαν 16KB / 48KB είτε σαν 48KB / 16KB για την L1 και την shared memory αντίστοιχα. Η shared memory έχει 32 banks τα οποία είναι οργανωμένα έτσι ώστε διαδοχικά 32-bit words να αναθέτονται σε διαδοχικά banks. Το κάθε bank έχει εύρος 32 bits ανά 2 κύκλους ρολογιού. Η shared memory "αρχιτεκτονικά" είναι cache προσφέροντας πολύ υψηλή ταχύτητα, πρακτικά όμως είναι είναι μνήμη διαχειριζόμενη από το εκτελούμενο πρόγραμμα, μιας και οι εγγραφές και οι αναγνώσεις σε αυτήν δεν γίνονται από το hardware, αλλά ορίζονται από τον προγραμματιστή.
- Ενιαία read-only constant cache μεγέθους 64KB που διαμοιράζεται σε όλα τα SPs και χρησιμοποιείται για να επιταχύνει τις αναγνώσεις από την constant memory, η οποία βρίσκεται στην κύρια μνήμη.
- Ενιαία read-only texture cache μεγέθους 8KB που διαμοιράζεται σε όλα τα SPs και χρησιμοποιείται για να επιταχύνει τις αναγνώσεις από την texture memory, η οποία βρίσκεται στην κύρια μνήμη. Κάθε SM έχει πρόσβαση στην cache αυτή μέσω ενός texture unit που υλοποιεί διάφορα addressing modes και data filtering. Στην cache αυτή αποθηκεύονται image objects προς επεξεργασία.
- Global memory address space (κύρια μνήμη της κάρτας γραφικών) το οποίο είναι read-write. Κάποιο μέρος του global memory address space υλοποιεί την local

memory, η οποία είναι private σε κάθε thread και συνήθως χρησιμοποιείται από τον compiler για αυτόματες αναθέσεις μεταβλητών. Η global memory έχει 400-600 κύκλους καθυστέρηση και είναι προσβάσιμη από όλα τα SMs. Υπάρχει η δυνατότητα χρήσης ECC μνήμης και χρησιμοποιείται 64-bit διευθυνσιοδότηση έτσι ώστε να μην υπάρχει πρακτικός περιορισμός στο μέγεθος της διαθέσιμης μνήμης.



Το μοντέλο μνήμης

3.3.5 OpenCL

Η OpenCL – Open Computing Language είναι ένα framework για την δημιουργία προγραμμάτων που εκτελούνται σε ετερογενείς αρχιτεκτονικές. Εμπεριέχει μία γλώσσα βασισμένη στην C99, για την συγγραφή πυρήνων – kernels καθώς και APIs που χρησιμοποιούνται για να χειριστούν την εκάστοτε πλατφόρμα. Η OpenCL παρέχει δυνατότητα για parallel computing χρησιμοποιώντας task-based και data-based parallelism. Δίνει λοιπόν την δυνατότητα να χρησιμοποιηθεί μία GPU για μη γραφικούς υπολογισμούς επεκτείνοντας έτσι την ισχύ των GPU. Οι υπολογισμοί αυτοί ονομάζονται General-Purpose Computing on Graphics Processing Units. Το μεγαλύτερο ίσως πλεονέκτημα της OpenCL, είναι ότι μπορεί να χρησιμοποιηθεί τόσο σε κάρτες γραφικών της Nvidia όσο και της AMD (ATI).

Μία κύρια διαφορά της OpenCL σε σχέση με τα υπόλοιπα GPGPU frameworks είναι ότι το compilation των kernels γίνεται κατά το runtime. Έτσι δίνεται η δυνατότητα στους προγραμματιστές να αλλάζουν τον κώδικα των kernels ή ακόμα και να προσθέτουν νέους kernels προς εκτέλεση, χωρίς να χρειάζεται να κάνουν compile όλο τον κώδικά τους κάθε φορά. Είναι πάγια τακτική να χρησιμοποιούνται 2 αρχεία, ένα με κατάληξη .cpp που περιέχει το πρόγραμμα και ένα με κατάληξη .cl που περιέχει τον κώδικα των kernels, το οποίο καλείται από το πρόγραμμα και γίνεται compile κατά το runtime.

3.4 Τεχνικές επίτευξης υψηλής επίδοσης

Η ιδιαίτερη αρχιτεκτονική των επεξεργαστών γραφικών διαφοροποιεί αρκετά τον τρόπο προγραμματισμού τους. Έτσι, για να επιτευχθούν υψηλές επιδόσεις ο προγραμματιστής θα πρέπει να έχει βαθιά γνώση της αρχιτεκτονικής για να μπορεί να βελτιστοποιήσει τον κώδικα πάνω σε αυτή. Σίγουρος τρόπος δεν υπάρχει καθώς το ποια στρατηγική θα επιφέρει το μεγαλύτερο κέρδος σε επίδοση εξαρτάται από το είδος της εφαρμογής. Ωστόσο οι τεχνικές που παρουσιάζονται παρακάτω συνήθως δίνουν την επιθυμητή απόδοση.

3.4.1 Αύξηση της χρησιμοποίησης του επεξεργαστή γραφικών

Όπως είναι λογικό, βασικός ρόλο στην επίδοση ενός προγράμματος που χρησιμοποιεί επεξεργαστές γραφικών για να υπολογίσει τα αποτελέσματά του, είναι η σωστή χρησιμοποίηση της προσφερόμενης επεξεργαστικής ισχύος. Όπως έχουμε αναλύσει πιο πάνω, οι επεξεργαστές γραφικών έχουν 2 επίπεδα επεξεργασίας: τους Streaming Multiprocessors (SMs) και τους Scalar Processors (SPs). Για να επιτευχθεί υψηλή απόδοση πρέπει ο βαθμός χρησιμοποίησης αυτών των επεξεργαστών να παραμένει υψηλός (βέλτιστα μέγιστος) καθ' όλη την διάρκεια εκτέλεσης του προγράμματος.

Διάρθρωση ενός επεξεργαστή γραφικών (Fermi - GF100)

Στο επίπεδο των Streaming Multiprocessors θα πρέπει να υπάρχει ικανός αριθμός thread blocks για να "γεμίσει" όλα τα διαθέσιμα SMs του επεξεργαστή γραφικών. Όταν όλα τα thread blocks έχουν ίδιο αριθμό δεδομένων προς επεξεργασία, αρκεί ο αριθμός των thread blocks να είναι μεγαλύτερος ή ίσος από τα διαθέσιμα SMs. Όταν όμως δεν υπάρχει

ομοιογένεια στο μέγεθος των δεδομένων, τα thread blocks πρέπει να είναι αρκετά περισσότερα, ώστε τα SM να έχουν πάντα δεδομένα προς επεξεργασία και να μην μένουν ανενεργά (idle). Σε αυτή την περίπτωση δεν υπάρχει ακριβής αριθμός thread blocks που πρέπει να υπάρχουν, καθώς η τιμή αυτή εξαρτάται και από το μέγεθος της ανομοιογένειας. Ο ακριβής αριθμός μπορεί να προσδιοριστεί μόνο από πειραματικές μετρήσεις με παρακολούθηση της συμπεριφοράς του προγράμματος. Ωστόσο, αν τα thread blocks είναι τουλάχιστον διπλάσια από τον αριθμό των SMs, συνήθως εμφανίζεται υψηλή απόδοση. Σε κάθε περίπτωση, ο χωρισμός των δεδομένων προς επεξεργασία σε όσο το δυνατόν περισσότερα ανεξάρτητα thread blocks αποτελεί καλή τακτική για την μεγιστοποίηση της χρήσης των SMs.

Ο κάθε Streaming Multiprocessor, στηρίζεται στον παραλληλισμό σε επίπεδο threads για να μεγιστοποιήσει την χρησιμοποίηση των Scalar Processors. Σαν αποτέλεσμα, η χρησιμοποίηση των SPs είναι άρρητα συνδεδεμένη με τον αριθμό των warps που είναι έτοιμα προς επεξεργασία. Αυτό συμβαίνει γιατί, πριν το issue κάποιας εντολής, ο warp scheduler διαλέγει ένα warp που είναι έτοιμο να εκτελέσει την επόμενη εντολή, που σημαίνει ότι το warp έχει όλα τα δεδομένα έτοιμα προς επεξεργασία, έχοντας λύσει τις τυχόν εξαρτήσεις. Όταν ο warp scheduler έχει μεγάλο αριθμό warps διαθέσιμα, τότε μπορεί να "κρύψει" τις τυχόν καθυστερήσεις (latency) που προκύπτουν είτε λόγω εξαρτήσεων, είτε λόγω προσβάσεων στην μνήμη. Ο ακριβής αριθμός των διαθέσιμων (resident) warps, δεν μπορεί να καθοριστεί με ακρίβεια, αφού είναι συνάρτηση πολλών χαρακτηριστικών, όπως τι καθυστερήσεις εμφανίζει ο κώδικας λόγω εξαρτήσεων, πόσες προσβάσεις γίνονται στην κύρια μνήμη και φυσικά τα διαφορετικά τεχνικά χαρακτηριστικά του κάθε επεξεργαστή γραφικών. Σίγουρα πάντως απαιτούνται τουλάχιστον 2 resident warps, αν και σαφής προσδιορισμός μπορεί να γίνει με πειραματικές μετρήσεις.

Σημαντική επίπτωση στην απόδοση μπορούν να έχουν και οι εντολές ελέγχου διακλάδωσης (if, switch, do, for, while), καθώς αποκλίνουν την εκτέλεση των threads, δηλαδή ακολουθούν διαφορετικό execution path (threads diverge). Όταν αυτό συμβαίνει, τα διαφορετικά execution paths σειριοποιούνται και όταν όλα τα διαφορετικά execution paths ολοκληρωθούν, τα threads συγκλίνουν και πάλι. Κατά την σειριοποίηση των execution paths, τα SPs που έχουν αναλάβει την εκτέλεση των threads με διαφορετικό execution path απενεργοποιούνται και έτσι ο βαθμός χρησιμοποίησης τους μειώνεται. Όπως είναι φανερό, οι εντολές ελέγχου διακλάδωσης θα πρέπει να αποφεύγονται και όταν αυτό δεν είναι δυνατό, θα πρέπει να

χρησιμοποιούνται με τέτοιο τρόπο ώστε threads που ανήκουν στο ίδιο warp να μην αποκλίνουν.

Συνολικά μπορούμε να πούμε ότι ο προγραμματιστής πρέπει να καταφέρει να διατηρήσει μια ισορροπία μεταξύ του αριθμού των thread blocks και των threads. Σε κάθε περίπτωση πρέπει να εμφανίζεται όσο το δυνατόν μεγαλύτερη χρησιμοποίηση των SMs, αλλά και των SPs.

3.4.2 Συνένωση προσβάσεων στην κύρια μνήμη

Όπως έχει ήδη αναφερθεί, οι προσβάσεις στην κύρια μνήμη έχουν υψηλό κόστος, λόγω του υψηλού latency που εμφανίζουν. Συγκεκριμένα, μία πρόσβαση έχει 400-600 κύκλους ρολογίου latency. Όμως το κόστος αυτό μπορεί να μειωθεί αισθητά, αν ο προγραμματισμός βασιστεί στις ιδιαιτερότητες της αρχιτεκτονικής των επεξεργαστών γραφικών.

Οι προσβάσεις στην κύρια μνήμη γίνονται με μεταφορές μεγέθους 32, 64 και 128 bytes. Όταν εκτελείται μία εντολή σε ένα warp η οποία χρειάζεται δεδομένα από την κύρια μνήμη, οι προσβάσεις στην κύρια μνήμη συνενώνονται σε μία ή περισσότερες προσβάσεις στην κύρια μνήμη, ανάλογα με το μέγεθος των δεδομένων που ζητούνται, αλλά και την κατανομή τους στην κύρια μνήμη. Έτσι, όταν τα δεδομένα δεν είναι καλά οργανωμένα στην μνήμη, οι προσβάσεις δεν συνενώνονται με αποτέλεσμα να γίνονται πολλές μεταφορές άχρηστων δεδομένων, που η κάθε μία έχει το κόστος σε latency που αναφέρεται πιο πάνω. Με τον τρόπο αυτό η απόδοση της κύριας μνήμης μειώνεται πολύ, έχοντας σημαντικό αντίκτυπο στην συνολική απόδοση του προγράμματος. Για παράδειγμα, αν κάθε thread χρειάζεται 4 bytes από την κύρια μνήμη, τα οποία δεν είναι σωστά ταξινομημένα, δημιουργείται μεταφορά 32 bytes για κάθε ένα thread και έτσι η απόδοση της μνήμης μειώνεται κατά 8 φορές.

Για να επιτευχθεί όσο το δυνατόν καλύτερη συνένωση στις προσβάσεις στην κύρια μνήμη (memory coalescing) πρέπει τα διαδοχικά threads του warp να ζητούν διαδοχικές θέσεις μνήμης. Με αυτόν τον τρόπο, όλες οι προσβάσεις στην μνήμη συνενώνονται και το συνολικό latency πρόσβασης ελαχιστοποιείται. Για να επιτευχθεί κάτι τέτοιο, αρκετές φορές θα πρέπει να αλλάξει ο τρόπος αποθήκευσης των δεδομένων στην μνήμη, έτσι ώστε η θέση να είναι συνάρτηση του thread ID. Ο τρόπος αποθήκευσης δεν είναι συγκεκριμένος, αλλά

διαφοροποιείται ανάλογα με τον αλγόριθμο.

3.4.3 Χρήση της γρήγορης on-chip μνήμης

Η shared memory είναι on-chip, πράγμα που σημαίνει ότι είναι πολύ γρήγορη. Συγκεκριμένα, είναι ταχύτερη από οποιαδήποτε άλλη διαθέσιμη μνήμη, και όταν χρησιμοποιηθεί σωστά προσφέρει επιδόσεις συγκρίσιμες με το register file.

Όπως αναφέρθηκε και πιο πάνω, η shared memory έχει 32 banks τα οποία είναι οργανωμένα με τέτοιο τρόπο ώστε διαδοχικά 32-bit words να αναθέτονται σε διαδοχικά banks. Η πρόσβαση σε αυτά τα banks μπορεί να γίνει ταυτόχρονα με μέγιστο εύρος 32 bits ανά 2 κύκλους ρολογιού για το κάθε bank. Έτσι, όταν οι διευθύνσεις οι οποίες ζητούνται για εγγραφή ή ανάγνωση βρίσκονται σε διαφορετικά banks η απόδοση της shared memory μεγιστοποιείται και ισούται με 32 * 32 bits ανά 2 κύκλους ρολογιού. Όταν όμως οι αιτήσεις μνήμης συμπίπτουν στο ίδιο bank, τότε οι προσβάσεις σειριοποιούνται και η απόδοση μειώνεται.

Λόγω της υψηλής απόδοσης που παρουσιάζει η shared memory, επιβάλλεται η χρήση της για δεδομένα που χρησιμοποιούνται συχνά. Ο συνήθης τρόπος χρήσης της ξεκινάει με την αντιγραφή των δεδομένων από την global στην shared memory, η διενέργεια των υπολογισμών χρησιμοποιώντας της shared memory, εκμεταλλευόμενοι τα πολλαπλά banks, και όταν τα δεδομένα δεν χρειάζονται άλλο, η αντιγραφή από την shared πίσω στην global memory. Όταν το μέγεθος της shared memory δεν επαρκεί για τα δεδομένα, θεμιτή είναι η χρήση των read-only constant ή texture memory, οι οποίες μπορεί να βρίσκονται στην κύρια μνήμη, όμως είναι cached και κατά περιπτώσεις προσφέρουν μεγαλύτερη απόδοση από την global memory.

ΚΕΦΑΛΑΙΟ 4

ΠΑΡΑΛΛΗΛΟΠΟΙΗΣΗ ΤΟΥ ΑΛΓΟΡΙΘΜΟΥ SMITH - WATERMAN

4.1 Ο αλγόριθμος Smith - Waterman στην πράξη

Η στοίχιση δύο ακολουθιών μπορεί να είναι δύσκολη να υπολογιστεί από τον άνθρωπο, όμως για έναν υπολογιστή είναι υπόθεση ελάχιστου χρόνου. Αν έπρεπε να στοιχιστούν λίγες μόνο ακολουθίες, δεν θα υπήρχε λόγος για τόσο μεγάλη ανάλυση του αλγορίθμου και των πιθανών υλοποιήσεών του. Όμως οι βιολόγοι ερευνητές, χρειάζεται να στοιχίζουν χιλιάδες ακολουθίες καθημερινά. Συγκεκριμένα, ακολουθίες όταν ανακαλύπτουν κάποιες ακολουθίες, οι οποίες είναι άγνωστες, τις στοιχίζουν με τις ήδη γνωστές ακολουθίες για να βρουν όμοιες περιοχές, οι οποίες μπορούν να τους δώσουν πολύ σημαντικές πληροφορίες, όπως αναλύθηκε και σε προηγούμενο κεφάλαιο.

Οι ήδη γνωστές ακολουθίες είναι αρκετές εκατοντάδες χιλιάδες. Συνήθως είναι αποθηκευμένες σε κάποια βάση δεδομένων που χρησιμοποιείται ως μέτρο σύγκρισης για τις προς στοίχιση ακολουθίες. Οι βάσεις δεδομένων αυτές αποτελούν πολύ σημαντικό εργαλείο καθώς η επιστημονική κοινότητα μπορεί να έχει ένα κοινό σημείο αναφοράς. Στις βάσεις προστίθονται συνεχώς οι νέες ακολουθίες που αναγνωρίζονται και όπως είναι λογικό έχουν διόλου ευκαταφρόνητο μέγεθος, αν αναλογιστεί κανείς ότι χρησιμοποιούνται για να

αποθηκεύσουν απλά κάποια strings και το μέγεθος τους είναι αρκετές εκατοντάδες megabytes. Οι πιο γνωστές εξ αυτών είναι:

- NCBI – American National Center of Biotechnology Information
- EBI – European Bioinformatics Istitute
- DDBJ – DNA Data Bank of Japan
- UniProt – Universal Protein Resource

.Οι προς στοίχιση ακολουθίες συμβολίζονται με χαρακτήρες του αγγλικού αλφάβητου. Συγκεκριμένα χρησιμοποιούνται 20 γράμματα για τον συμβολισμό των αμινοξέων και άλλα 4 για τον συμβολισμό των νουκλεϊκών οξέων. Το γεγονός αυτό υποδεικνύει τον τρόπο χειρισμού του πίνακα αντικατάστασης. Έτσι, ο πίνακας ταξινομήθηκε σε αλφαβητική σειρά και εισάχθηκαν κενοί χαρακτήρες για τους μη χρησιμοποιούμενους χαρακτήρες του αλφάβητου. Ο λόγος που έγινε αυτό είναι η ευκολία πρόσβασης του πίνακα αντικατάστασης. Έτσι δεν χρειάζεται καμία τροποποίηση στον συμβολισμό των ακολουθιών, οι οποίες συμβολίζονται με χαρακτήρες ενώ η πρόσβαση στον πίνακα αντικατάστασης γίνεται με την εξής συνάρτηση:

$$(i\,,\,j)=(ascii(A_i)-ascii('A'),ascii(B_i)-ascii('A'))$$

Οι προς στοίχιση ακολουθίες αναφέρονται ως ένα query στην ήδη υπάρχουσα database (βάση δεδομένων) ακολουθιών. Ο αριθμός των προς στοίχιση ακολουθιών αναφέρεται ως το μέγεθος του query, ενώ το πλήθος των ακολουθιών που υπάρχουν στην βάση αναφέρεται ως το μέγεθος της database. Το μεγάλο μέγεθος του query που γίνεται (συνήθως είναι μεγαλύτερο από 100), αποτελεί ένα ακόμα σημείο παραλληλοποίησης, το οποίο όπως εξηγούμε στην συνέχεια είναι πολύτιμο για την επίτευξη υψηλής απόδοσης.

4.2 Υλοποίηση σε κεντρικούς επεξεργαστές (CPU)

Για την υλοποίηση του αλγορίθμου Smith – Waterman σε πολυπύρηνους επεξεργαστές χρησιμοποιήθηκε το API του γνωστού OpenMP (Open MultiProcessing). Το OpenMP υποστηρίζει παράλληλο προγραμματισμό χρησιμοποιώντας C, C++ ή Fortran. Ουσιαστικά αποτελεί μια επέκταση για τις γλώσσες αυτές, η οποία χρησιμοποιώντας κατάλληλες οδηγίες στον μεταγλωττιστή (compiler directives) αλλά και μια βιβλιοθήκη χρόνου εκτέλεσης (run-time library), δημιουργεί threads που εκμεταλλεύονται τους πυρήνες των σύγχρονων επεξεργαστών. Η υλοποίηση έγινε σε γλώσσα C++.

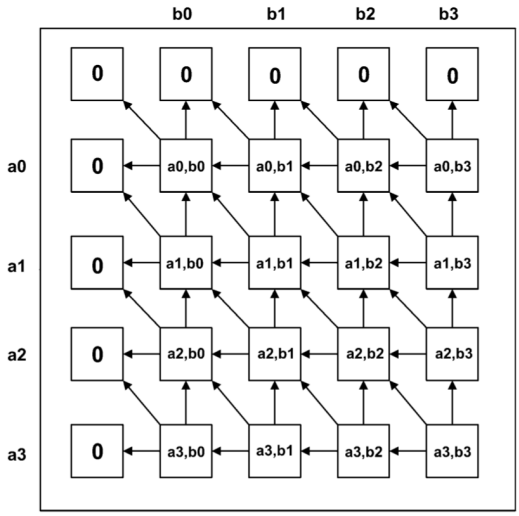
Η λογική που ακολουθήθηκε για την παραλληλοποίηση είναι task centric. Από την στιγμή που πρέπει να στοιχιστούν οι πολλές ακολουθίες του query με τις ακόμα περισσότερες ακολουθίες της database, παραλληλοποιήσαμε στο επίπεδο των queries. Στη συνέχεια παρουσιάζεται ο ψευδοκώδικας που δείχνει το σημείο παραλληλοποίησης.

```
parallel for all q in queries
     for all d in database
          compute_Smith_Waterman (q, d);
     end for
end parallel for
```

Σε κάθε thread ανατίθεται η στοίχιση μιας ακολουθίας από το query με όλες τις ακολουθίες της database. Ο υπολογισμός του πίνακα βαθμολογίας του αλγορίθμου Smith – Waterman γίνεται σειριακά. Η υλοποίηση αυτής της coarse-grained παραλληλοποίησης επιλέχθηκε για δύο λόγους. Καταρχάς, το μεγάλο πλήθος των προς στοίχιση ακολουθιών εξασφαλίζει την δημιουργία ικανού αριθμού threads για να εκμεταλλευτούν όλους τους πυρήνες οποιουδήποτε σύγχρονου συστήματος. Επίσης, η σειριακή εκτέλεση του υπολογιστικού πυρήνα του αλγορίθμου Smith – Waterman εξασφαλίζει την ύπαρξη όσο το δυνατόν μεγαλύτερου data locality, το οποίο είναι κλειδί για υψηλή επίδοση.

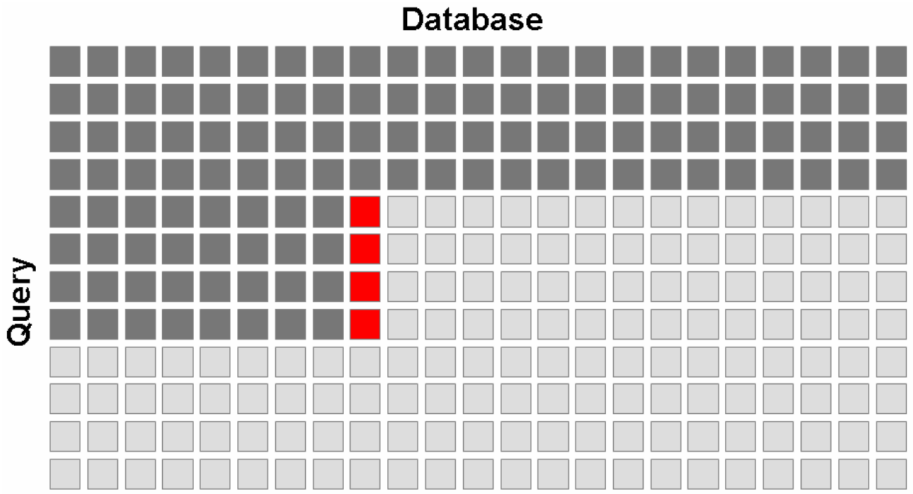
Κατά την σειριακή εκτέλεση του υπολογιστικού πυρήνα του αλγορίθμου Smith – Waterman, η πρόσβαση στους πίνακες υιοθετεί row-wise λογική. Έτσι υπολογίζοντας την μία γραμμή μετά την άλλη, επιλύονται οι εξαρτήσεις δεδομένων από το πάνω και το πάνω-αριστερά κελί. Υπολογίζοντας το στοιχείο της κάθε γραμμής από δεξιά προς τα αριστερά επιλύεται η από

αριστερά εξάρτηση δεδομένων. Τέλος, η αριθμητική δεικτών που απαιτείται για την πρόσβαση των πινάκων είναι απλή και χωρίς πολλούς υπολογισμούς.



Οι εξαρτήσεις δεδομένων

Στο σχήμα που ακολουθεί φαίνεται η σειρά με την οποία στοιχίζονται όλα τα ζευγάρια ακολουθιών. Με γκρι σκούρο απεικονίζονται τα ζευγάρια που έχουν ήδη στοιχιστεί, με γκρί ανοιχτό τα ζευγάρια που δεν έχουν στοιχιστεί ακόμα, ενώ με κόκκινο τα ζευγάρια που στοιχίζονται σε κάποια χρονική στιγμή. Στο παράδειγμα αυτό θεωρείται ότι υπάρχουν διαθέσιμοι 4 επεξεργαστικοί πυρήνες, οπότε 4 ακολουθίες του query στοιχίζονται ταυτόχρονα με μία ακολουθία της database.



Μοντέλο παραλληλοποίησης για CPU

Υλοποιήθηκαν δύο εκδόσεις, μία που κάνει χρήση ενός κανονικού πίνακα αντικατάστασης

και μία που είναι παραλλαγή της πρώτης, έχει μόνο δύο σκορ, ένα επιβράβευσης το οποίο προστίθεται αν οι χαρακτήρες που συγκρίνονται είναι ίδιοι και ένα ποινής που αφαιρείται αν οι χαρακτήρες διαφέρουν. Ο λόγος της δεύτερης υλοποίησης είναι ο έλεγχος της επίπτωσης που έχει ένα conditional branch μέσα σε έναν απαιτητικό υπολογιστικό πυρήνα σαν τον Smith – Waterman. Η υλοποίηση με τον πίνακα αντικατάστασης αποφεύγει αυτό το branch καθώς το σκορ πάντα προστίθεται στο αποτέλεσμα και είναι θετικό όταν οι χαρακτήρες είναι ίδιοι και αρνητικό όταν διαφέρουν. Το μειονέκτημά του είναι ότι ο πίνακας αντικατάστασης δεσμεύει χώρο στην κρυφή μνήμη.

4.3 Υλοποίηση σε επεξεργαστές γραφικών (GPU)

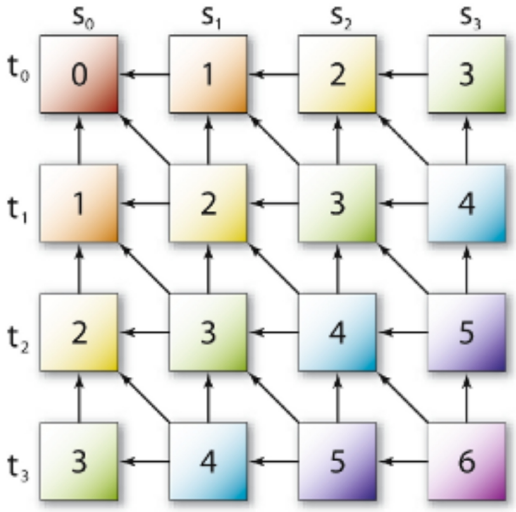
Η υλοποίηση του αλγορίθμου Smith – Waterman σε πολυνηματικούς επεξεργαστές γραφικών βασίστηκε στην αρχιτεκτονική CUDA της Nvidia και έγινε χρησιμοποιώντας το API της OpenCL (Open Computing Language), που βασίζεται στην γλώσσα C99 προστίθοντάς της τα απαραίτητα extentions για την συγγραφή υπολογιστικών πυρήνων.

Όπως αναφέρθηκε και στο προηγούμενο κεφάλαιο, προκειμένου να επιτευχθεί υψηλή απόδοση όταν χρησιμοποιούνται επεξεργαστές γραφικών, η παραλληλοποίηση πρέπει να γίνει σε δύο επίπεδα. Έτσι η λογική που ακολουθήθηκε ήταν ταυτόχρονα task-parallel και data-parallel. Έτσι παραλληλοποιήσαμε τον κώδικα τόσο στο επίπεδο των queries, όσο και στο εσωτερικά στον υπολογιστικό πυρήνα του αλγορίθμου Smith – Waterman. Στη συνέχεια παρουσιάζεται ο ψευδοκώδικας που δείχνει τα δύο αυτά σημεία παραλληλοποίησης.

```
for all d in database
     //generate thread blocks
     SM_parallel for all q in queries
          //generate threads inside each thread block
          SP_parallel_compute_Smith_Waterman (q, d);
     end SM_parallel for
end for
```

Για κάθε ακολουθία της database, δημιουργούνται query_size thread blocks, όπου query_size είναι ο αριθμός των ακολουθιών που έχει το query. Αυτό σημαίνει ότι σε κάθε thread block ανατίθεται η στοίχιση ενός ζευγαριού ακολουθιών. Τα thread blocks αυτά μοιράζονται στα διαθέσιμα SMs. Το κάθε thread block αποτελείται από n threads, όπου n είναι το μήκος της μεγαλύτερης αντιδιαγωνίου. Ο υπολογισμός του πίνακα βαθμολογίας του αλγορίθμου Smith – Waterman γίνεται παράλληλα από τα threads του thread block, που μοιράζονται στα SPs του κάθε SM. Με τον τρόπο αυτό έχουμε ταυτόχρονα coarse-grained παραλληλοποίηση των στοιχίσεων στα SMs και fine-grained παραλληλοποίηση του υπολογισμού του πίνακα βαθμολογίας στα SPs. Το μεγάλο πλήθος των προς στοίχιση ακολουθιών εξασφαλίζει την δημιουργία ικανού αριθμού thread blocks για να εκμεταλλευτούν όλους τους Streaming Multiprocessors και το μεγάλο μέγεθος του πίνακα βαθμολογίας εξασφαλίζει την δημιουργία ικανού αριθμού threads για να εκμεταλλευτούν όλους τους Scalar Processors. Έτσι πετυχαίνουμε υψηλό βαθμό χρησιμοποίησης σε όλες τις επεξεργαστικές μονάδες της κάρτας γραφικών.

Κατά τον παράλληλο υπολογισμό του πίνακα βαθμολογίας η πρόσβαση στα κελιά του πίνακα γίνεται με την μέθοδο των αντιδιαγωνιών. Έτσι, υπολογίζοντας την μία αντιδιαγώνιο μετά την άλλη, επιλύονται οι εξαρτήσεις δεδομένων που υπάρχουν με τις δύο προηγούμενες αντιδιαγωνίους, και επίσης τα στοιχεία της κάθε αντιδιαγωνίου μπορούν να υπολογιστούν παράλληλα από τα SPs. Αυτό φαίνεται καθαρά στο σχήμα που ακολουθεί.

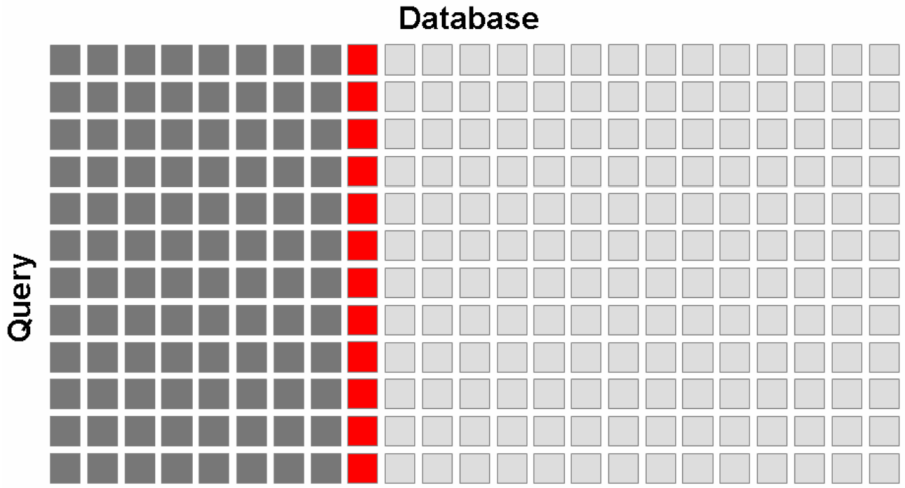


Ο υπολογισμός του πίνακα βαθμολογίας κατά αντιδιαγωνίους

Η αντιδιαγώνιος 3 υπολογίζεται μετά την 1 και την 2, άρα δεν υπάρχει θέμα εξάρτησης δεδομένων. Επίσης το κάθε thread αναλαμβάνει ένα στοιχείο της αντιδιαγωνίου, δίνοντας την

δυνατότητα παράλληλης εκτέλεσης.

Στο σχήμα που ακολουθεί φαίνεται η σειρά με την οποία στοιχίζονται όλα τα ζευγάρια ακολουθιών. Με γκρι σκούρο απεικονίζονται τα ζευγάρια που έχουν ήδη στοιχιστεί, με γκρί ανοιχτό τα ζευγάρια που δεν έχουν στοιχιστεί ακόμα, ενώ με κόκκινο τα ζευγάρια για τα οποία έχουν δημιουργηθεί thread blocks και στοιχίζονται σε κάποια χρονική στιγμή.



Μοντέλο παραλληλοποίησης για GPU

Υλοποιήθηκαν 3 διαφορετικές εκδόσεις, με δύο παραλλαγές για την κάθε μια, χρησιμοποιώντας πίνακα αντικατάστασης ή conditional branch. Η κάθε έκδοση είναι βελτίωση της προηγούμενης και ο σκοπός τους είναι να αναδείξουν ποια τεχνικά χαρακτηριστικά του επεξεργαστή γραφικών προσφέρουν τα μεγαλύτερα οφέλη σε απόδοση. Έτσι έχουμε τις εξής εκδόσεις:

- *Memory coalescing.* Σε αυτήν την έκδοση η μόνη τεχνική που χρησιμοποιήθηκε ήταν η συνένωση των προσβάσεων στην global memory. Για να επιτευχθεί αυτό, έπρεπε να αλλάξει ριζικά ο τρόπος αποθήκευσης των πινάκων στη μνήμη. Συγκεκριμένα, τα κελιά του πίνακα που ανήκουν στην ίδια αντιδιαγώνιο πρέπει να βρίσκονται σε διαδοχικές θέσεις μνήμης, οπότε οι τιμές αποθηκεύονται με τη σειρά που ορίζουν οι αντιδιαγώνιοι. Ένα παράδειγμα για την καλύτερη κατανόηση αυτού του τρόπου αποθήκευσης φαίνεται στο παρακάτω σχήμα. Τα στοιχεία του πίνακα που βρίσκονται στην ίδια διαγώνιο, στην πραγματικότητα αποθηκεύονται στις διαδοχικές θέσεις μνήμης που ορίζουν οι δείκτες που φαίνονται στο κάθε κελί.

	0	C	A	G	C	C	U	C	G	C	U	U	A	G
0	0	1	3	6	10	15	21	28	36	45	55	66		
A	2	4	7	11	16	22	29	37	46	56	67			
A	5	8	12	17	23	30	38	47	57	68				
U	9	13	18	24	31	39	48	58	69					
G	14	19	25	32	40	49	59	70						
C	20	26	33	41	50	60	71							
C	27	34	42	51	61	72								
A	35	43	52	62	73									
U	44	53	63	74										
U	54	64	75											
G	65	76												
C	77													
C														
G														
G														

Τρόπος αποθήκευσης για την επίτευξη memory coalescing

- Χρήση της *Shared Memory*. Εκτός της συνένωσης των προσβάσεων στην κύρια μνήμη, τα αποτελέσματα των 2 προηγούμενων αντιδιαγωνίων αποθηκεύονται στην shared memory. Συγκεκριμένα, όταν υπολογίζεται η αντιδιαγώνιος n, οι τιμές των αντιδιαγωνίων n-1 και n-2 βρίσκονται στην ταχύτατη shared memory. Έτσι εκμεταλλεύεται το data reuse που γίνεται στον κάθε υπολογισμό αντιδιαγωνίου και μειώνονται οι προσβάσεις στην κύρια μνήμη. Όταν υπολογιστούν όλες οι τιμές της αντιδιαγωνίου n, οι τιμές της αντιδιαγωνίου n-2 που μέχρι τώρα βρίσκονταν στην shared memory δεν χρειάζονται άλλο, οπότε γράφονται πίσω στην global memory και την θέση τους παίρνουν οι τιμές της αντιδιαγωνίου n. Έτσι, στον επόμενο υπολογισμό της αντιδιαγωνίου n+1, στην shared memory θα υπάρχουν τα απαραίτητα στοιχεία των 2 προηγούμενων αντιδιαγωνίων n και n-1.

- Περαιτέρω *μείωση των εγγραφών στην Global Memory*. Στην προηγούμενη έκδοση τα στοιχεία της προ-προηγούμενης αντιδιαγωνίου και για τους 3 πίνακες αντιγράφονται στην κύρια μνήμη. Όμως οι τιμές των βοηθητικών πινάκων E και F, δεν χρησιμοποιούνται ποτέ ξανά, οπότε δεν χρειάζονται. Έτσι σε αυτήν την έκδοση, μετά το πέρας υπολογισμού των στοιχείων της n αντιδιαγωνίου, τα στοιχεία της n-2 αντιδιαγωνίου μόνο του πίνακα βαθμολογίας αντιγράφονται πίσω στην global memory. Τα στοιχεία των βοηθητικών πινάκων E και F απλά αντικαθιστώνται με τις νέες τιμές. Με τον τρόπο αυτό υποτριπλασιάζονται τις εγγραφές στην αργή global memory, και αναδεικνύεται το ποσοστό που το memory bandwidth επηρεάζει την απόδοση του αλγορίθμου.

ΚΕΦΑΛΑΙΟ 5

ΠΕΙΡΑΜΑΤΙΚΕΣ ΜΕΤΡΗΣΕΙΣ

5.1 Σύστημα δοκιμών και διαδικασία μετρήσεων

Το σύστημα δοκιμών που χρησιμοποιήθηκε κατά την διαδικασία των πειραματικών μετρήσεων ήταν εξοπλισμένο με 2 επεξεργαστές Intel Xeon X5650, 48 gigabytes DDR3 μνήμης καθώς και μία κάρτα γραφικών Nvidia Tesla M2050 computing module. Τα τεχνικά χαρακτηριστικά των επεξεργαστών αλλά και της κάρτας γραφικών παρουσιάζονται αναλυτικά στους παρακάτω πίνακες.

Τεχνικά Χαρακτηριστικά επεξεργαστή	
Μοντέλο	Intel Xeon X5650
Πυρήνες	6
Threads	12
Συχνότητα	2.67 GHz
Intel QPI Speed	2 x 6.4 GT/s
L1 Instruction Cache	6 x 32 KB
L1 Data Cache	6 x 32 KB
L2 Unified Cache	6 x 256 KB
L3 Unified Cache	12 MB
Ταχύτητα μνήμης	1333 Mhz DDR3 triple channel

Πίνακας τεχνικών χαρακτηριστικών επεξεργαστή

Ο Intel Xeon X5650 είναι βασισμένος στην αρχιτεκτονική Westmere, που είναι βελτίωση της πολύ επιτυχημένης αρχιτεκτονικής Nehalem και αποτελεί έναν από τους πιο γρήγορους

επεξεργαστές της αγοράς. Ο επεξεργαστής διαθέτει και Turbo Core (τεχνική για αύξηση της συχνότητας λειτουργίας του ανάλογα με το φορτίο), το οποίο όμως είχε απενεργοποιηθεί για να μην επηρεάζει την διαδικασία των μετρήσεων.

Τεχνικά Χαρακτηριστικά κάρτας γραφικών	
Μοντέλο	Nvidia Tesla M2050
Πυρήνας	GF100 (Fermi)
Streaming Multiprocessors	14
Scalar Processors (CUDA cores)	448 (14 x 32)
Compute Capability	2.0
Ταχύτητα πυρήνα	575 MHz
Ταχύτητα cuda cores	1150 MHz
Μέγεθος μνήμης	3072 MB ECC
Εύρος μνήμης	384 bit
Ταχύτητα μνήμης	3092 MHz GDDR5

Πίνακας τεχνικών χαρακτηριστικών κάρτας γραφικών

Το λειτουργικό σύστημα ήταν Ubuntu Linux 64 bit. Ο Linux kernel ήταν ο 3.2.0-29. Η έκδοση του CUDA Toolkit ήταν 4.2.1 και της OpenCL 1.1. Η έκδοση του gcc ήταν η 4.6.3.

Για την ακρίβεια των μετρήσεων ακολουθήσαμε την εξής διαδικασία: Όλοι οι υπολογισμοί των αποτελεσμάτων γινόταν 10 φορές και οι συνολικοί χρόνοι διαιρούταν με το 10. Η κάθε μέτρηση επαναλαμβανόταν 5 φορές και από αυτές τις τιμές επιλεγόταν η διάμεσος (median).

5.2 Σύγκριση απόδοσης των διάφορων υλοποιήσεων

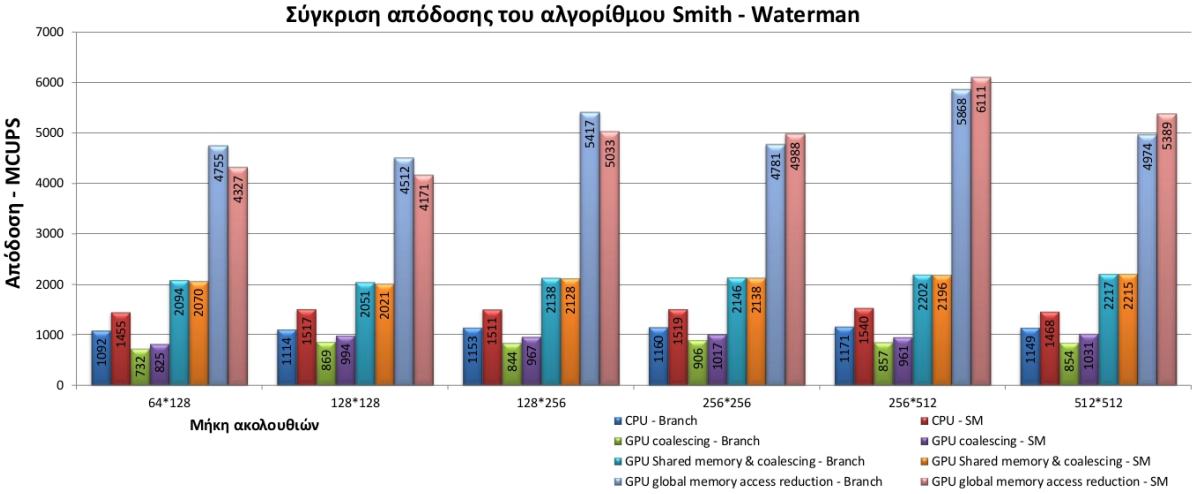
Για την μέτρηση της απόδοσης των διαφόρων υλοποιήσεων του αλγορίθμου Smith – Waterman, χρησιμοποιείται σαν μονάδα μέτρησης το CUPS (Cell Updates Per Second), το οποίο συμβολίζει το πλήθος των κελιών του πίνακα βαθμολογίας που υπολογίζονται στην μονάδα του χρόνου. Οι βοηθητικοί πίνακες E και F αγνοούνται στην μέτρηση αυτή. Έτσι, για τον υπολογισμό της απόδοσης χρησιμοποιείται ο παρακάτω τύπος:

$$CUPS = \frac{\textit{μήκος ακολουθίας query x μήκος ακολουθίας database}}{\textit{απαιτούμενος χρόνος}}$$

Ο παραπάνω τύπος μας δίνει την απόδοση για την στοίχιση ενός ζευγαριού ακολουθιών.

Όταν στοιχίζονται πολλές ακολουθίες, ο αριθμητής μετασχηματίζεται κατάλληλα ώστε να ισούται με το σύνολο όλων των κελιών που υπολογίζονται (όλων των πιθανών ζευγαριών). Στην πράξη σαν μονάδα μέτρησης χρησιμοποιείται το MCUPS (Mega CUPS) που ισούται με CUPS / 10^6.

Αρχικά παρουσιάζεται η απόδοση που επιτυγχάνει κάθε μία από τις υλοποιήσεις που περιγράφτηκαν στο κεφάλαιο 4, για διάφορα συνηθισμένα μήκη ακολουθιών. Πρέπει να διευκρινιστεί ότι οι υπολογισμοί στους CPU έγιναν με 6 threads στον κάθε επεξεργαστή με σκοπό να γεμίσουν και οι συνολικά 12 πυρήνες των δύο επεξεργαστών, χωρίς όμως να γίνεται χρήση του Hyper-Threading. Από το διάγραμμα αυτό σχηματίζεται μια γενική εικόνα της απόδοσης των διάφορων υλοποιήσεων.



Διάγραμμα σύγκρισης απόδοσης των διάφορων υλοποιήσεων

Όσον αφορά λοιπόν την υλοποίηση σε CPU, φαίνεται ξεκάθαρα ότι η έκδοση με το conditional branch επιτυγχάνει μικρότερη επίδοση (περίπου 1150 MCUPS) από την έκδοση που χρησιμοποιεί πίνακα αντικατάστασης (περίπου 1500 MCUPS). Αυτό συμβαίνει επειδή το μικρό μέγεθος του πίνακα αντικατάστασης του επιτρέπει να ανέβει ψηλά στην ιεραρχία κρυφής μνήμης, άρα η πρόσβαση σε αυτόν να έχει μηδαμινό κόστος. Συγκεκριμένα ο πίνακας αντικατάστασης έχει 26 x 26 ακεραίους (32 bit), άρα έχει μέγεθος 26 x 26 x 4 bytes = 2704 bytes. Η L1 data cache των επεξεργαστών που χρησιμοποιήθηκαν έχει μέγεθος 32 KB πράγμα που σημαίνει ότι πιθανότατα, ο πίνακας αντικατάστασης είναι cached στην ταχύτατη L1. Παρατηρούμε λοιπόν ότι ένα και μόνο conditional branch μέσα σε έναν "βαρύ" υπολογιστικό πυρήνα αρκεί για να μειώσει την επίδοση, και καλό είναι να αποφεύγεται.

Θετική εντύπωση προκαλεί η σταθερότητα της απόδοσης της υλοποίησης σε CPU, ανεξαρτήτως του μήκους των ακολουθιών που στοιχίζονται. Η μόνη εξαίρεση είναι η στοίχιση ακολουθιών μήκους 512. όπου η απόδοση παρουσιάζει μια μικρή πτώση. Σε αυτή την περίπτωση, οι τρεις πίνακες που υπολογίζονται έχουν συνολικό μέγεθος 3 x 513 x 513 x 4 bytes = 3 MB. Άρα όταν χρησιμοποιούνται 6 threads ανά επεξεργαστή, υπολογίζονται συνολικά 6 x 3 = 18 MB δεδομένων, τα οποία είναι μεγαλύτερα από την L3 cache που έχει μέγεθος 12 MB. Το γεγονός αυτό αναγκάζει τον επεξεργαστή να κάνει κάποιες προσβάσεις στην κύρια μνήμη με αποτέλεσμα μια μείωση της τάξης του 5% στην επίδοση. Η μείωση αυτή είναι σχετικά μικρή και αυτό οφείλεται στο data locality που εμφανίζουν οι υπολογισμοί και αποτρέπει μεγάλο μέρος των misses στην cache.

Περνώντας στην πρώτη υλοποίηση σε GPU, ο πυρήνας που κάνει χρήση μόνο της συνένωσης των προσβάσεων στην κύρια μνήμη δεν καταφέρνει να ξεπεράσει την απόδοση των δύο επεξεργαστών (αν και ξεπερνάει την απόδοση του ενός). Ο λόγος της σχετικά χαμηλής απόδοσης είναι οι πολλές προσβάσεις στην αργή global memory που ακόμα και αν είναι συνενωμένες, δεν μπορούν να προσφέρουν την απόδοση που προσφέρουν οι on-chip μνήμες. Αξίζει να σημειώσουμε ότι η M2050 σε αντίθεση με τις παλιότερες γενιές, είναι εξοπλισμένη σε κάποιες μικρές L1 και L2 caches, οι οποίες όμως σε καμία περίπτωση δεν πλησιάζουν το συνολικό μέγεθος που έχουν οι caches των επεξεργαστών. Η έκδοση που χρησιμοποιεί πίνακα αντικατάστασης είναι ταχύτερη, προσφέροντας επίδοση περίπου 1000 MCUPS.

Η εικόνα αλλάζει αρκετά όταν γίνεται χρήση της shared memory. Η μνήμη αυτή έχει επιδόσεις cache αφού βρίσκεται μέσα στους Streaming Multiprocessors, με τη διαφορά όμως ότι ο προγραμματιστής ορίζει τι δεδομένα θα γραφτούν σε αυτή. Έτσι όταν χρησιμοποιηθεί για να αποθηκεύσει τα δεδομένα που επαναχρησιμοποιούνται, δηλαδή για τις δύο προηγούμενες αντιδιαγωνίους, η απόδοση υπερδιπλασιάζεται ξεπερνώντας τα 2000 MCUPS. Οι δύο παραλλαγές εδώ προσφέρουν σχεδόν την ίδια επίδοση, η οποία είναι σταθερή για όλα τα μήκη των ακολουθιών.

Με την τελευταία βελτιστοποίηση η επίδοση ανεβαίνει κατά πολύ, φτάνοντας μέχρι και 3 φορές μεγαλύτερη σε σύγκριση με την προηγούμενη έκδοση. Η διαφορά των δύο αυτών εκδόσεων είναι ότι η βελτιστοποιημένη δεν γράφει πίσω στην global memory τους βοηθητικούς πίνακες E και F, άρα οι προσβάσεις υποτριπλασιάζονται. Το γεγονός ότι ο υποτριπλασιασμός των προσβάσεων στην global memory τριπλασιάζει την επίδοση είναι

σαφής ένδειξη ότι ο υπολογιστικός πυρήνας του αλγορίθμου Smith-Waterman περιορίζεται πολύ από την ταχύτητα της μνήμης, άρα μπορεί να χαρακτηριστεί ως memory-bound αλγόριθμος.

Αξίζει να αναφέρουμε ότι η έκδοση που χρησιμοποιεί τον πίνακα αντικατάστασης, είναι ταχύτερη όταν στοιχίζονται μεγάλες ακολουθίες, αλλά πιο αργή όταν στοιχίζονται μικρές. Αυτό συμβαίνει για τον εξής λόγο. Τα accesses στον πίνακα αντικατάστασης είναι εντελώς τυχαία, αφού εξαρτώνται από τους χαρακτήρες που συγκρίνονται κάθε φορά. Άρα δεν μπορούν να συνενωθούν και για να κρυφτεί το μεγάλο latency που εισάγουν τα accesses αυτά, πρέπει να υπάρχουν πολλά threads ενεργά. Για το λόγο αυτό παρατηρούμε και στο διάγραμμα ότι για συγκρίσεις μεγαλύτερες του 256 με 256 (άρα 256 ενεργά threads) η έκδοση που χρησιμοποιεί τον πίνακα αντικατάστασης προσφέρει την υψηλότερη επίδοση. Τέλος να αναφέρουμε ότι η απόδοση της συγκεκριμένης υλοποίησης παρουσιάζει αρκετή διακύμανση ανάλογα με το μήκος των ακολουθιών που συγκρίνονται, κάτι που θα αναλυθεί πιο κάτω.

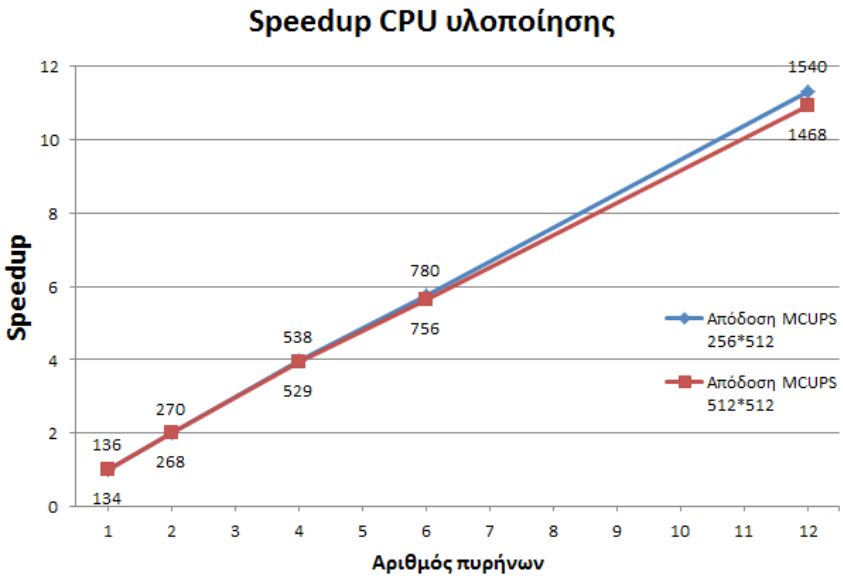
5.3 Ανάλυση συμπεριφοράς των υλοποιήσεων

Στη συνέχεια γίνεται προσπάθεια για περαιτέρω ανάλυση των αποτελεσμάτων με σκοπό την εξαγωγή σίγουρων συμπερασμάτων. Η κάθε υλοποίηση εξετάζεται ξεχωριστά και παρουσιάζονται επιπλέον μετρήσεις που θα βοηθήσουν στην αποσαφήνιση της συμπεριφοράς τους.

5.3.1 Παράλληλη υλοποίηση σε CPU

Κατά την διάρκεια των μετρήσεων, παρατηρήθηκε σταθερή συμπεριφορά της συγκεκριμένης υλοποίησης. Η απόδοση είχε μικρές διακυμάνσεις, ανεξαρτήτως των μεγεθών εισόδου. Στο επόμενο διάγραμμα παρουσιάζουμε τη επιτάχυνση που επιτεύχθηκε στην παράλληλη

υλοποίηση σε CPU. Τα νούμερα που παρουσιάζονται αφορούν στοιχίσεις 128 queries των 256 χαρακτήρων (μπλε γραμμή) και 512 χαρακτήρων (κόκκινη γραμμή) το καθένα, πάνω σε database που πο ακολουθίες της έχουν μήκος 512 χαρακτήρες. Τα 1, 2, 4 και 6 threads αφορούν μετρήσεις πάνω σε έναν επεξεργαστή, ενώ τα 12 threads κατανέμονται ανά 6 στους δύο διαθέσιμους επεξεργαστές.



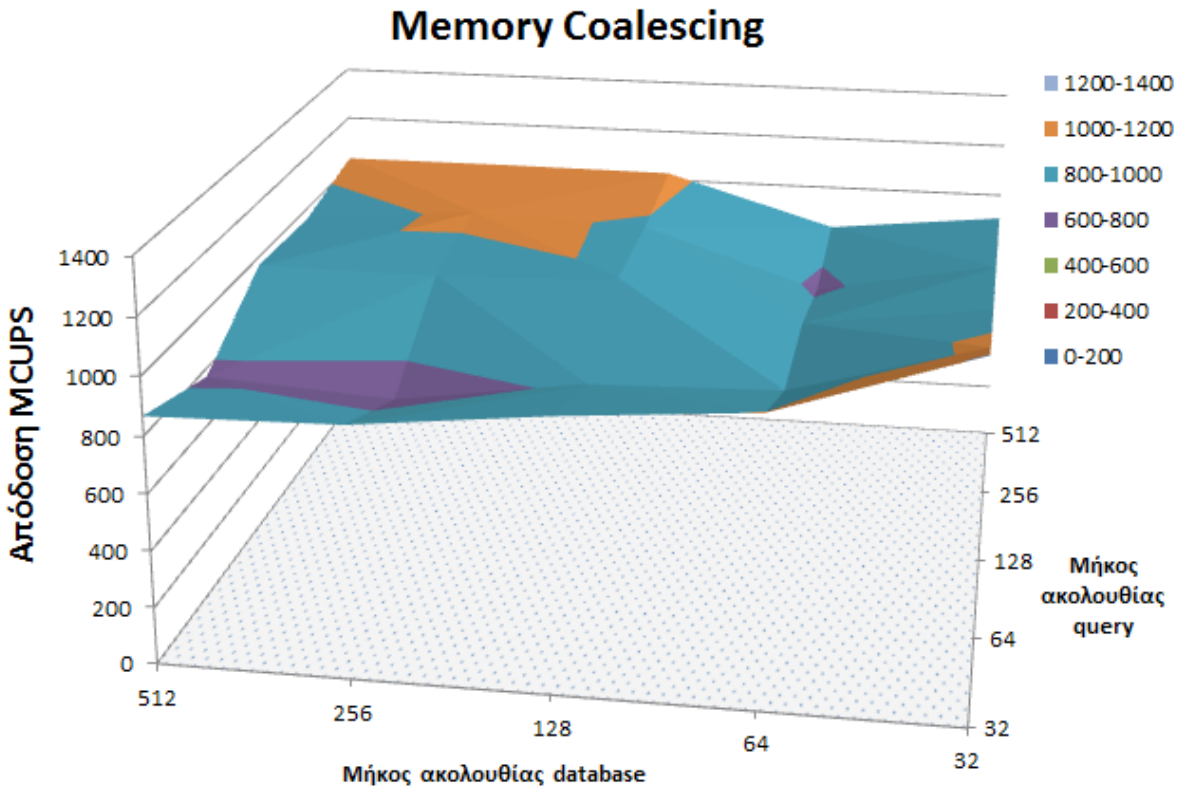
Επιτάχυνση υλοποίησης σε CPU

Όπως γίνεται αμέσως φανερό, η επιτάχυνση που παρουσίασε η υλοποίηση αυτή για την περίπτωση που οι πίνακες χωράνε στις caches (256*512) είναι πολύ καλή και πολύ κοντά στην γραμμική επιτάχυνση. Συγκεκριμένα όταν γίνεται χρήση μέχρι 4 πυρήνων, η επιτάχυνση είναι απόλυτα γραμμική, ενώ όταν χρησιμοποιούνται και οι 6 πυρήνες του ενός επεξεργαστή, η επιτάχυνση πέφτει ελαφρά στο 5.75x. Αυτό πιθανότατα συμβαίνει λόγο ανταγωνιστικότητας στις caches. Όταν οι πίνακες δεν χωράνε στις caches (512*512), η επιτάχυνση πέφτει ακόμα λίγο στο 5.63x.

Όταν χρησιμοποιούνται και οι 2 επεξεργαστές του συστήματος, η επιτάχυνση είναι στο 11.35x, αριθμός πολύ καλός αν ληφθεί υπ' όψην η NUMA (Non Uniform Memory Access) αρχιτεκτονική του συστήματος. Αντίστοιχα, όταν οι πίνακες δεν χωράνε στις caches (512*512) η επιτάχυνση διαμορφώνεται στο 11.00x. Για να επιτευχθεί αυτή η επίδοση έπρεπε να γίνει σωστό allocation στους πίνακες. Αυτό έγινε κάνοντας τα allocation εντός της παράλληλης περιοχής του κώδικα και με αυτόν τον τρόπο το κάθε core κάνει δεσμεύει χώρο στη τοπική του κύρια μνήμη.

5.3.2 Υλοποίηση σε GPU με συνένωση προσβάσεων στην global memory

Στο επόμενο τρισδιάστατο διάγραμμα φαίνεται η διακύμανση της απόδοσης καθώς τα μήκη των ακολουθιών αλλάζουν.

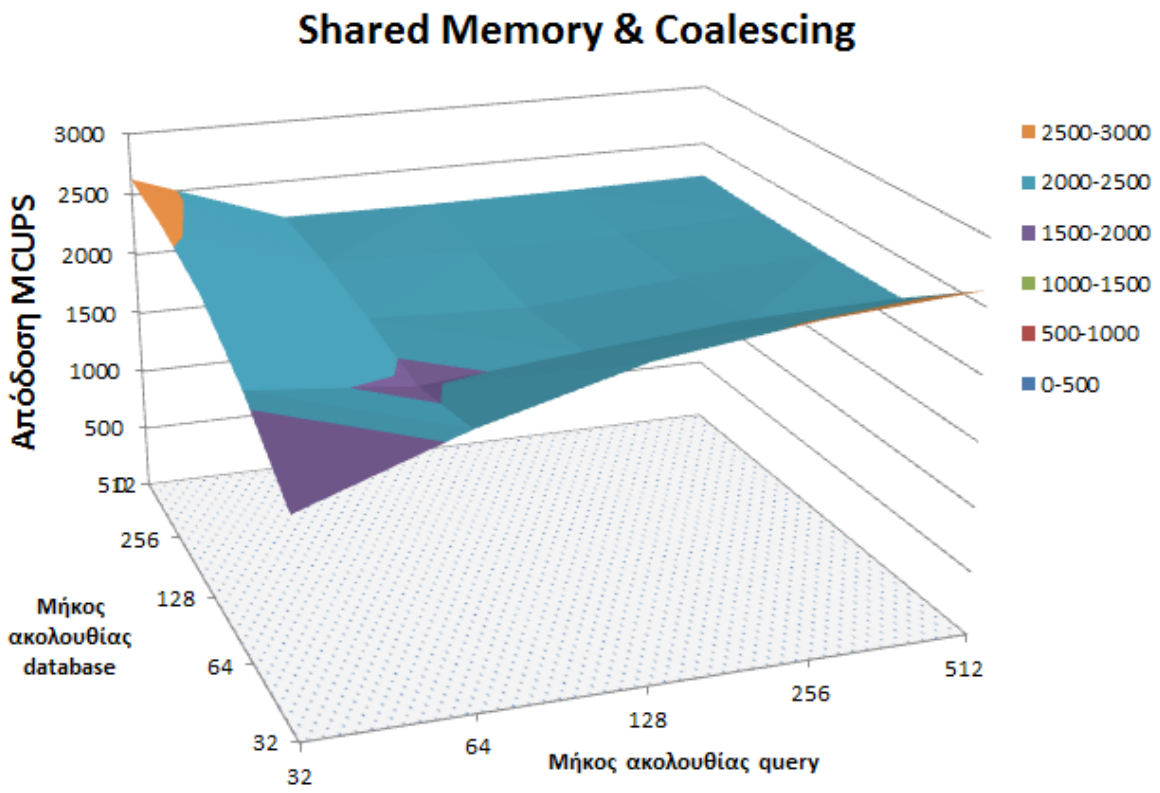


Διακύμανση επίδοσης της coalescing υλοποίησης συναρτήσει του μήκους των ακολουθιών

Από το διάγραμμα μπορεί να παρατηρηθεί ότι η απόδοση της υλοποίησης αυξάνεται όταν οι ακολουθίες έχουν μήκος μεγαλύτερο από 64. Σε αυτό το σημείο πρέπει να διευκρινιστεί ότι το ελάχιστο από τα μήκη των δύο ακολουθιών που στοιχίζονται, ορίζει και το μέγιστο μέγεθος της αντιδιαγωνίου, άρα τα στοιχεία που μπορούν να υπολογιστούν παράλληλα και σαν συνέπεια τον αριθμό των threads.. Όταν λοιπόν το ελάχιστο μήκος ακολουθίας ξεπερνάει τα δύο warps η απόδοση αρχίζει και αυξάνεται, καθώς το latency των προσβάσεων στην κύρια μνήμη κρύβεται. Η απόδοση της συγκεκριμένης υλοπίησης είναι μικρή όμως όχι αμελητέα, καθώς ξεπερνάει την απόδοση που επιτυγχάνει ο ένα επεξεργαστής.

5.3.3 Υλοποίηση σε GPU με συνένωση προσβάσεων στην κύρια μνήμη και χρήση της shared memory

Η διακύμανση της επίδοσης σε αυτήν την περίπτωση διαφοροποιείται αρκετά, όπως φάινεται στο επόμενο διάγραμμα.

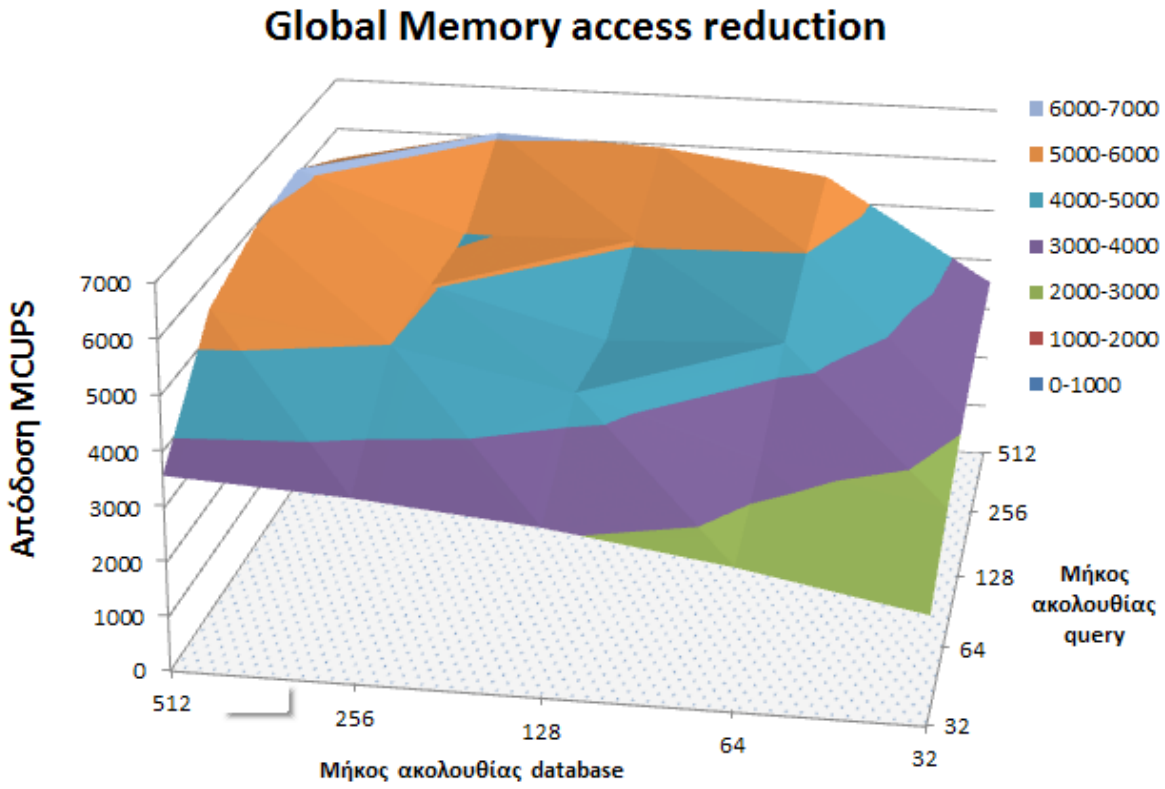


Διακύμανση επίδοσης της coalescing & shared memory υλοποίησης συναρτήσει του μήκους των ακολουθιών

Αντίθετα με πριν, όταν τα μήκη των ακολουθιών που συγκρίνονται είναι μεγαλύτερα από 64 (άρα και τα threads περισσότερα από 64) η επίδοση της συγκεκριμένης υλοποίησης μένει σταθερή κοντά στα 2200 MCUPS. Αυτό δεν είναι λογικό, καθώς συνήθως όσα περισσότερα warps υπάρχουν διαθέσιμα για εκτέλεση, τόσο πιο αποδοτικά τα χρησιμοποιεί ο hardware scheduler για να κρύψει τις καθυστερήσεις. Μπορούμε λοιπόν να συμπεράνουμε ότι τροχοπέδη στην επίδοση δεν αποτελεί το πλήθος των warps, αλλά το memory bandwidth.

5.3.4 Υλοποίηση σε GPU μειώνοντας τις εγγραφές στην global memory

Το συμπέρασμα της μέχρι τώρα ανάλυσης είναι ότι βασικό πρόβλημα του αλγόριθμου Smith-Waterman είναι το εύρος της μνήμης. Για το λόγο αυτό, σύμφωνα με την λογική που εξηγήθηκε στο προηγούμενο κεφάλαιο, δεν επιστρέφουμε τις τιμές των βοηθητικών πινάκων E και F στην global memory, υποτριπλασιάζοντας τις εγγραφές σε αυτήν.



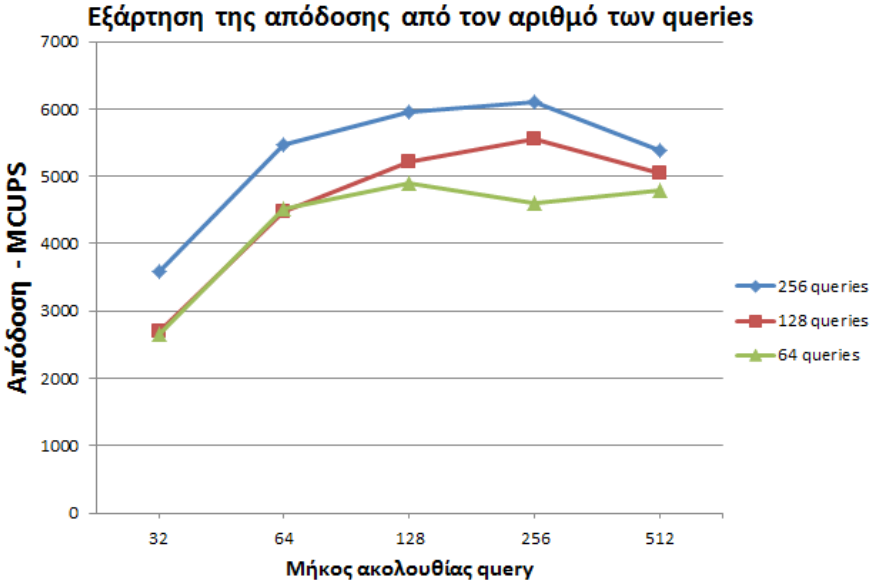
Διακύμανση επίδοσης της τελικής υλοποίησης συναρτήσει του μήκους των ακολουθιών

Στην τελική βελτιστοποίηση, η διακύμανση της επίδοσης παρουσιάζει πολύ φυσιολογική εικόνα, αφού όσο μεγαλώνει το μέγεθος των ακολουθιών, αυξάνεται και η επίδοση της υλοποίησης. Φαίνεται να έχει ξεπεραστεί σε μεγάλο βαθμό το πρόβλημα του memory bandwidth και για αυτό το λόγο παρατηρείται αύξηση επίδοσης τέτοιου βαθμού. Συγκεκριμένα, κάποιες φορές η απόδοση ξεπερνάει τα 6000 MCUPS.

Αξίζει να αναφερθεί ότι παρατηρείται μία μικρή πτώση επίδοσης όταν τα μήκη των ακολουθιών που συγκρίνονται είναι ίδια. Σε αυτή την περίπτωση ο πίνακας βαθμολογίας είναι τετράγωνος, κάτι που έχει ελαφριά επίπτωση στην απόδοση, αφού δεν μπορεί να

εκμεταλλευτεί καλά τον αριθμό των threads. Αυτό συμβαίνει γιατί ο αριθμός των threads που δημιουργούνται είναι όσο το μήκος της μεγαλύτερης αντιδιαγωνίου. Σε τετραγωνικό πίνακα όμως, το μήκος αυτής της αντιδιαγωνίου εμφανίζεται μόνο μία φορά, και έτσι δεν υπάρχει αποδοτική εκμετάλλευση των threads. Αντίθετα, στους πίνακες που ο αριθμός γραμμών και στηλών διαφέρει αρκετά, υπάρχουν αρκετές αντιδιαγώνιοι με μέγιστο μήκος, άρα το σύνολο των threads είναι ενεργό για μεγαλύτερο χρονικό διάστημα.

Η αύξηση του παραλληλισμού εσωτερικά των SMs, στο επίπεδο των SPs, δίνει αρκετή ώθηση στην επίδοση, όπως φαίνεται από το προηγούμενο διάγραμμα. Στη συνέχεια εξετάζουμε τι επίπτωση έχει στην επίδοση η αύξηση του παραλληλισμού στο επίπεδο το SMs, δηλαδή μέσω της ύπαρξης περισσότερων thread blocks. Για να γίνει αυτό, θα πρέπει να διαφοροποιηθεί το μέγεθος του query, δηλαδή το πλήθος των ακολουθιών που δίνονται προς στοίχιση.



Εξάρτηση της απόδοσης του βελτιστοποιημένου πυρήνα από τον αριθμό των queries

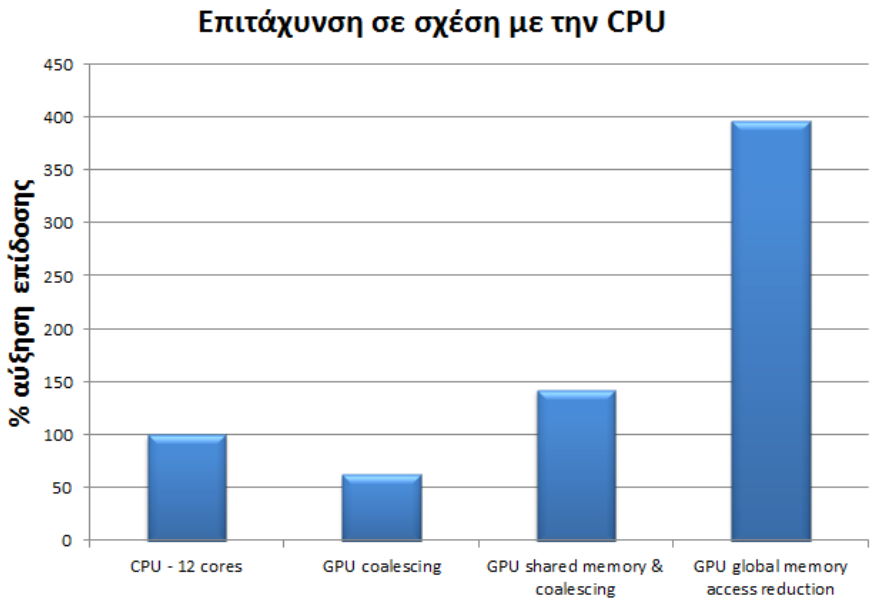
Όπως είναι λογικό, όσο μεγαλύτερη παραλληλοποίηση είναι δυνατή, τόσο καλύτερο scaling κάνει ο αλγόριθμος προσφέροντας υψηλότερες επιδόσεις.

ΚΕΦΑΛΑΙΟ 6

ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΜΕΛΛΟΝΤΙΚΕΣ ΚΑΤΕΥΘΥΝΣΕΙΣ

6.1 Συμπεράσματα

Στην παρούσα διπλωματική περιγράψαμε την αναγκαιότητα της στοίχισης ακολουθιών και του αλγορίθμου Smith – Waterman ως βασικό εργαλείο στοίχισης για την επιστήμη της Βιοπληροφορικής. Εξηγήσαμε αναλυτικά την λειτουργία του και βρήκαμε τα κατάλληλα σημεία παραλληλοποίησής του. Ακόμη, αναλύσαμε σε βάθος την αρχιτεκτονική των επεξεργαστών γραφικών, που μπορούν να παίξουν το ρόλο πανίσχυρων συνεπεξεραστών, κάνοντας τις κάποτε φιλοδοξίες του χώρου του GPGPU πραγματικότητα. Τέλος εφαρμόσαμε τις λειτουργίες του αλγορίθμου Smith – Waterman πάνω στα ιδιαίτερη αρχιτεκτονική των επεξεργαστών γραφικών, και αναζητήσαμε τρόπους για να επιτύχουμε υψηλή επίδοση. Τα αποτελέσματα των υλοποιήσεων που παρουσιάσαμε, μπορούν να συνοψιστούν στο επόμενο διάγραμμα.



Διάγραμμα επιτάχυνσης των GPU υλοποιήσεων σε σχέση με την CPU

Η βελτιστοποιημένη υλοποίηση του αλγορίθμου Smith – Waterman επεξεργαστές γραφικών Nvidia, παρουσιάζει 4 φορές μεγαλύτερη επίδοση όταν εκτελείται σε μια κάρτα Nvidia Tesla M2050 σε σχέση με δύο εξαπύρηνους Westmere Xeon επεξεργαστές.

Ο αλγόριθμος Smith – Waterman αποτελεί ένα ακόμα παράδειγμα συνήθους επιστημονικού υπολογισμού που προσφέρει υψηλή επιτάχυνση όταν χρησιμοποιηθούν επεξεργαστές γραφικών για τον υπολογισμό του. Τα ολοένα και περισσότερα θετικά αποτελέσματα του χώρου του GPGPU έχουν καταστήσει αναγκαία την ευρεία αποδοχή του, κάτι που γίνεται πραγματικότητα με το πέρασμα του χρόνου. Σε αυτό συμβάλουν και οι προγραμματιστές του χώρου που ολοένα και πληθαίνουν, και μαζί με αυτούς πληθαίνουν και οι ιδέες για αποδοτικότερη χρήση των επεξεργαστών γραφικών για υπολογισμούς γενικού σκοπού. Όπως επίσης συμβάλουν και οι εταιρίες σχεδιασμού επεξεργαστών γραφικών, αφού τόσο η Nvidia όσο και η ATI προσπαθούν συνεχώς να καταστήσουν πιο εύκολο τον προγραμματισμό των προϊόντων τους αλλά και να αυξήσουν την επίδοσή τους, προσθέτοντας χαρακτηριστικά σχεδιασμένα εξ ολοκλήρου για υπολογισμούς γενικού σκοπού. Το τελευταίο γίνεται εύκολα αντιληπτό από γενιά σε γενιά επεξεργαστών γραφικών.

6.2 Μελλοντικές κατευθύνσεις

Όπως έγινε κατανοητό από την ανάλυση των πειραματικών μετρήσεων, με τις προτεινόμενες βελτιστοποιήσεις το πρόβλημα του bottleneck που δημιουργεί το εύρος μνήμης στον αλγόριθμο Smith – Waterman αντιμετωπίστηκε σε ένα αρκετά μεγάλο βαθμό. Αυτό βέβαια δεν σημαίνει ότι η αναζήτηση της απόδοσης πρέπει να σταματήσει εδώ. Αρκετές τεχνικές και ιδέες θα μπορούσαν να αναλυθούν στο χαρτί και να εφαρμοστούν στην πράξη για να διευκρινιστεί η συμβολή τους στο πρόβλημα της στοίχισης ακολουθιών.

Οι τρεις πίνακες που χρησιμοποιεί η έκδοση που εφαρμόζει τον δυναμικό προγραμματισμό στον αλγόριθμο, θα μπορούσαν να αποθηκευτούν με διαφορετικό τρόπο, που να αυξάνει ακόμα περισσότερο το locality των τιμών. Ίσως ένας συνδιασμός των τριών πινάκων σε έναν μεγάλο, να προσέφερε ακόμα μεγαλύτερο πρακτικό εύρος μνήμης, με την εφαρμογή ολικής συνένωσης προσβάσεων στη μνήμη.

Επίσης, η επίδοση των διαφορετικών διαθέσιμων μνημών που προσφέρει ένας επεξεργαστής γραφικών θα πρέπει να ελέγχεται κατά περίπτωση. Στην Nvidia Tesla M2050 που είχαμε στην διάθεσή μας, ελέγχθηκε για παράδειγμα η χρήση της constant memory για την αποθήκευση του πίνακα αντικατάστασης, όμως προσέφερε χαμηλότερη επίδοση από την αποθήκευση στην global memory. Στις παλιότερες γενιές τέτοιου είδους τεχνικές είχαν θετικό αντίκτυπο στην επίδοση. Επειδή όμως η αρχιτεκτονική αλλάζει αρκετά από γενιά σε γενιά, λόγω του ότι το GPGPU βρίσκεται στην αρχή του ακόμα, δοκιμές πρέπει να γίνονται πάντα για να εξακριβώνεται ο καλύτερος συνδυασμός των διαθέσιμων τεχνικών χαρακτηριστικών.

Τέλος και οι χαρακτήρες των ακολουθιών αλλά και τα σκορ των πινάκων είναι μικρές τιμές και δεν χρειάζονται πολλά bits για την απεικόνισή τους, θα μπορούσε να εξελιχθεί τρόπος compression, δηλαδή να αποθηκεύονται περισσότερες από μία τιμές σε μία μεταβλητή αλλά και να επεξεργάζονται περισσότερες από μια τιμές. Με σωστό σχεδιασμό και μαθηματικές αποδείξεις, κάτι τέτοιο είναι πολύ πιθανό να μπορεί να εφαρμοστεί, και σε αυτή την περίπτωση θα υπήρχαν μεγάλα οφέλη τόσο στην επίδοση, όσο και στον χώρο αποθήκευσης.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] T. F .Smith, M. S. Waterman, Identification of Common Molecular Subsequences, J . Mol. Biol. (1981)

[2] Ali Khajeh-Saeed, Stephen Poole, J. Blair Perot , Acceleration of the Smith–Waterman algorithm using single and multiple graphics processors, Journal of Computational Physics 229

[3] Michael Farrar , Striped Smith–Waterman speeds database searches six times over other SIMD implementations , Bioinformatics Vol. 23 no. 2 2007

[4] Ashwin M. Aji, Wu-chun Feng, Filip Blagojevic, Dimitrios S. Nikolopoulos , Cell-SWat: Modeling and Scheduling Wavefront Computations on the Cell Broadband Engine

[5] Svetlin A Manavski, Giorgio Valle , CUDA compatible GPU cards as efficient hardware accelerators for Smith-Waterman sequence alignment , Italian Society of Bioinformatics Annual Meeting 2007

[6] Yuma Munekawa, Fumihiko Ino, Kenichi Hagihara , Design and Implementation of the Smith-Waterman Algorithm on the CUDA-Compatible GPU

[7] Yang Liu, Wayne Huang, John Johnson, Sheila Vaidya, GPU Accelerated Smith-

Waterman ICCS6

[8] Affan Zidan, Talal Bonny, Khaled N. Salama , High Performance Technique for Database Applications Using a Hybrid GPU/CPU Platform

[9] Łukasz Ligowski, Witold Rudnicki , An efficient implementation of Smith Waterman algorithm on GPU using CUDA, for massively parallel scanning of sequence databases, IPDPS 09

[10] Nvidia OpenCL Programming Guide

[11] Nvidia OpenCL Best Practices Guide

[12] Nvidia Fermi Compute Architecture Whitepaper

[13] Nvidia Fermi Tuning Guide

[14] http://en.wikipedia.org/wiki/Smith_waterman

[15] http://en.wikipedia.org/wiki/Sequence_alignment

[16] http://en.wikipedia.org/wiki/Gap_penalty

[17] http://en.wikipedia.org/wiki/Substitution_matrix

[18] http://en.wikipedia.org/wiki/BLOSUM

[19] http://www.anandtech.com/show/2849

[20] http://www.anandtech.com/show/2977/