ΤΕΙ ΑΘΗΝΑΣ ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πτυχιακη Εργασια

Προγραμματισμός με τεχνικές GPGPU

Φοιτητής:
Γεωργιος Σοφιανος
A.M: 031053

Επόπτης:
κ. Γεωργιος Μπαρδης

12 Οκτωβρίου 2014

Περιεχόμενα

Εισαγωγή

> "Anyone can build a fast CPU. The trick is to build a fast system."
>
> Seymour Cray

Τα αρχικά GPGPU πηγάζουν απο την φράση General Purpose computation on Graphics Processing Units, ή αλλιώς γνωστή ώς GPU Computing, δηλαδή υπολογισμός γενικού σκοπού σε μονάδες επεξεργασίας γραφικών.

Οι GPUs, είναι επεξεργαστές υψηλών επιδόσεων με δυνατότητα πολύ υψηλού υπολογισμού και διεκπεραιωτικότητας δεδομένων. Σχεδιασμένες αρχικά για γραφικά υπολογιστών με αρκετές δυσκολίες στον προγραμματισμό τους, οι σημερινές μονάδες επεξεργασίας γραφικών είναι παράλληλοι επεξεργαστές γενικής χρήσης με υποστήριξη για προσβάσιμες προγραμματιστικές διεπαφές και βιομηχανικά πρότυπα γλωσσών όπως η C.

Οι προγραμματιστές που μεταφέρουν τις εφαρμογές τους σε GPUs συνήθως πετυχαίνουν ταχύτητες πολλαπλάσιες από ότι μια αντίστοιχη εφαρμογή ειδικά βελτιστοποιημένη για κεντρική μονάδα επεξεργασίας (CPU). Ο όρος GPGPU δημιουργήθηκε από τον Mark Harris το 2002 όταν συνειδητοποίησε ότι αναπτυσσόταν μια τάση για χρήση των μονάδων επεξεργασίας γραφικών για εφαρμογές που δεν είχαν σχέση με γραφικά.

Από το 2012, οι GPU έχουν αναπτυχθεί σε συστήματα πολυπύρηνων επεξεργαστών παράλληλου υπολογισμού δίνοντας μας την δυνατότητα για πολύ αποδοτικό χειρισμό μεγάλου όγκου δεδομένων. Αυτός ο σχεδιασμός είναι πιο αποδοτικός από ότι οι κεντρικές μονάδες επεξεργασιάς (CPU) για αλγόριθμους όπου η επεξεργασία μεγάλου όγκου δεδομένων γίνεται παράλληλα, όπως σε αλγορίθμους sort μεγάλων λιστών, μετασχηματισμό κυμάτων δυο διαστάσεων, προσομοίωση βιολογικών δυναμικών.

0.1 Παράλληλος υπολογισμός

Για 30 χρόνια, ένας από τους πιο σημαντικούς τρόπους για να βελτιώσουμε την απόδοση των υπολογιστικών συσκευών των καταναλωτών ήταν η αύξηση της ταχύτητας στην οποία λειτουργεί το ρολόι ενός επεξεργαστή. Ξεκινώντας από περίπου το 1MHZ το 1980, οι περισσότεροι σύγχρονοι επεξεργαστές έχουν ταχύτητες μεταξύ 1GHz και 4GHz, δηλαδή είναι περίπου 1000 φορές πιο γρήγοροι. Αν και δεν είναι ο μόνος τρόπος με τον οποίο

έχουν βελτιωθεί οι επεξεργαστές, αποτελεί συνήθως μια αξιόπιστη πηγή για αύξηση της απόδοσης.

Τα τελευταία χρόνια όμως, οι κατασκευαστές έχουν αναγκαστεί να ψάξουν για εναλλακτικούς τρόπους αύξησης της υπολογιστικής δύναμης. Εξ αιτίας διάφορων περιορισμών στην κατασκευή ενσωματωμένων κυκλωμάτων, δεν είναι πλέον εύκολο να αυξάνουμε την ταχύτητα του ρολογιού του επεξεργαστή σαν τρόπο αύξησης της απόδοσης στις υπάρχουσες αρχιτεκτονικές. Στην αναζήτηση για επιπλέον υπολογιστική δύναμη για τους προσωπικούς επεξεργαστές, οι ερευνητές χρησιμοποίησαν τεχνολογίες που ήταν ήδη γνωστές από τους υπερ-υπολογιστές, στους οποίους είναι σύνηθες φαινόμενο να αποτελούνται από δεκάδες ή εκατοντάδες επεξεργαστές, οι οποίοι εκτελούν παράλληλες διεργασίες. Έτσι το 2005, οι κύριοι κατασκευαστές επεξεργαστών άρχισαν να προσφέρουν επεξεργαστές με δύο πυρήνες αντί για έναν.

Τα επόμενα χρόνια, ακολούθησαν υλοποιήσεις με τρεις,τέσσερις,έξι, ακόμα και οκτώ πυρήνες. Έχει ξεκινήσει ήδη μια μεγάλη στροφή της βιομηχανίας υπολογιστών στον παράλληλο υπολογισμό. Με την κυκλοφορία των διπύρηνων μέχρι και 8 ή 16 πυρήνων επεξεργαστών για σταθμούς εργασίας, ο παράλληλος υπολογισμός δεν είναι πλέον υπόθεση που αφορά μόνο τους εξωτικούς υπερ-υπολογιστές. Επίσης οι φορητές συσκευές όπως κινητά τηλέφωνα και φορητές συσκευές μουσικής έχουν αρχίσει να ενσωματώνουν δυνατότητες παράλληλου υπολογισμού σε μια προσπάθεια να προσφέρουν δυνατότητες πολύ ανώτερες από τους προγόνους τους. Όλο και περισσότερο, οι προγραμματιστές λογισμικού πρέπει να εξοικειωθούν με πλατφόρμες και τεχνολογίες παράλληλου υπολογισμού ώστε να προμηθεύουν με πλούσιες εμπειρίες την βάση των χρηστών τους. Το μέλλον αποτελείται από πολύ-νηματικές εφαρμογές, και από φορητές συσκευές που μπορούν ταυτόχρονα να παίζουν μουσική, να εξερευνούν το διαδίκτυο, και να παρέχουν GPS υπηρεσίες.

0.2 GPU Computing

Εργαλεία που χρησιμοποιήθηκαν

1.1 Εισαγωγή

Για την εκπόνηση της πτυχιακής εργασίας, χρησιμοποιήθηκαν αρκετά εργαλεία για λόγους ευχρηστίας αλλά και για εκπαιδευτικούς σκοπούς, σε μια προσπάθεια να αποκτηθούν γνώσεις για τεχνολογίες που πιστεύω πως χρειάζονται σε έναν απόφοιτο πληροφορικής.

1.1.1 XeTeX

Το XeTeX είναι μια μηχανή τυπογραφίας τύπου TeX η οποία χρησιμοποιεί κωδικοποίηση Unicode και υποστηρίζει σύγχρονες τεχνολογίες γραμματοσειρών όπως οι Opentype, Graphite και Apple Advanced Typography. Έχει σχεδιαστεί από τον Jonathan Kew και διανέμεται κάτω από την ελεύθερη άδεια λογισμικού X11. Ενώ δημιουργήθηκε αποκλειστικά για το Mac OS X, πλέον είναι διαθέσιμο για όλες τις γνωστές πλατφόρμες. Υποστηρίζει κωδικοποίηση Unicode και τα αρχεία κειμένου είναι εξαρχής σε μορφή UTF-8.

Το XeTeX μπορεί να χρησιμοποιήσει τις γραμματοσειρές που είναι εγκατεστημένες στο σύστημα, και να κάνει χρήση των ανεπτυγμένων τυπογραφικών δυνατοτήτων τους. Υποστηρίζει και μικρο-τυπογραφία, δηλαδή μια σειρά από μεθόδους που βελτιώνουν την αισθητική του κειμένου και το καθιστούν πιο ευανάγνωστο. Οι μέθοδοι συμπεριλαμβάνουν την μείωση μεγάλων κενών μεταξύ των λέξεων(expansion), την επέκταση των γραμμών όταν τελειώνουν με κάποιο μικρό σύμβολο, όπως η τελεία η ένα στρογγυλό γράμμα όπως το "o" (protrusion), και ο χωρισμός γραμμών (hyphenation). Αν και το XeTeX είναι υποδεέστερο του LaTeX στην μικρο-τυπογραφία, επιλέχτηκε για αυτήν την πτυχιακή εργασία λόγω των πολλών άλλων πλεονεκτημάτων, όπως η χρήση unicode χαρακτήρων αλλά και η ευκολία διαχείρισης γραμματοσειρών. Το XeTeX

1.1.2 Texmaker

Το Texmaker είναι ένα επεξεργαστής κειμένου για LaTeX και XeLaTeX. Είναι cross-platform ανοιχτού κώδικα με ενσωματωμένο PDF Viewer. Το Texmaker είναι μια εξολοκλήρου εφαρμογή Qt. Ο επεξεργαστής κειμένου έχει άρτια υποστήριξη unicode, λειτουργία ελέγχου ορθογραφίας, αυτόματη συμπλήρωση κειμένου, κ.α. Επίσης υποστηρίζει regular expressions για την

εύρεση και αντικατάσταση κειμένου.

Το Texmaker περιέχει λειτουργίες για αυτόματη χρήση των παρακάτω λειτουργιών:

- Παραγωγή νέου εγγράφου, γράμματος, βιβλίου, κ.α
- Παραγωγή πινάκων, περιβάλλοντος σχημάτων, κ.α
- Εξαγωγή κειμένου σε HTML ή ODT διάταξης

Μερικές από τις ετικέτες LaTeX και μαθηματικά σύμβολα μπορούν να χρησιμοποιηθούν με ένα click. Ο Texmaker βρίσκει αυτόματα τα λάθη και προειδοποιήσεις και τα εμφανίζει στο αρχείο καταγραφής.

1.1.3 TortoiseGit

Το TortoiseGIT είναι μια εφαρμογή που υλοποιεί ένα σύστημα διαχείρισης εκδόσεων λογισμικού. Το σύστημα αυτό, είναι ένα υποσύνολο του ελέγχου αναθεωρήσεων. Ο έλεγχος αναθεωρήσεων χρησιμοποιείται για την διαχείριση των αλλαγών σε κείμενα, προγράμματα υπολογιστών, ιστοσελίδες, και άλλες συλλογές απο πληροφορίες. Οι αλλαγές συνήθως αναγνωρίζονται απο έναν αριθμό η απο ένα γράμμα, το οποίο ονομάζεται αριθμός αναθεώρησης.

Για παράδειγμα, αν τα αρχεία είχαν αρχικό αριθμό το "αναθεώρηση 1", μετά τις αλλαγές θα αποκτήσει τον αριθμό "αναθεώρηση 2", κτλ. Κάθε αναθεώρηση σχετίζεται με μια χρονοσήμανση, και με το άτομο το οποίο πραγματοποίησε την αλλαγή. Οι αναθεωρήσεις μπορούν να συγκριθούν, να αποκαθιστούν, και με κάποιους τύπους αρχείων, να συγχωνευτούν.

Η ανάγκη για ένα λογικό τρόπο οργάνωσης και ελέγχου αναθεωρήσεων υπάρχει σχεδόν απο τότε που εφευρέθηκε η γραφή, αλλά ο έλεγχος αναθεωρήσεων έγινε πιο σημαντικός και πολύπλοκος όταν ξεκίνησε η εποχή των υπολογιστών. Η αρίθμηση των εκδόσεων βιβλίων είναι ένα απο τα παραδείγματα της προηγούμενης εποχής. Σήμερα, τα πιο πολύπλοκα εργαλεία ελέγχου αναθεωρήσεων είναι αυτά που χρησιμοποιούνται στην δημιουργία λογισμικού, στα οποία μια ομάδα απο ανθρώπους μπορεί να αλλάξει τα ίδια αρχεία.

Μπορεί το σύστημα ελέγχου να είναι ένα ξεχωριστό πρόγραμμα, αλλα ο έλεγχος αναθεωρήσεων βρίσκεται ενσωματωμένος και μέσα σε διάφορους τύπους λογισμικού όπως οι επεξεργαστές κειμένου, αλλα και συστήματα διαχείρισης περιεχομένου π.χ Wikipedia, Wordpress, κ.α.

Ο έλεγχος αναθεωρήσεων δίνει την δυνατότητα να επαναφέρουμε ένα κείμενο σε μια προηγούμενη αναθεώρηση, το οποίο είναι πολύ σημαντικό για να υπάρχει έλεγχος στις αλλαγές, να διορθώνονται τα λάθη, και να προστατεύεται το περιεχόμενο από βανδαλισμό και ανεπιθύμητα μηνύματα.

1.1.4 Github

Το Github είναι μια διαδικτυακή υπηρεσία αποθηκευτικού χώρου Git, που επιτρέπει πλήρη κατανεμημένο ελέγχο αναθεωρήσεων και διαχείριση πηγαίου κώδικα μέσω Git, ενώ προσθέτει επιπλέον λειτουργίες σε αυτό. Το

Github προσφέρει ένα web-based γραφικό περιβάλλον, εφαρμογές σταθερών υπολογιστών, αλλά και εφαρμογές κινητών συσκευών. Επίσης παρέχει έλεγχο πρόσβασης και αρκετά χαρακτηριστικά συνεργασίας, όπως wiki, διαχείριση έργων, παρακολούθηση λαθών, και αιτήματα χαρακτηριστικών για κάθε σχέδιο.

Η ανάπτυξη του Github ξεκίνησε το 2007. Η γλώσσα προγραμματισμού που χρησιμοποιήθηκε είναι η Ruby on Rails και η Erlang. Η ιστοσελίδα ανακοινώθηκε τον Απρίλιο του 2008 απο τον Tom Preston-Werner, τον Chris Wanstrath, και τον Pj Hyett.

Το Github παρέχει δωρεάν λογαριασμούς, που συνηθίζεται να χρησιμοποιούνται για έργα λογισμικού ανοιχτού κώδικα. Ως το 2014, οι χρήστες του φτάνουν τα 3.4 εκατομμύρια, καθιστώντας το το μεγαλύτερο αποθηκευτικό χώρο πηγαίου κώδικα στον πλανήτη.

Σχήμα 1.1: Λογότυπο Github

Υλοποιήσεις

2.1 Ιστορία

Παραδοσιακά, η σειρά αγωγών των γραφικών, αποτελείται από τις καταστάσεις μετατροπή και φωτισμός, συναρμολόγηση αρχέγονων, μετατροπή σε pixels, και σκίαση. Οι πρώτες GPU είχαν όλες τις λειτουργίες που χρειάζονται για να εκτελεστεί η σειρά αγωγών, αλλά με τον καιρό όλο και περισσότερες καταστάσεις έγιναν δυνατό να προγραμματιστούν με την έλευση ειδικών επεξεργαστών, όπως επεξεργαστές κορυφών και τεμάχια επεξεργαστών, που κατέστησαν κάποιες λειτουργίες πιο ευέλικτες.

Όταν οι τιμές συνέχισαν να πέφτουν ενώ η υπολογιστική δύναμη αυξανόταν, η ερευνητική κοινότητα σκέφτηκε τρόπους να αξιοποιηθεί αυτή η δύναμη για τον υπολογισμό δύσκολων λειτουργιών. Όμως, καθώς η δυνατότητα των επεξεργαστών ήταν περιορισμένη και η διεπαφή προγραμματιστικής διεπαφής (API) των οδηγών γραφικών ήταν σχεδιασμένη για να υλοποιεί συγκεκριμένα την σειρά αγωγών, έπρεπε να ληφθούν υπόψιν πολλές παράμετροι.

Για παράδειγμα, όλα τα δεδομένα έπρεπε να κωδικοποιηθούν σε υφές ως πίνακες δυο διαστάσεων που αναπαριστούν pixel, με περιεχόμενο τιμές χρωμάτων και κάποιο κανάλι alpha για την διαφάνεια. Επιπλέον, οι υφές είναι αντικείμενα μόνο προσπελάσιμα, και δεν επαναγράφονταν, κάτι που ανάγκαζε τους προγραμματιστές να αποθηκεύουν κάθε φορά καινούρια υφή με τις αλλαγές. Τέλος, οι περισσότερες GPU υποστήριζαν μόνο λειτουργίες μονής κινητής υποδιαστολής, αναγκάζοντας τους προγραμματιστές να προσομοιώνουν λογικές λειτουργίες.

Αυτοί οι περιορισμοί, ήταν ο μεγαλύτερος λόγος που ώθησε τους κατασκευαστές GPU (AMD,NVIDIA,INTEL), να δημιουργήσουν προγραμματιστικές διεπαφές ειδικές για την κοινότητα του GPGPU και να εξελίξουν τις συσκευές τους για καλύτερη υποστήριξη.

Το πεδίο του προγραμματισμού γενικής χρήσης έχει αναπτυχθεί με ταχύτατους ρυθμούς τα τελευταία χρόνια, έτσι ώστε τώρα υπάρχουν αρκετές υλοποιήσεις για τον προγραμματισμό των μονάδων επεξεργασίας γραφικών. Πρόσφατα, έχουν γίνει προσπάθειες δημιουργίας προτύπων. Ο προγραμμα-

τισμός των GPUs αναπτύχθηκε όταν το CUDA και το Stream κατέφθασαν στο τέλος του 2006. Αυτές οι διεπαφές και οι γλώσσες, σχεδιάστηκαν από τις εταιρίες κατασκευής των GPUs σε πολύ κοντινή σχέση με το υλικό, το οποίο αποτέλεσε μεγάλο βήμα προς ένα πιο εύχρηστο, ταιριαστό και μελλοντικά-ασφαλές προγραμματιστικό μοντέλο.

Η ανοιχτή γλώσσα προγραμματισμού (OpenCL) δημιουργήθηκε για να παρέχει ένα γενικό API ετερογενή υπολογισμού σε διάφορες μορφές παράλληλων συσκευών, συμπεριλαμβανομένου μονάδων επεξεργασίας γραφικών, πολυπύρηνων κεντρικών μονάδων επεξεργασίας, κ.α

2.1.1 Μέλλον

Οι πρόσφατες δραστηριότητες των μεγάλων κατασκευαστών μας δείχνουν ότι τα μελλοντικά σχέδια των μικροεπεξεργαστών και μεγάλων HPC συστημάτων θα είναι υβριδικά/ετερογενούς φύσης. Αυτά τα συστήματα θα βασίζονται στην ενσωμάτωση δύο τύπων εξαρτημάτων:

- Τεχνολογία πολυπύρηνων CPU: ο αριθμός των πυρήνων θα συνεχίσει να αυξάνεται λόγω της επιθυμίας να ενσωματώσουμε περισσότερα εξαρτήματα σε ένα τσιπ.
- Ειδικού τύπου υλικό και μαζικά παράλληλους επιταχυντές: Για παράδειγμα, οι GPUs υπερτερούν των CPUs σε απόδοση κινητής υποδιαστολής, τα τελευταία χρόνια. Επίσης ο προγραμματισμός σε αυτές έχει γίνει εύκολος, αν όχι ευκολότερος, από ότι στις CPUs

Η σχετική ισορροπία στα μελλοντικά σχέδια δεν είναι ξεκάθαρη και μπορεί να αλλάξει με την πάροδο του χρόνου. Δεν υπάρχει καμία αμφιβολία ότι οι μελλοντικές γενιές των υπολογιστικών συστημάτων, από τους φορητούς υπολογιστές μέχρι και τους υπερ-υπολογιστές θα αποτελείται από μια σύσταση ετερογενών συστημάτων.

2.1.2 Προβλήματα

Τα προβλήματα και οι προκλήσεις για τους προγραμματιστές στο καινούριο περιβάλλον των υβριδικών συστημάτων, είναι υπαρκτά. Κρίσιμα τμήματα του λογισμικού ήδη δυσκολεύονται να προλάβουν τον ρυθμό των αλλαγών. Σε μερικές περιπτώσεις, η απόδοση δεν είναι ανάλογη του αριθμού των πυρήνων, γιατί ένα μεγάλο μέρος του χρόνου ξοδεύεται στην μετακίνηση των δεδομένων παρά στους υπολογισμούς. Σε άλλες περιπτώσεις, το βελτιστοποιημένο λογισμικό για το συγκεκριμένο υλικό, παραδίδεται χρόνια μετά από την παράδοση του υλικού, και έτσι είναι απαρχαιωμένο όταν παραδοθεί. Και σε άλλες περιπτώσεις, όπως σε μερικές πρόσφατες υλοποιήσεις GPU, το λογισμικό δεν εκτελείται καθόλου γιατί το προγραμματιστικό περιβάλλον έχει αλλάξει υπερβολικά.

2.2 CUDA

2.2.1 Εισαγωγή

Το CUDA είναι μια πλατφόρμα παράλληλου υπολογισμού, που δημιουργήθηκε από την NVIDIA και υλοποιήθηκε στις κάρτες γραφικών τις οποίες παράγει η ίδια. Το CUDA δίνει στους προγραμματιστές άμεση πρόσβαση στο σετ εικονικών εντολών και την μνήμη των στοιχείων του παράλληλου υπολογισμού σε κάρτες γραφικών NVIDIA.

Αξιοποιώντας το CUDA, οι κάρτες γραφικών(GPU) μπορούν να χρησιμοποιηθούν για υπολογισμό γενικής χρήσης (δηλαδή όχι αποκλειστικά για γραφικά).Οι GPU έχουν μια αρχιτεκτονική παράλληλης εξόδου η οποία δίνει έμφαση στην εκτέλεση πολλών threads με μικρή ταχύτητα, σε αντίθεση με τις CPU όπου εκτελείται ένα thread με μεγάλη ταχύτητα.

Σχήμα 2.1: Βιολογία και πληροφορική

Η πλατφόρμα CUDA είναι προσβάσιμη στους προγραμματιστές μέσω βιβλιοθηκών, εντολών μεταγλώττισης, και προεκτάσεων σε γλώσσες προγραμματισμού βιομηχανικής κλίμακας, όπως η C, C++ και Fortran.

Οι προγραμματιστές της C/C++, χρησιμοποιούν το CUDA C/C++, μεταγλωττισμένο με το nvcc, έναν LLVM βασισμένο μεταγλωττιστή, και οι προγραμματιστές της Fortran χρησιμοποιούν το CUDA Fortran, μεταγλωττισμένο με τον μεταγλωττιστή PGI CUDA Fortran απο το The Portland Group. Εκτώς απο τα παραπάνω, η πλατφόρμα CUDA υποστηρίζει και άλλες διεπαφές υπολογισμού, όπως το OpenCL του Khronos Group, το DirectCompute της Microsoft, και το C++ AMP.

Στην βιομηχανία των υπολογιστών, οι GPUs δεν χρησιμοποιούνται μόνο για τα γραφικά αλλά και στους υπολογισμούς φυσικής παιχνιδιών (π.χ καπνός, φωτιά, ροή υγρών). Γνωστά παραδείγματα αποτελούν οι μηχανές PhysX και η Bullet. Το CUDA επίσης χρησιμοποιείται για να επιταχύνει μη-γραφικές εφαρμογές στην βιοπληροφορική, στην κρυπτογραφία, και σε πολλά άλλα πεδία.

Γενικότερα, η υπολογιστική δύναμη της GPU, βασίζεται στην παράλληλη αρχιτεκτονική της. Για αυτό, η πλατφόρμα του CUDA παρουσιάζει το νήμα(thread) ως το μικρότερο στοιχείο παραλληλισμού. Όμως, σε σύγκριση με την κεντρική μονάδα επεξεργασίας, τα νήματα της GPU έχουν μικρότερο

κόστος χρήσης πόρων και μικρότερο κόστος δημιουργίας και αντικατάστασης.

Σημειώνεται ότι οι GPU είναι αποτελεσματικές, μόνο όταν τρέχει μεγάλος αριθμός απο τέτοια νήματα. Μια ομάδα από νήματα, που εκτελούνται παράλληλα, επικοινωνούν και συγχρονίζονται μεταξύ τους ονομάζεται block. Ο μέγιστος αριθμός των νημάτων σε ενα block είναι ένας περιορισμός που υπάρχει στην κάθε μονάδα γραφικής επεξεργασίας. Τέλος, μια ομάδα από blocks τα οποία έχουν την ίδια διάσταση και εκτελούνται απο το ίδιο πρόγραμμα CUDA παράλληλα, ονομάζεται πλέγμα.

Για να επιτρέψει βέλτιστη επίδοση για διαφορετικά πρότυπα, το CUDA εκτελεί ένα ιεραρχικό μοντέλο μνήμης, αντίθετα με τα παραδοσιακά μοντέλα που συναντάμε συνήθως στους υπολογιστές. Ο υπολογιστής και η συσκευή, έχουν τις δικές τους περιοχές μνήμης, τις οποίες ονομάζουν host memory και device memory, αντίστοιχα. Το CUDA παρέχει βελτιστοποιημένες λειτουργίες για να μεταφέρει δεδομένα από και προς αυτούς τους ξεχωριστούς χώρους. Κάθε νήμα κατέχει το δικό του αρχείο καταχώρησης, το οποίο μπορεί να προσπελαστεί και να εγγραφεί.

Επιπλέον, μπορεί να προσπελάσει το δικό του αντίγραφο της τοπικής μνήμης. Όλα τα νήματα στο ίδιο πλέγμα μπορούν να προσπελάσουν και να γράψουν στην περιοχή της κοινόχρηστης μνήμης (shared memory). Για να αποφευχθούν κίνδυνοι από ταυτόχρονη προσπέλαση, μηχανισμοί συγχρονισμού νημάτων πρέπει να χρησιμοποιηθούν. Η κοινόχρηστη μνήμη, είναι οργανωμένη σε ομάδες που ονομάζονται τράπεζες, οι οποίες μπορούν να προσπελαστούν παράλληλα. Όλα τα νήματα έχουν επίσης πρόσβαση στον χώρο μνήμης που ονομάζεται καθολική μνήμη (global memory) και στις περιοχές που ονομάζονται μνήμη σταθερών (constant memory) και μνήμη υφής (texture memory).

2.2.2 Πλεονεκτήματα

Το CUDA έχει τα εξής πλεονεκτήματα σε σχέση με τους παραδοσιακούς τρόπους υπολογισμού γενικής χρήσης που εκτελούνται μέσω προγραμματιστικών διεπαφών γραφικών:

- Διασκορπισμένες προσπελάσεις - ο κώδικας μπορεί να διαβαστεί από αυθαίρετες διευθύνσεις στην μνήμη.
- Ενοποιημένη εικονική μνήμη (CUDA 6)
- Κοινόχρηστη μνήμη - το CUDA εκθέτει μια γρήγορη περιοχή κοινόχρηστης μνήμης (μέχρι 48KB για κάθε επεξεργαστή) η οποία μπορεί να μοιραστεί ανάμεσα στα threads. Αυτή μπορεί να χρησιμοποιηθεί σαν κρυφή μνήμη διαχειρίσιμη απο τον χρήστη, επιτρέποντας μεγαλύτερο εύρος δεδομένων απο ότι είναι δυνατό με τις προσπελάσεις υφών.
- Πιο γρήγορες μεταφορτώσεις και προσπελάσεις από και προς την GPU
- Πλήρης υποστήριξη για ακέραιες και bitwise λειτουργίες, για παράδειγμα τις προσπελάσεις υφών.

2.2.3 Περιορισμοί

Το CUDA δεν υποστηρίζει ολόκληρο το πρότυπο της γλώσσας C, καθώς τρέχει μέσω ενός μεταγλωττιστή C++, ο οποίος εμποδίζει συγκεκριμένα μέρη

Feature support (unlisted features are supported for all compute capabilities)	Compute capability (version)

της γλώσσας C να μεταγλωττιστούν.

2.3 OpenCL

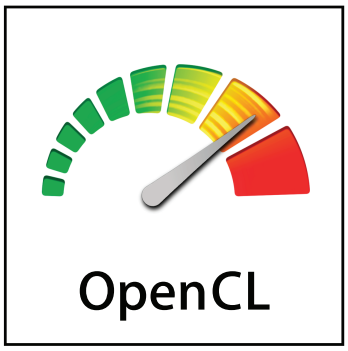
2.3.1 Εισαγωγή

Το πρώτο GPGPU framework δημιουργήθηκε απο την NVIDIA και ήταν το CUDA. Το CUDA παρείχε στους χρήστες ένα προγραμματιστικό περιβάλλον σε C like γλώσσα για την GPU. Όμως ήταν κλειστού κώδικα και μπορεί να τρέχει μόνο σε NVIDIA κάρτες γραφικών. Λόγω της μεγάλης δημοτικότητας του CUDA, η ανάγκη για ένα ανοιχτό πρότυπο αρχιτεκτονικής που θα υποστηρίζει διάφορα είδη συσκευών απο διάφορους κατασκευαστές γινόταν όλο και πιο σημαντική. Έτσι τον Ιούνιο του 2008 το Khronos Group δημιούργησε το OpenCL 1.0. Αρκετοί κατασκευαστές σταδιακά παρείχαν εργαλεία για προγραμματισμό σε OpenCL συμπεριλαμβανομένων των: Nvidia OpenCL Drivers and Tools, AMD APP SDK,Intel SDK for OpenCL applications,IBM Server with OpenCL development Kit, κ.α. Σήμερα το OpenCL επιτρέπει πολυπύρηνο προγραμματισμό, προγραμματισμό GPU, κ.α.[**opencl-1**]

Το OpenCL είναι ένα πλαίσιο για κατασκευή εφαρμογών που εκτελούνται σε ετερογενή συστήματα που αποτελούνται από κεντρικές μονάδες επεξεργασίας(CPU), μονάδες επεξεργασίας γραφικών(GPU), επεξεργαστές ψηφιακών σημάτων(DPS), συστοιχίες προγραμματιζόμενων θυρίδων(FPGA), και άλλους επεξεργαστές. Το OpenCL περιέχει μια γλώσσα, υποσύνολο του ISO C99 με επεκτάσεις, για τον προγραμματισμό αυτών των συσκευών, προγραμματιστικές διεπαφές εφαρμογών(API) για τον έλεγχο της πλατφόρμας και την εκτέλεση προγραμμάτων στις υπολογιστικές συσκευές. Το OpenCL παρέχει παράλληλο υπολογισμό χρησιμοποιώντας παραλληλισμό διεργασιών και δεδομένων.
Αποτελεί το πρώτο ανοιχτό, ελεύθερο από τέλη αδειών πρότυπο για cross-platform, παράλληλο προγραμματισμό μοντέρνων επεξεργαστών, που χρησιμοποιούνται συνήθως σε προσωπικούς υπολογιστές, διακομιστές, και φορητές/ενσωματωμένες συσκευές. Το OpenCL (Open Computing Language) βελτιώνει αισθητά την ταχύτητα και την απόκριση μεγάλου εύρους εφαρμογών σε διάφορες κατηγορίες αγορών από παιχνίδια και ψυχαγωγία, μέχρι επιστημονικές εφαρμογές και εφαρμογές υγείας. Συντηρείται απο το μή-κερδοσκοπικό τεχνολογικό συνεταιρισμό Khronos Group. Έχει υιοθετηθεί από πολλές μεγάλες εταιρίες όπως η Apple, Intel, Qualcomm, AMD, Nvidia,Samsung,ARM Holdings.

2.3.2 Επισκόπηση

Το OpenCL ορίζει μια διεπαφή προγραμματισμού εφαρμογών με την οποία επιτρέπει στα προγράμματα που τρέχουν στον οικοδεσπότη να εκτελέσουν πυρήνες στην συσκευή υπολογισμού, και να διαχειριστούν την μνήμη



Σχήμα 2.2: Λογότυπο OpenCL

της συσκευής, που είναι ξεχωριστή από την μνήμη του οικοδεσπότη. Τα προγράμματα του OpenCL είναι σχεδιασμένα ώστε να μεταγλωττίζονται την ώρα της εκτέλεσης, και με αυτόν τον τρόπο γίνεται δυνατό να εκτελεστούν σε διάφορες συσκευές. Το πρότυπο του OpenCL ορίζει διεπαφές προγραμματισμού για γλώσσα C και C++. Διεπαφές επίσης υπάρχουν και για άλλες γλώσσες, όπως Python, Julia, και Java. Μια εφαρμογή του OpenCL προτύπου αποτελείται απο μια βιβλιοθήκη που υλοποιεί την διεπαφή για C και C++, και έναν μεταγλωττιστή OpenCL για τις συσκευές υπολογισμού.

Ιεραρχία μνήμης

Το OpenCL ορίζει ιεραρχία τεσσάρων επιπέδων για την μνήμη των συσκευών υπολογισμού:

- Καθολική μνήμη: διαμοιράζεται σε όλες τις συσκευές υπολογισμού, αλλά έχει μεγάλη καθυστέρηση απόκρισης
- Μνήμη προσπέλασης: μικρότερη, χαμηλή καθυστέρηση απόκρισης, εγγράψιμη από την κεντρική μονάδα επεξεργασίας του οικοδεσπότη, αλλά όχι των συσκευών υπολογισμού.
- Τοπική μνήμη: διαμοιράζεται σε πολλά στοιχεία υπολογισμού μιας συσκευής
- Ιδιωτική μνήμη στοιχείου (καταχωρητές)

Δεν είναι απαραίτητο για όλες τις συσκευές να υλοποιήσουν την ιεραρχία της μνήμης στο υλικό. Η συνέπεια στα διάφορα επίπεδα της ιεραρχίας είναι χαλαρή, και επιβάλλεται μόνο από κατηγορηματικά στοιχεία συγχρονισμού, όπως τα εμπόδια.

2.3.3 Ιστορία

Το OpenCL δημιουργήθηκε αρχικά απο την Apple Inc., η οποία κατέχει τα πνευματικά δικαιώματα, και εξευγενίστηκε σε αρχική πρόταση σε συνεργασία με τεχνικές ομάδες της AMD,IBM,Qualcomm,Intel, και Nvidia. Η Apple καταχώρησε την πρόταση στο Khronos Group, και τον Ιούνιο του 2008 διαμορφώθηκε το Khronos Compute Working Group με αντιπρόσωπους από εταιρίες επεξεργαστών, μονάδων υλικού γραφικών, ενσωματωμένων-επεξεργαστών, και λογισμικού. Αυτό το group εργάστηκε για 5 μήνες ώστε

να φέρει σε πέρας τον πρώτο προσδιορισμό για το OpenCL 1.0, ο οποίος κυκλοφόρησε τον Δεκέμβριο του 2008.

OpenCL 1.0

Η AMD, αν και αρχικά εργαζόταν πάνω στο πρότυπο Close to Metal, αποφάσισε να στραφεί και να υποστηρίξει το OpenCL. Η Nvidia ανακοίνωσε πλήρης υποστήριξη στην εργαλειοθήκη υπολογισμού GPU. Το 2009, η IBM κυκλοφόρησε την πρώτη έκδοση του μεταγλωττιστή της με υποστήριξη για OpenCL

OpenCL 1.1

Το OpenCL 1.1 επικυρώθηκε από το Khronos Group τον Ιούνιο του 2010, και προσθέτει σημαντικές λειτουργίες για βελτιωμένη ευελιξία παράλληλου προγραμματισμού, λειτουργικότητα, και επιδόσεις.

OpenCL 1.2

Το OpenCL 1.2 ανακοινώθηκε τον Νοέμβριο του 2011 απο το Khronos Group, το οποίο προσθέτει αρκετές λειτουργίες σε σχέση με τις προηγούμενες εκδόσεις όσον αφορά τις επιδόσεις και χαρακτηριστικά για παράλληλο προγραμματισμό.

OpenCL 2.0

Το OpenCL 2.0 επικυρώθηκε και κυκλοφόρησε τον Νοέμβριο του 2013 απο το Khronos Group και αποτελεί την τελευταία έκδοση του OpenCL.

2.3.4 Στόχοι

Ο στόχος του OpenCL είναι να κάνει ορισμένους τύπους παράλληλου προγραμματισμού πιο εύκολους, και να παρέχει ανεξαρτήτου κατασκευαστή παράλληλη εκτέλεση κώδικα μέσω επιτάχυνσης υλικού. Το OpenCL είναι το πρώτο ανοιχτό, ελεύθερο πρότυπο για παράλληλο προγραμματισμό γενικού σκοπού ετερογενών συστημάτων. Παρέχει ένα προγραμματιστικό περιβάλλον που βοηθάει τους προγραμματιστές να γράψουν αποδοτικό, φορητό κώδικα για συστήματα υψηλής απόδοσης, προσωπικούς υπολογιστές, και κινητές συσκευές χρησιμοποιώντας ένα μείγμα πολυπύρηνων CPUs,GPUs, και DSPs.
Το OpenCL παρέχει στους προγραμματιστές ένα κοινό σετ εργαλείων εύκολης χρήσης, ώστε αυτοί να εκμεταλλευτούν οποιαδήποτε συσκευή που περιέχει οδηγό OpenCL για την εκτέλεση παράλληλου κώδικα. Το OpenCL framework ορίζει μια γλώσσα C like για την δημιουργία των πυρήνων, και ένα σετ απο APIs για την δημιουργία και την διαχείριση αυτών των πυρήνων. Οι πυρήνες είναι διαδικασίες οι οποίες μπορούν να εκτελούνται σε διαφορετικές συσκευές. Οι πυρήνες μεταγλωττίζονται απο έναν μεταγλωτιστή runtime, μέσω κατάλληλου προγράμματος. Αυτό επιτρέπει στα προγράμματα να εκμεταλλεύονται όλες τις συσκευές ενός συστήματος με ένα σετ φορητών υπολογιστικών πυρήνων.

2.3.5 Μοντέλο εκτέλεσης

Τα κύρια μέρη εκτέλεσης ενός προγράμματος OpenCL είναι ο πυρήνας και το πρόγραμμα ξενιστή. Οι πυρήνες εκτελούνται στην συσκευή OpenCL και το πρόγραμμα ξενιστή, στον υπολογιστή που εκτελείται το πρόγραμμα. Ο σκοπός του προγράμματος ξενιστή είναι να δημιουργήσει και να ζητήσει την πλατφόρμα και τις ιδιότητες της συσκευής, να ορίσει το περιεχόμενο, να κατασκευάσει τον πυρήνα, και να διαχειριστεί την εκτέλεση των πυρήνων. Όταν καταχωρηθεί ο πυρήνας απο τον ξενιστή στην συσκευή, δημιουργείται ένα N διαστάσεων χώρος ευρετηρίου, με το N να είναι από 1 έως 3. Κάθε περιστατικό πυρήνα δημιουργείται στις συντεταγμένες του χώρου ευρετηρίου. Αυτό το περιστατικό ονομάζεται αντικείμενο εργασίας και ο χώρος ευρετηρίου καλείται NDRange.

2.3.6 TODO: Hello word

Σε αυτήν την υπό-ενότητα θα δούμε πώς εκτελείται ένα πρόγραμμα OpenCL.

2.3.7 WebCL

Το WebCL 1.0 ορίζει ένα Javascript binding στο πρότυπο OpenCL για ετερογενή παράλληλο υπολογισμό. Το WebCL επιτρέπει σε εφαρμογές ιστού να εκμεταλλευτούν τις δυνατότητες της GPU και τον παράλληλο υπολογισμό πολυπύρηνων CPU, μέσα απο έναν Web Browser, ενεργοποιώντας σημαντική επιτάχυνση των εφαρμογών όπως επεξεργασία βίντεο και εικόνας, και ανώτερης εξομοίωσης φυσικής για παιχνίδια WebGL. Το WebCL έχει αναπτυχτεί σε στενή συνεργασία με την κοινότητα του web, και παρέχει την δυνατότητα να επεκταθούν οι δυνατότητες των HTML5 browsers ώστε να επιταχύνουν τις εφαρμογές υψηλών απαιτήσεων υπολογισμού και πλούσιου οπτικού υπολογισμού.

Σχήμα 2.3: Λογότυπο WebCL

- Khronos Launching new WebCL initiative
  - Ανακοινώθηκε τον Μάρτιο του 2011
  - API definitions already underway
- Javascript binding για OpenCL
  - Η ασφάλεια πρώτη προτεραιότητα
- Πολλές περιπτώσεις χρήσης
  - Μηχανές φυσικής για συμπλήρωση του WebGL
  - Επεξεργασία εικόνας και βίντεο σε browser
- Πολύ στενή σχέση με το πρότυπο OpenCL

- Maximum flexibility
- Foundation for higher-level middleware

2.4 Compute Shaders

Τα shaders υπολογισμού είναι μια κατάσταση shader που χρησιμοποιείται σχεδόν αποκλειστικά για υπολογισμούς αυθαίρετης πληροφορίας.Αν και μπορεί να χρησιμοποιηθεί για απόδοση, συνήθως χρησιμοποιείται για διεργασίες που δεν σχετίζονται άμεσα με σχεδιασμό τριγώνων και pixel.[**computeshaders-1**]

2.4.1 Shaders

Εισαγωγή

Στον τομέα των γραφικών υπολογιστών, ένα shader είναι ένα πρόγραμμα που χρησιμοποιείται για να εκτελέσει την λεγόμενη σκίαση: την παραγωγή συγκεκριμένου επιπέδου χρώματος μέσα σε μια εικόνα, ή την παραγωγή ειδικών εφέ ή μετατροπές βίντεο. Ένας όρος που περιγράφει την σκίαση είναι "ένα πρόγραμμα που μαθαίνει τον υπολογιστή πώς να ζωγραφίσει κάτι με έναν ειδικό και μοναδικό τρόπο".
Τα shader υπολογίζουν αποδόσεις εφέ σε υλικό γραφικών, με ένα μεγάλο βαθμό ευλυγισίας. Τα περισσότερα shader είναι σχεδιασμένα για χρήση σε μονάδα επεξεργασίας γραφικών (GPU), όμως αυτό δεν είναι αποκλειστική ανάγκη. Οι γλώσσες σκίασης, χρησιμοποιούνται συνήθως για να προγραμματίσουν την γραμμή σωλήνα απόδοσης της GPU. Η θέση, η απόχρωση, ο κορεσμός, η φωτεινότητα, και η αντίθεση όλων των στοιχείων, κορυφών, ή υφών, χρησιμοποιούνται για να αποδώσουν μια τελική εικόνα που μπορούμε να επεξεργαστούμε απευθείας με χρήση αλγορίθμων ορισμένων στα shader, είτε με αλλαγές απο εξωτερικές μεταβλητές που εισάγει το πρόγραμμα το οποίο καλεί τον shader. Τα shader χρησιμοποιούνται πολύ στην κινηματογραφική επεξεργασία, στις εικόνες που αποδίδονται απο τον υπολογιστή, αλλα και σε παιχνίδια υπολογιστών, για να παράγουν ένα μεγάλο αριθμό απο εφέ. Εκτός απο τα απλά μοντέλα φωτισμού, μερικά απο τα πολύπλοκα εφέ επεξεργάζονται την εικόνα και προσθέτουν blur, light bloom, volumetric lightning, normal mapping, bokeh, cel shading,posterization,bump mapping,distortion,chroma keying, edge detection, motion detection, κ.α
Η σύγχρονη χρήση των shader ξεκίνησε απο την Pixar, τον Μάιο του 1988. Όσο οι μονάδες επεξεργασίας γραφικών εξελίσσονταν, οι γνωστές βιβλιοθήκες γραφικών ξεκίνησαν να υποστηρίζουν τα shader. Οι πρώτες κάρτες γραφικών υποστήριζαν μόνο pixel shader, αλλά σύντομα ακολούθησε η εισαγωγή των vertex shader όταν οι προγραμματιστές κατάλαβαν τις δυνατότητες τους. Τα shader γεωμετρίας εισήχθηκαν μόλις με το Direct3D 10 και το OpenGL 3.2

Τύποι

Υπάρχουν διάφοροι τύποι shader που χρησιμοποιούνται γενικά. Ενώ οι παλιές κάρτες γραφικών είχαν ξεχωριστό τρόπο επεξεργασίας στοιχείων

για κάθε τύπο shader, οι καινούριες έχουν ενωμένους shader που έχουν την δυνατότητα να εκτελούν οποιονδήποτε τύπο shader. Αυτό επιτρέπει στις κάρτες γραφικών να έχουν πιο αποδοτική χρήση της επεξεργαστικής τους δύναμης.

- Vertex shaders - Μετατρέπουν κάθε θέση τρισδιάστατη στον εικονικό χώρο σε δισδιάστατη. Μπορούν να επεξεργαστούν ιδιότητες όπως η θέση, το χρώμα, και συντεταγμένες υφής, αλλα δεν μπορούν να δημιουργήσουν καινούρια vertices.
- Pixel shaders - Υπολογίζουν το χρώμα και άλλες ιδιότητες ενός τεμαχίου. Οι απλές μορφές τους αποδίδουν ένα pixel εικόνας, ενώ οι πιο πολύπλοκες μορφές αποδίδουν πολλά. Στα τρισδιάστατα γραφικά, ένας pixel shader δεν μπορεί να παράγει πολύπλοκα εφέ, γιατί επεξεργάζεται μόνο ένα τεμάχιο, χωρίς κάποια γνώση της γεωμετρίας της οθόνης. Μπορούν όμως να εφαρμοστούν σε εφέ δύο διαστάσεων και χρησιμοποιούνται για επεξεργασία υφών. Για παράδειγμα είναι ο μόνος τύπος shader που μπορεί να λειτουργήσει σαν φίλτρο για μια ροή βίντεο.
- Geometry shaders - Τα shader γεωμετρίας είναι σχετικά καινούριος τύπος shader. Μπορεί να δημιουργήσει γραφικά αρχικά στοιχεία όπως γραμμές, τρίγωνα. Τα shader γεωμετρίας εκτελούνται μετά απο τα vertex shader. Τυπικές χρήσεις των shader γεωμετρίας συμπεριλαμβάνουν γεωμετρική ψηφίδωση, εξώθηση σκιώδους όγκου, και απόδοση σε χάρτη κύβου. Για παράδειγμα, σε μια καμπυλωτή γραμμή, τα δεδομένα των στοιχείων εισάγονται σαν είσοδος στον shader, και αυτός αναλαμβάνει να δημιουργήσει αυτόματα επιπλέον γραμμές που δίνουν πιο εύστοχη απεικόνιση της καμπύλης.
- Tesselation shaders - Με την κυκλοφορία του OpenGL 4.0 και του Direct3D 11, ένας καινούριος τύπος shader έχει προστεθεί που ονομάζεται shader ψηφίδωσης. Επιτρέπει στα αντικείμενα κοντά στην κάμερα να έχουν καλύτερη λεπτομέρεια, ενώ τα αντικείμενα που είναι πιο μακριά να έχουν μικρότερη λεπτομέρεια αλλά να μοιάζουν όμοια στην ποιότητα.
- Compute shaders - Είναι ένας τύπος shader που χρησιμοποιείται αποκλειστικά για υπολογισμό αυθαίρετης πληροφορίας. Αν και μπορεί να χρησιμοποιηθεί για απόδοση, συνήθως χρησιμοποιείται για διεργασίες που δεν σχετίζονται άμεσα με σχεδιασμό τριγώνων και pixel.

2.4.2 Εισαγωγή

Τα shaders υπολογισμού λειτουργούν διαφορετικά από τις άλλες καταστάσεις shader. Όλες οι καταστάσεις shader έχουν προκαθορισμένου τύπου τιμές εισόδου, μερικές ενσωματωμένες και μερικές καθορισμένες από τον χρήστη. Η συχότητα στην οποία εκτελείται μια κατάσταση shader εξαρτάται από την φύση της κατάστασης. Για παράδειγμα, τα shader κορυφών εκτελούνται μια φορά για κάθε κορυφή.
Τα shader υπολογισμού λειτουργούν πολύ διαφορετικά. Ο "χώρος" στον οποίο ένα shader υπολογισμού λειτουργεί είναι αφηρημένος. Είναι στην κρίση του κάθε shader υπολογισμού να αποφασίσει τι σημαίνει αυτός ο "χώρος". Ο αριθμός των εκτελέσεων των shader υπολογισμού ορίζεται από την διεργασία που χρησιμοποιείται για να εκτελεστεί η υπολογιστική λει-

τουργία. Πιο σημαντικό από όλα, τα shader υπολογισμού δεν έχουν εισόδους καθορισμένες απο τον χρήστη και ούτε καμία έξοδο. Οι ενσωματωμένες είσοδοι ορίζουν μόνο το πού στον "χώρο" της εκτέλεσης βρίσκεται ένας συγκεκριμένος shader υπολογισμού.
Έτσι, αν κάποιος shader υπολογισμού πρέπει να πάρει κάποιες τιμές σαν είσοδο, είναι στην ευθύνη του shader να αποκτήσει τα δεδομένα, μέσω πρόσβασης υφών, αυθαίρετης φόρτωσης εικόνας, ή άλλες μορφές διεπαφής. Παρομοίως, αν ένας shader υπολογισμού υπολογίζει κάτι, θα πρέπει να το αποθηκεύσει σε μια εικόνα ή σε ένα block αποθήκευσης shader.

2.4.3 Χώρος υπολογισμού

Ο χώρος στον οποίο λειτουργεί ένα shader υπολογισμού είναι αφηρημένος. Υπάρχει η έννοια της ομάδας εργασίας. Είναι ο μικρότερος αριθμός από λειτουργίες υπολογισμού τις οποίες μπορεί να εκτελέσει ο χρήστης. Ο αριθμός των ομάδων εργασίας με τον οποίο μια λειτουργία υπολογισμού εκτελείται, ορίζεται απο τον χρήστη όταν επικαλείται την λειτουργία υπολογισμού. Ο χώρος αυτών των ομάδων είναι τρισδιάστατος, οπότε έχει ένα αριθμό απο ομάδες "X","Y","Z". Κάθε ένας απο αυτούς μπορεί να είναι 1, οπότε είναι δυνατή η εκτέλεση λειτουργιών υπολογισμού δυο ή και μίας διάστασης αντί για τρεις διαστάσεις. Αυτό είναι χρήσιμο για την επεξεργασία δεδομένων εικόνας ή γραμμικών πινάκων ενός συστήματος.
Η κάθε ομάδα εργασίας μπορεί να αποτελείται απο πολλούς shader υπολογισμού. Αυτό ονομάζεται τοπικό μέγεθος της ομάδας εργασίας. Κάθε shader υπολογισμού έχει τρισδιάστατο τοπικό μέγεθος (το οποίο μπορεί να είναι 1 επιτρέποντας δισδιάστατη ή μονοδιάστατη επεξεργασία). Αυτό ορίζει τον αριθμό των επικλήσεων ενός shader που θα εκτελεστούν σε κάθε ομάδα εργασίας. Για παράδειγμα, αν το τοπικό μέγεθος ενός shader υπολογισμού είναι (128, 1, 1) και εκτελεστεί με έναν αριθμό ομάδων εργασίας (16, 8 , 64) τότε θα έχουμε 1,048,576 ξεχωριστές επικλήσεις shader. Κάθε επίκληση θα έχει ένα σετ από εισόδους που αναγνωρίζουν μοναδικά την κάθε επίκληση. Αυτός ο διαχωρισμός είναι χρήσιμος για διάφορες μορφές συμπίεσης και αποσυμπίεσης εικόνας. Το τοπικό μέγεθος θα είναι το μέγεθος ενός block δεδομένων εικόνας (8x8 για παράδειγμα), ενώ ο αριθμός των ομάδων θα είναι το μέγεθος της εικόνας διαιρούμενο με το μέγεθος του block. Κάθε block κατεργάζεται σαν μια μοναδική ομάδα εργασίας.

2.4.4 Υλοποίηση OpenGL

Αποστολή

Ένα αντικείμενο προγράμματος μπορεί να έχει shader υπολογισμού μέσα του. Ο shader υπολογισμού συνδέεται με καταστάσεις shader μέσω κάποιων λειτουργιών απόδοσης. Υπάρχουν δύο λειτουργίες για να ξεκινήσουν οι διαδικασίες υπολογισμού. Χρησιμοποιούν οποιονδήποτε shader υπολογισμού είναι ενεργός. Οι λειτουργίες είναι οι εξής:

- void glDispatchCompute(GLuint num_groups_x, GLuint num_groups_u, GLuint num_groups_z); - Οι παράμετροι num_groups_* ορίζουν τον αριθμό των ομάδων εργασίας, σε τρεις διαστάσεις. Αυτοί οι αριθμοί δεν μπορούν να

είναι μηδέν. Υπάρχουν όρια στον αριθμό των ομάδων εργασίας που μπορούν να αποσταλούν.
- void glDispatchComputeIndirect(GLintptr indirect); - Η παράμετρος indirect είναι το αντιστάθμισμα του buffer GL_DISPATCH_INDIRECT_BUFFER. Ισχύουν τα ίδια όρια του αριθμού ομάδων εργασίας, όμως η αποστολή indirect παρακάμπτει τον έλεγχο λαθών του OpenGL. Έτσι, η αποστολή με εκτός ορίων μεγέθους ομάδας εργασίας, μπορεί να προκαλέσει προβλήματα ακόμα και πάγωμα του συστήματος.

Είδοδοι

Τα shader υπολογισμού δεν μπορούν να έχουν μεταβλητές καθορισμένες απο τον χρήστη. Τα shader υπολογισμού έχουν τις παρακάτω ενσωματωμένες μεταβλητές εξόδου:

- in uvec3 gl_NumWorkGroups; - Αυτή η μεταβλητή περιέχει τον αριθμό των ομάδων εργασίας για την λειτουργία αποστολής
- in uvec3 gl_WorkGroupID; - Αυτή η μεταβλητή περιέχει την ισχύουσα ομάδα εργασίας για την επίκληση του shader.
- in uvec3 gl_LocalInvocationID; - Αυτή η μεταβλητή περιέχει την ισχύουσα επίκληση του shader μέσα στην ομάδα εργασίας.
- in uvec3 gl_GlobalInvocationID; - Αυτή η μεταβλητή αναγνωρίζει μοναδικά την συγκεκριμένη επίκληση του shader υπολογισμού ανάμεσα σε όλες τις επικλήσεις της κλήσης αποστολής υπολογισμού. Είναι μια συντόμευση για τον μαθηματικό υπολογισμό gl_WorkGroupID * gl_WorkGroupSize + gl_LocalInvocationID;
- in uint gl_LocalInvocationIndex;

Τοπικό μέγεθος

Το τοπικό μέγεθος ενός shader υπολογισμού ορίζεται απο τον shader, χρησιμοποιώντας μια ειδική δήλωση εισόδου: layout(local_size_x = X, local_size_y = Y, local_size_z = Z) in; Αρχικά, τα τοπικά μεγέθη είναι 1, οπότε αν θέλουμε μονοδιάστατο ή δισδιάστατο χώρο ομάδων εργασίας, μπορούμε να ορίσουμε μόνο το X ή το X και το Y. Πρέπει να είναι σταθερές εκφράσεις τιμής μεγαλύτερης του 0. Οι τιμές πρέπει να ορίζονται σε σχέση με τους περιορισμούς που υπάρχουν παρακάτω. Σε αντίθετη περίπτωση προκύπτουν λάθη. Το τοπικό μέγεθος είναι διαθέσιμο στον shader σαν σταθερά, οπότε δεν χρειάζεται να την ορίζουμε εμείς.

- const uvec3 gl_WorkGroupSize;

Περιορισμοί

Ο αριθμός των ομάδων εργασίας που μπορούν να αποσταλούν, ορίζεται από την GL_MAX_COMPUTE_WORK_GROUP_COUNT. Αυτή η σταθερά πρέπει να διαβαστεί απο την glGetIntegeri_v, με τιμές ανάμεσα στο κλειστό όριο [0,2]. Προσπάθεια να καλέσουμε την glDispatchCompute με τιμές που ξεπερνούν το όριο είναι λάθος. Προσπάθεια κλήσης της glDispatchComputeIndirect είναι χειρότερα, μπορεί να διακόψει την λειτουργία του προγράμματος ακόμα

και να παγώσει το σύστημα. Σημείωση: ο μικρότερος αριθμός αυτών των τιμών πρέπει να είναι 65535 σε όλους τους άξονες. Αυτό δίνει αρκετό χώρο για εργασία. Υπάρχουν όρια στο τοπικό μέγεθος επίσης. Συγκεκριμένα, υπάρχουν δύο τύποι περιορισμών.

- Ο γενικός περιορισμός των διαστάσεων τοπικού μεγέθους, σε συνδυασμό με την GL_MAX_COMPUTE_WORK_GROUP_SIZE, όπως και παραπάνω. Η διαφορά είναι οτι ο μικρότερος αριθμός των τιμών είναι πολύ μικρότερος. 1024 για τον X και τον Y, και μόνο 64 για τον Z.
- Ο αριθμός των επικλήσεων μέσα σε μια ομάδα εργασίας. Δηλαδή, το προϊόν των στοιχείων X,Y,Z του τοπικού μεγέθους πρέπει να είναι μικρότερο απο GL_MAX_WORK_GROUP_INVOCATIONS. Η μικρότερη τιμή είναι 1024.

Υπάρχει ακόμα ο περιορισμός του ολικού μεγέθους αποθήκευσης για όλες τις κοινές μεταβλητές ενός shader υπολογισμού. Ορίζεται απο την GL_MAX_COMPUTE_SHAR
που αναφέρεται σε bytes. Η μικρότερη τιμή για το OpenGL είναι 32KB.

2.4.5 Υλοποίηση DirectX

Ένας shader υπολογισμού είναι μια κατάσταση shader υπολογισμού που εξαπλώνει το Microsoft Direct3D 11 πέρα απο τον προγραμματισμό γραφικών. Η τεχνολογία αυτή είναι γνωστή και ως τεχνολογία DirectCompute[**computeshaders-4**
]
Όπως όλα τα προγραμματιστικά shader (για παράδειγμα shader γεωμετρίας και κορυφών), ένα shader υπολογισμού είναι σχεδιασμένο να χρησιμοποιεί μια Γλώσσα Υψηλού Προγραμματισμού Shader(HLSL) για το DirectX. Η HLSL, χρησιμοποιείται για το DirectX και μας δίνει την δυνατότητα να δημιουργήσουμε C like shaders για την γραμμή σωλήνων Direct3D. Η HLSL δημιουργήθηκε ξεκινώντας απο το DirectX 9 για την κατασκευή προγραμματιζόμενων τρισδιάστατων γραμμής σωλήνα. Μας δίνει την δυνατότητα να προγραμματίσουμε την γραμμή σωλήνα με τον συνδυασμό οδηγιών assembly, οδηγιών HLSL, και δηλώσεις καθορισμένων λειτουργιών.
Ένα shader υπολογισμού προμηθεύει υψηλής ταχύτητας υπολογισμούς γενικού προγραμματισμού, και εκμεταλλεύεται τον μεγάλο αριθμό παράλληλων επεξεργαστών που βρίσκονται στην μονάδα επεξεργασίας γραφικών (GPU). Τα shader υπολογισμού προμηθεύει διαμοιρασμό μνήμης και συγχρονισμό νημάτων, για να επιτρέψει καλύτερες μεθόδους παράλληλου προγραμματισμού. Με την κλήση των μεθόδων ID3D11DeviceContext::Dispatch ή ID3D11DeviceContext::DispatchIndirect γίνεται η εκτέλεση εντολών σε ένα shader υπολογισμού, οι οποίες μπορούν να εκτελεστούν παράλληλα σε πολλά νήματα.

2.5 PathScale Enzo

Εφαρμογές

3.1 Εισαγωγή

Τα τελευταία περίπου 20 χρόνια οι εταιρίες παραγωγής υλικού γραφικών έχουν εστιάσει στην προσπάθεια να παράγουν γρήγορες μονάδες γραφικής επεξεργασίας (GPU), ειδικότερα για την κοινότητα των gamer. Αυτό έχει ως αποτέλεσμα πρόσφατα να δημιουργηθούν συσκευές οι επιδόσεις των οποίων ξεπερνούν τις κεντρικές μονάδες επεξεργασίας (CPU), σε συγκεκριμένες εφαρμογές, ειδικότερα σε μετρήσεις εκατομμυρίων εντολών το δευτερόλεπτο (MIPS). Έτσι, καθιερώθηκε μια κοινότητα για να αξιοποιήσει αυτήν την μεγάλη δύναμη των GPU για υπολογισμούς γενικής χρήσης(GPGPU). Τα τελευταία δύο χρόνια έχουν εξαλειφθεί οι περισσότεροι περιορισμοί που υπήρχαν όσον αφορά το σετ εντολών και την διαχείριση μνήμης, με την ενσωμάτωση ενοποιημένων υπολογιστικών μονάδων στις κάρτες γραφικών, δίνοντας έτσι την δυνατότητα στους προγραμματιστές να δημιουργήσουν ένα πλήθος από προγράμματα με εφαρμογές σε πολλούς τομείς.

3.2 Κρυπτογράφηση

Στο πεδίο της ασύμμετρης κρυπτογράφησης, η ασφάλεια όλων των πρακτικών κρυπτοσυστημάτων βασίζεται στην δυσκολία υπολογισμού προβλημάτων, εξαρτημένη από την επιλογή των παραμέτρων. Με την όποια αύξηση των παραμέτρων όμως (συνήθως στο εύρος 1024-4096 bits), οι υπολογισμοί γίνονται όλο και πιο απαιτητικοί για τον εκάστοτε επεξεργαστή. Σε σύγχρονο υλικό, ο υπολογισμός μιας μονής εντολής κρυπτογράφησης δεν είναι κρίσιμος, όμως σε ένα σύστημα επικοινωνίας πολλών-προς-ένα, για παράδειγμα ένας κεντρικός server στο κέντρο δεδομένων μιας εταιρίας, μπορεί να αντιμετωπίσει ταυτόχρονα εκατοντάδες η και χιλιάδες ταυτόχρονες συνδέσεις και εντολές κρυπτογράφησης. Ως αποτέλεσμα, η πιο συνήθης λύση για ένα τέτοιο σενάριο είναι η χρήση καρτών επιτάχυνσης κρυπτογράφησης. Λόγω της μικρής αγοράς, η τιμή τους φτάνει συνήθως αρκετά χιλιάδες ευρώ η δολάρια.
Τελευταία, η ερευνητική κοινότητα έχει αρχίσει να εξερευνά τεχνικές για επιτάχυνση των αλγορίθμων κρυπτογράφησης με χρήση της GPU.

3.3 Βιοπληροφορική

3.3.1 Εισαγωγή

Η συνεχής αύξηση της ποσότητας βιολογικών δεδομένων, η ανάγκη για ανάλυση τους και το συνεχές ενδιαφέρον από την επιστημονική κοινότητα για την κατανόηση των δομικών λειτουργιών των βιολογικών μορίων, αποτέλεσαν τους κύριους λόγους για την ανάπτυξη της βιοπληροφορικής. Για να κατανοήσουμε τις κυτταρικές και βιομοριακές λειτουργίες, τα βιολογικά δεδομένα πρέπει να συνενωθούν για να σχηματίσουν μια ακριβής εικόνα. Οι ερευνητές της βιοπληροφορικής, έχουν αναπτύξει υπολογιστικές τεχνικές για την επεξεργασία των βιολογικών δεδομένων, όπως νουκλεοτιδικές αλληλουχιών, αλληλουχίες αμινο οξέων, τρισδιάστατων δομών, όπως επίσης βιολογικών σημάτων και εικόνων. Μεγάλες ερευνητικές προσπάθειες του πεδίου συμπεριλαμβάνουν αναγνώριση προτύπων, ευθυγράμμιση αλληλουχιών, ανάλυση πρωτεϊνικών δομών, φυλογενητική ανάλυση, μοριακή δυναμική, ανάλυση γονιδιώματος, σχεδιασμός φαρμάκων και ανάπτυξη φαρμάκων. Επίσης, υπάρχουν προφητικές τεχνικές ειδικές για τις εκφράσεις γονιδίων, και την αλληλεπίδραση πρωτεϊνών.

Η Βιοπληροφορική παίζει μεγάλο ρόλο σε πολλές πτυχές της βιολογίας.

Σχήμα 3.1: Βιολογία και πληροφορική

Στην πειραματική μοριακή βιολογία, οι τεχνικές βιοπληροφορικής όπως επεξεργασία εικόνας και σήματος, επιτρέπει την εξόρυξη χρήσιμων αποτελεσμάτων από μεγάλο όγκο δεδομένων. Στο πεδίο της γενετικής και γονιδιωματικής, συμβάλλει στην αλληλουχία και υποσημείωση γονιδιωμάτων και την παρατήρηση των μεταλλάξεων τους. Παίζει μεγάλο ρόλο στην εξόρυξη τεχνικών όρων και στην κατασκευή βιολογικών και γονιδιακών οντολογιών για την οργάνωση και αναζήτηση βιολογικών δεδομένων. Έχει επίσης μεγάλο ρόλο στην ανάλυση των γονιδίων και στην ρύθμιση πρωτεϊνών. Τα εργαλεία της Βιοπληροφορικής συμβάλουν στην σύγκριση γενετικών και γονιδιακών δεδομένων και γενικότερα στην κατανόηση των αναπτυξιακών πτυχών της μοριακής βιολογίας. Σε πιο εσωτερικό επίπεδο, συμβάλει στην ανάλυση και κατηγοριοποίηση των βιολογικών διαδρόμων και δικτύων τα οποία είναι σημαντικό κομμάτι της συστεμικής βιολογίας. Στην Δομική βιολογία, συμβάλει στην εξομοίωση και μοντελισμό του DNA, RNA, και δομές πρωτεϊνών όπως

και μοριακών αλληλεπιδράσεων.

3.3.2 Μοριακή δυναμική

Η Βιοπληροφορική είναι ένα επιστημονικό πεδίο που εστιάζει στην εφαρμογή της τεχνολογίας υπολογιστών στην διαχείριση βιολογικών δεδομένων. Με το πέρασμα του χρόνου, οι εφαρμογές βιοπληροφορικής έχουν χρησιμοποιηθεί για να αποθηκεύσουν, αναλύσουν και να ενσωματώσουν βιολογικές και γενετικές πληροφορίες, χρησιμοποιώντας ένα μεγάλο εύρος μεθοδολογιών. Μια από τις πλέον γνωστές τεχνικές για την κατανόηση των φυσικών κινήσεων των ατόμων και των μορίων, είναι η μοριακή δυναμική. Η μοριακή δυναμική είναι μια μέθοδος εξομοίωσης των φυσικών κινήσεων των ατόμων και των μορίων κάτω από συγκεκριμένες συνθήκες. Έχει ρόλο κλειδί σε επιστήμες όπως η βιολογία, η χημεία, η φυσική, ιατρική. Λόγω της πολυπλοκότητας τους, οι υπολογισμοί της μοριακής δυναμικής χρειάζονται μεγάλες ποσότητες μνήμης και υπολογιστικής δύναμης, και για αυτό η εκτέλεση τους είναι συχνά μεγάλο πρόβλημα.
Οι εξομοιώσεις της μοριακής δυναμικής χρησιμοποιούν πολύπλοκους αριθμητικούς υπολογισμούς, που πολλές φορές οδηγούν σε αριθμητικά λάθη. Πριν την ανακάλυψη του προγραμματισμού γενικής χρήσης, οι GPU χρησιμοποιούνταν μόνο για διαδικασίες απεικόνισης των μοριακών δομών, και η εκτέλεση των αλγορίθμων μοριακής δυναμικής μπορούσε να διαρκέσει από ώρες, έως και μέρες. Η λύση προήλθε από το GPGPU, καθώς οι GPU έχουν πολλές αριθμητικές μονάδες που μπορούν να εκτελεστούν παράλληλα.
Στο πεδίο της μοριακής δυναμικής, έχουν αναπτυχθεί πολλές εφαρμογές εφαρμογές βασισμένα στο GPGPU, που υποστηρίζουν εξομοιώσεις σε πολλαπλές μονάδες. Αυτή η καινοτομία δημιουργεί ευκαιρίες για το μέλλον, ειδικά για μικρότερες ερευνητικές ομάδες. Μειώνει τον χρόνο που απαιτείται για διαδικασίες και τα απαραίτητα κονδύλια για έρευνα-ανάπτυξη, προάγει την ανάπτυξη καινούριων εφαρμογών και την επιστημονική πρόοδο.

Μετάβαση από CPU σε GPU

Η διαφορά στην αρχιτεκτονική μεταξύ CPU και GPU, είναι ότι στην τελευταία είναι δυνατή η εκτέλεση πολλαπλών παράλληλων διεργασιών, κάτι που επιτρέπει την καλύτερη εκτέλεση πολύπλοκων αλγορίθμων και καλύτερη διαχείριση μεγάλου όγκου δεδομένων. Επίσης, μια μονάδα επεξεργασίας γραφικών έχει λιγότερες ενεργειακές απαιτήσεις, έτσι η δημιουργία υπερ-υπολογιστών με χρήση GPU εξαλείφει την ανάγκη για τεράστιους χώρους γεμάτους με υπολογιστές. Η εγκατάσταση μιας επιπλέον μονάδας, αντιγράφει τον παραλληλισμό του προγραμματισμού, χωρίς καμία επιπλέον ενέργεια. Επιπλέον, η GPU έχει εντυπωσιακές δυνατότητες υπολογισμού floating point και μεγάλο εύρος ζώνης μνήμης, δίνοντας την δυνατότητα για βελτιστοποιημένη πρόσβαση στην μνήμη, ελεγχόμενη εκτέλεση επιλογών, και διαχείριση πόρων, με χρήση λίγων γραμμών κώδικα. Συγκεκριμένα, οι εφαρμογές μοριακής δυναμικής, κβαντικής χημείας, η απεικόνιση των αποτελεσμάτων τους, τρέχουν μέχρι και 5 φορές πιο γρήγορα.
Από την αρχή του GPU προγραμματισμού μέχρι και σήμερα, η προγραμ-

ματιστική ανάπτυξη συνεχίζεται αδιάκοπα. Ο αριθμός των εφαρμογών βασισμένων σε αρχιτεκτονικές GPU, φτάνουν τις 200, το οποίο είναι αύξηση της τάξεως πάνω από 60% μέσα σε δύο χρόνια. Οι καλύτερες εφαρμογές βασισμένες σε GPGPU έχουν σχεδιαστεί για την μοριακή δυναμική, τον σχεδιασμό φαρμάκων, κβαντική χημεία, το κλίμα, την φυσική σύμφωνα με την Nvidia[**bioinformatics-2**]

3.3.3 GPUGRID.net

> "I hope mankind will acknowledge people like you, its real heroes."
>
> Grzegorz Granowski, Volunteer & Donor

Το GPUGRID είναι ένα εθελοντικό κατανεμημένο σύστημα, το οποίο στοχεύει στην βιοϊατρική έρευνα από το πανεπιστήμιο Universitat Pompeu Fabra της Ισπανίας. Το GPUGRID αποτελείται από πολλές μονάδες επεξεργασίας γραφικών, που συνεργάζονται μεταξύ τους για να παραδώσουν υψηλών επιδόσεων εξομοιώσεις βιομορίων. Οι μοριακές εξομοιώσεις πού εκτελούνται από τους εθελοντές του, αποτελούν μερικούς απο τους πιο συνήθης τύπους εξομοιώσεων που εκτελούνται απο τους επιστήμονες του πεδίου, αλλά ταυτόχρονα είναι από τους πιο απαιτητικούς σε υπολογιστική δύναμη και συνήθως απαιτούν υπερ-υπολογιστές.

Σχήμα 3.2: Βιολογία και πληροφορική

Το σύστημα ερευνά μεταξύ άλλων τα παρακάτω προβλήματα

- Εξομοίωση της ωρίμανσης πρωτεολυτικών του HIV - Μια απο τις πιο σημαντικές πτυχές της ωρίμανσης του HIV είναι το πώς η πρωτεΐνη "ψαλιδιών", δημιουργείται. Η απάντηση σε αυτό το ερώτημα χρειάζεται εξομοιώσεις μοριακής δυναμικής στο όριο των μοντέρνων υπολογιστικών δυνατοτήτων. Το GPUGRID μας επιτρέπει να λύσουμε αυτο το πρόβλημα και έχουμε καταφέρει να δείξουμε οτι τα πρώτα "ψαλίδια" κόβονται απο το "σκοινί" που είναι δεμένα. Αυτό το γεγονός συμβαίνει στην αρχή της ωρίμανσης, και αν σταματήσουμε την ωρίμανση των πρωτεολυτικών, τότε θα σταματήσουμε και την ωρίμανση του HIV σαν σύνολο.
- Ανακάλυψη του ρόλου των μεμβρανών λιπιδίων στην δραστηριότητα ενζύμων.
- Μοριακή εξομοίωση αισθητήρων ντοπαμίνης κάτω απο φυσιολογικές ιονικές δυνάμεις.
- Αποκάλυψη των μηχανισμών αντίδρασης φαρμάκων καρκίνου παχέος εντέρου - Ο καρκίνος είναι βασικά ή ανεξέλεγκτη ανάπτυξη ιστών και

εισβολή από μεταλλαγμένα κύτταρα σε έναν οργανισμό. Σε αντίθεση με τις παραδοσιακές χημειοθεραπείες ή ραδιοθεραπείες, οι νεότερες θεραπείες στοχεύουν σε συγκεκριμένους στόχους κακοήθων κυττάρων. Αυτό επιτυγχάνεται με τον εντοπισμό ορισμένων πρωτεϊνών που εκφράζονται διαφορικά σε ογκογεννητικά κύτταρα. Με την βοήθεια του GPUGRID, επιτυγχάνεται η επεξήγηση των μοριακών μηχανισμών που συμβαίνουν στα μεταλλαγμένα μόρια των κυττάρων.

Η εκτέλεση του GPUGRID στις GPUs, καινοτομεί στον εθελοντικό υπολογισμό, παραδίδοντας εφαρμογές υπερ-υπολογιστών, σε υποδομές χαμηλού κόστους. Η απόδοση των μονάδων γραφικής επεξεργασίας, καταγράφεται και συγκρίνεται σε σχέση με άλλους χρήστες, ανάλογα με την διάρκεια ολοκλήρωσης των WU (Work Units).

Rank	User name	WU id	Timestamp (h)	GPU description
1	Retvari Zoltan*	10083132	4.59	[2] NVIDIA GeForce GTX 780 Ti (3071MB) driver: 344.11
2	valterc	10106939	4.81	NVIDIA GeForce GTX 780 Ti (3071MB)
3	petebe	10092106	5.20	[3] NVIDIA GeForce GTX TITAN (4095MB) driver: 335.28
4	Matt	10092250	5.24	[2] NVIDIA GeForce GTX 780 Ti (3072MB) driver: 344.11
5	Justin	10093651	5.31	[2] NVIDIA GeForce GTX 780 Ti (3072MB) driver: 340.52
6	jgis	10087293	5.33	NVIDIA GeForce GTX 780 Ti (3072MB) driver: 344.11
7	TJ	10106709	5.33	[2] NVIDIA GeForce GTX 780 Ti (3072MB) driver: 337.88
8	Pubodee	10106064	5.49	NVIDIA GeForce GTX 780 Ti (3072MB) driver: 344.11
9	[VENETO] sabayonino	10094096	5.50	[2] NVIDIA GeForce GTX 780 (3071MB)

Σχήμα 3.3: Βιολογία και πληροφορική

3.4 Αστροφυσική

Οι πρώτες καταγραφές της θεωρητικής αστρονομίας χρονολογούνται το 1550-1292 π.Χ Οι υπολογισμοί που βρέθηκαν σχηματισμένοι σε αιγυπτιακούς τάφους δείχνουν ότι υπήρχαν τεχνικές για την αναγνώριση και την καταγραφή προτύπων στον ουρανό, ένα πεδίο της αστρονομίας που είναι χρήσιμο ακόμα και σήμερα για την καταγραφή και χαρτογράφηση. Μπορεί οι σημερινοί ερευνητές να μην χρησιμοποιούν τα αρχαία εργαλεία, αλλά η ανάπτυξη τους έγινε σταδιακά.
Το μέγεθος των τηλεσκοπίων χρειάστηκε 400 χρόνια για να μεγαλώσει από 1 τετραγωνικό μέτρο σε 110 τετραγωνικά μέτρα. Οι ψηφιακοί υπολογιστές εμφανίστηκαν στο τέλος του 1940 με υπολογιστική ταχύτητα περίπου 100

floating point λειτουργιών το δευτερόλεπτο (FLOPS), και εξελίχθηκαν σε περίπου $3 * 10^{16}$ FLOPS σε λιγότερο από 65 χρόνια. Αυτή η επανάσταση των υπολογιστών συνεχίζεται ακόμα και σήμερα, και έχει οδηγήσει σε ένα καινούριο κομμάτι έρευνας στο οποίο οι εγκαταστάσεις δεν βρίσκονται στο ψηλότερο βουνό του κόσμου, αλλά στο διπλανό δωμάτιο. Οι Αστρονόμοι κατανόησαν γρήγορα ότι μπορούν να χρησιμοποιήσουν τους υπολογιστές για να καταγράψουν, αναλύσουν, αρχειοθετήσουν, τις τεράστιες ποσότητες πληροφοριών που καταγράφονται από τις εκστρατείες παρατηρητών.
Την μεγαλύτερη όμως επίδραση στον τρόπο με τον οποίο αντιμετωπίζουν

Σχήμα 3.4: Αστροφυσική

οι αστρονόμοι τα αναπάντητα ερωτήματα τους, έχει το πεδίο της εξομοίωσης. Με την χρήση των υπολογιστών, είναι δυνατόν να μελετήσουμε την λειτουργία του Διαγαλαξιακούς κενού, την φυσική των μαύρων τρυπών, κ.α

3.5 Άδεια χρήσης

3.5.1 Εισαγωγή

Η άδεια χρήσης λογισμικού, είναι ένα νομικό μέσο (συνήθως της μορφής του δικαίου των συμβάσεων) το οποίο ορίζει την χρήση και την διαμοίραση του λογισμικού. Σύμφωνα με τον Ευρωπαϊκό νόμο, όταν κάποιος γράφει ένα λογισμικό, η πνευματική ιδιοκτησία για αυτό το έργο προστατεύεται από τον νόμο, όπως ένα έργο λογοτεχνίας η τέχνης. Ο νόμος της πνευματικής ιδιοκτησίας, δίνει στον ιδιοκτήτη του έργου συγκεκριμένα δικαιώματα, και βάζει όρια στο πώς οι άλλοι μπορούν να χρησιμοποιήσουν το έργο. Η πνευματική ιδιοκτησία στο λογισμικό προέρχεται από την προστασία των γραπτών έργων, και είναι σημαντικό να γνωρίζουμε ότι το λογισμικό του υπολογιστή αντιμετωπίζεται από τον νόμο σαν ένα έργο λογοτεχνικό. Η ιδιοκτησία πνευματικών δικαιωμάτων ενός έργου, είτε βιβλίου είτε λογισμικού, σημαίνει ότι ο ιδιοκτήτης, δηλαδή ο εκδότης ή ο εργοδότης, αποφασίζει για το ποιος μπορεί να το αντιγράψει, να το τροποποιήσει και να το διανέμει. Εξ αρχής, μόνο ο ιδιοκτήτης μπορεί να το κάνει αυτό.[**EUPL-guideline**] Όποιος αντιγράψει,τροποποιήσει ή διανέμει έργο που ανήκει σε κάποιον άλλον χωρίς την άδεια του, μπορεί να βρεθεί αντιμέτωπος με τον νόμο.
Για να αποκτήσει κάποιος άδεια για τα παραπάνω, έχουμε συγκεκριμένες άδειες που θεωρούνται συμβόλαια, μεταξύ του εκδότη του λογισμικού και του χρήστη, ο οποίος μπορεί να το χρησιμοποιήσει όπως αναγράφεται στους όρους της άδειας χρήσης. Σημείωση: σε περίπτωση που δεν συμφωνεί ο χρήστης με τους όρους της άδειας, δεν μπορεί να χρησιμοποιήσει,αντιγράψει,τροποποιήσει, ή διανέμει το λογισμικό. Σε αντίθετη περίπτωση, παραβιάζει τον νόμο περί πνευματικών δικαιωμάτων.
Συνήθως οι άδειες χρήσεις λογισμικού εντάσσονται σε μια από τις παρα-

κάτω κατηγορίες, ενώ η διαφορά τους βρίσκεται στους όρους με τους οποίους ο τελικός χρήστης μπορεί στην συνέχεια να διαμοιράσει το λογισμικό.

Άδειες αποκλειστικής ιδιοκτησίας

Το σήμα κατατεθέν του λογισμικού αποκλειστικής ιδιοκτησίας είναι ότι ο εκδότης επιτρέπει την χρήση μιας η περισσότερων αντίτυπων του λογισμικού, κάτω από μια συμφωνητικού τύπου άδεια χρήσης (EULA), αλλά η ιδιοκτησία των αντίτυπων παραμένει στον εκδότη. Αυτό το χαρακτηριστικό αυτής της άδειας, σημαίνει ότι συγκεκριμένα δικαιώματα σχετικά με το λογισμικό έχουν παρακρατηθεί από τον εκδότη του λογισμικού. Γιαυτό είναι συνηθισμένο, στις EULA να υπάρχουν όροι που ορίζουν της χρήσεις του λογισμικού, όπως ο αριθμός των εγκαταστάσεων που επιτρέπονται από τους όρους της διανομής.
Η μεγαλύτερη επίδραση αυτού του τύπου άδειας, είναι ότι, αν η ιδιοκτησία του λογισμικού παραμένει στον εκδότη, τότε ο τελικός χρήστης πρέπει να αποδεχτεί την άδεια του λογισμικού, ενώ σε περίπτωση που δεν το κάνει, δεν μπορεί να χρησιμοποιήσει το λογισμικό. Σε αυτού του τύπου άδειες περιέχεται συνήθως και μια εκτενής λίστα με λειτουργίες που απαγορεύονται, όπως για παράδειγμα reverse engineering, ταυτόχρονη χρήση λογισμικού από πολλούς χρήστες, κ.α

Άδειες ελεύθερου και ανοιχτού κώδικα

Οι άδειες ελεύθερου και ανοιχτού κώδικα, κατατάσσονται σε μια από τις δύο κατηγορίες:

- Permissive άδειες - Αυτές οι άδειες έχουν λιγότερες απαιτήσεις για το πώς θα χρησιμοποιηθεί και διανεμηθεί το λογισμικό. Παράδειγμα permissive άδειας είναι η άδεια BSD και η άδεια MIT, που δίνουν απόλυτη ελευθερία για χρήση,μελέτη, και ιδιωτική τροποποίηση του λογισμικού, και περιέχει μόνο ελάχιστες απαιτήσεις για την διανομή. Αυτό δίνει σε κάποιον την δυνατότητα να χρησιμοποιήσει τον κώδικα σαν μέρος ενός λογισμικού κλειστού κώδικα, ή λογισμικού αποκλειστικής ιδιοκτησίας.

Σχήμα 3.5: Λογότυπο Open Source Initiative

- Copyleft άδειες - Αυτές οι άδειες έχουν ως στόχο να διατηρήσουν τις ελευθερίες που δίνονται στον χρήστη. Ένα παράδειγμα copyleft άδειας χρήσης λογισμικού, είναι το GPL. Αυτή η άδεια, στοχεύει στο να δώσει στους χρήστες την ελευθερία για χρήση,μελέτη, και ιδιωτική τροποποίηση του λογισμικού, και αν ο χρήστης εμμείνει στους όρους και προϋποθέσεις

του GPL, ελευθερία για διανομή του λογισμικού και κάθε τροποποίησης που έχει γίνει σε αυτό. Για παράδειγμα, κάθε τροποποίηση που γίνεται από τον χρήστη πρέπει να συνοδεύεται από τον πηγαίο κώδικα των αλλαγών όταν το λογισμικό διανεμηθεί, και η άδεια για οποιαδήποτε παράγωγο του έργου να μην έχει τους οποιεσδήποτε περιορισμούς εκτός από αυτούς που το GPL ορίζει.

Σχήμα 3.6: Λογότυπο Free Software Foundation

3.5.2 Επιλογή

Μετά από εκτενή αναζήτηση και έρευνα πάνω στις άδειες χρήσεις λογισμικου, αποφασίστηκε να χρησιμοποιηθούν δύο άδειες για την εκπόνηση της πτυχιακής εργασίας. Μία για τον πηγαίο κώδικα και μία για το τελικό αποτέλεσμα.

Πηγαίος κώδικας

Η άδεια χρήσης EUPL 1.1 κρίθηκε η καλύτερη επιλογή για την πτυχιακή εργασία. Η EUPL είναι μια άδεια λογισμικού ελεύθερου, ανοιχτού κώδικα. Έχει εγκριθεί το 2009 απο το Open Source Initiative, καθώς εκπληρώνει τις προϋποθέσεις του Open Source Definition (OSD)[1]. Η άδεια εκπληρώνει επίσης τις προϋποθέσεις του Free Software Foundation (FSF) κάτι το οποίο μπορεί να συνοψίσει τέσσερις σημαντικές ελευθερίες στην άδεια

- Ελευθερία για χρήση ή εκτέλεση για οποιαδήποτε χρήση και για οποιονδήποτε αριθμό χρηστών
- Ελευθερία απόκτησης του πηγαίου κώδικα (για μελέτη του τρόπου με τον οποίο λειτουργεί το λογισμικό)
- Ελευθερία αναδιανομής αντιτύπων του λογισμικού
- Ελευθερία μετατροπής, προσάρμοσης, βελτίωσης του λογισμικού σύμφωνα με συγκεκριμένες ανάγκες, και αναδιανομή αυτών των μετατροπών.

Βιβλίο

Για την επιλογή της άδειας χρησιμοποιήθηκε ο οδηγός στο site της creativecommons `http://creativecommons.org/choose/` Επέλεξα την άδεια Attribution-ShareAlike 4.0 International γιατί θεωρώ ότι η πτυχιακή εργασία μου μπορεί να βοηθήσει στο μέλλον και άλλους φοιτητές να γνωρίσουν την τεχνολογία του LaTeX,αλλά ταυτόχρονα είναι και συμβατή με την άδεια χρήσης του πηγαίου κώδικα.

3.6 Βιβλιογραφικές αναφορές

Η πρώτη βιβλιογραφική αναφορά είναι: [2], ενώ η δεύτερη είναι [3].

Βιβλιογραφία

[1] Open Source Initiative. *The Open Source Definition (Annotated)*. Online; accessed 7-September-2014. 2014. URL: http://opensource.org/docs/osd (see p. 26).

[2] Frank Mittelbach. *Is a PDF output from a LaTeX document, a "derived work" from the LPPL standard packages?* Online; accessed 6-September-2014. 2012. URL: http://tex.stackexchange.com/questions/82713/is-a-pdf-output-from-a-latex-document-a-derived-work-from-the-lppl-standard-p (see p. 27).

[3] European Parliament. *Directive 2009/24/EC of the European Parliament and of the Council of 23 April 2009 on the legal protection of computer programs*. Online; accessed 6-September-2014. 2009. URL: http://eur-lex.europa.eu/legal-content/EN/ALL/;ELX_SESSIONID=9zyGJMmVgxnf3yVpTg8B1rSJchKv390981438?uri=CELEX:32009L0024 (see p. 27).