



ΤΕΙ ΑΘΗΝΑΣ ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

**Προγραμματισμός γενικού
σκοπού σε μονάδες γραφικής
επεξεργασίας
(GPGPU)**

Φοιτητής:

ΓΕΩΡΓΙΟΣ ΣΟΦΙΑΝΟΣ

A.M: 031053

Επόπτης:

Κ. ΓΕΩΡΓΙΟΣ ΜΠΑΡΔΗΣ

Ημερομηνία παράδοσης:
28 Οκτωβρίου 2014

Περίληψη

Περίληψη

Σκοπός αυτής της εργασίας είναι η διερεύνηση και αξιολόγηση προτύπων και τεχνολογιών για προγραμματισμό γενικού σκοπού με χρήση μονάδων επεξεργασίας γραφικών, και η παρουσίαση εφαρμογών υψηλών υπολογιστικών απαιτήσεων οι οποίες εκμεταλλεύονται τις δυνατότητες αυτών των τεχνολογιών για επιτάχυνση και αύξηση των επιδόσεων τους.

Abstract

The purpose of this thesis is the research and evaluation of patterns and technologies used by general purpose programming on graphics processing units, and the demonstration of high processing computing applications which exploit the capabilities of these technologies in order to accelerate and increase their performance.

Περιεχόμενα

Περίληψη	III
Εισαγωγή	VII
0.1 Παράλληλος υπολογισμός	VIII
0.2 GPU Computing	VIII
1 Υλοποιήσεις	1
1.1 Ιστορία	1
1.1.1 Μέλλον	2
1.1.2 Προβλήματα	2
1.2 CUDA	3
1.2.1 Εισαγωγή	3
1.2.2 Πλεονεκτήματα	4
1.2.3 Περιορισμοί	4
1.2.4 Μνήμη	5
1.2.5 CUDA streams	6
1.3 OpenCL	7
1.3.1 Εισαγωγή	7
1.3.2 Επισκόπηση	8
1.3.3 Ιστορία	9
1.3.4 Στόχοι	9
1.3.5 Μοντέλο εκτέλεσης	10
1.3.6 WebCL	10
1.4 Compute Shaders	11
1.4.1 Shaders	11
1.4.2 Εισαγωγή	13
1.4.3 Χώρος υπολογισμού	13
1.4.4 Υλοποίηση OpenGL	14
1.4.5 Υλοποίηση DirectX	15
1.5 PathScale Enzo	16
2 Περισσότερα	17
2.0.1 CPUs vs GPUs	17
2.1 Συμπλέγματα GPU	18
2.1.1 VirtualCL	18
3 Εφαρμογές	21
3.1 Εισαγωγή	21
3.2 Κρυπτογράφηση	21
3.3 Βιοπληροφορική	22

3.3.1	Εισαγωγή	22
3.3.2	Μοριακή δυναμική	23
3.3.3	GPUGRID.net	24
3.4	Ψυχαγωγία	25
3.4.1	Εισαγωγή	25
3.4.2	Παιχνίδια	26
3.4.3	Υλοποιήσεις	27
3.5	Μετατροπή	28
3.5.1	Εισαγωγή	28
3.5.2	Μειονεκτήματα	28
3.5.3	Βίντεο	29
3.5.4	Συμπύεση δεδομένων γενετικής	30
3.6	Γεωπληροφορική	30
3.6.1	Γεωγραφικά πληροφοριακά συστήματα	30
3.6.2	Ωκεανοί	30
3.6.3	Σεισμική δραστηριότητα	30
3.7	Αστροφυσική	34
4	Προσδιορισμός του προβλήματος και διεθνής εμπειρία	35
5	Μεθοδολογία και Εφαρμογή	37
5.1	Εργαλεία που χρησιμοποιήθηκαν	37
5.1.1	X _Y TeX	37
5.1.2	Texmaker	37
5.1.3	TortoiseGit	38
5.1.4	Github	38
5.2	Άδεια χρήσης	39
5.2.1	Εισαγωγή	39
5.2.2	Επιλογή	41

Εισαγωγή

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

Seymour Cray

Τα αρχικά GPGPU πηγάζουν από την φράση General Purpose computation on Graphics Processing Units, ή αλλιώς γνωστή ως 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 επεξεργασίας (CPU) για αλγόριθμους όπου η επεξεργασία μεγάλου όγκου δεδομένων γίνεται παράλληλα, όπως σε αλγόριθμους sort μεγάλων λιστών, μετασχηματισμό κυμάτων δυο διαστάσεων, προσομοίωση βιολογικών δυναμικών.

Σκοπός αυτής της εργασίας είναι να διερευνήσει και αξιολογήσει πρότυπα και τεχνολογίες για προγραμματισμό γενικού σκοπού με χρήση μονάδων επεξεργασίας γραφικών, ειδικότερα όσον αφορά εφαρμογές υψηλών υπολογιστικών απαιτήσεων οι οποίες εκμεταλλεύονται τις δυνατότητες αυτών των τεχνολογιών για επιτάχυνση και αύξηση των επιδόσεων τους. Το θέμα είναι μεγάλης σημασίας

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 περισσότεροι, 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

Υλοποιήσεις

1.1 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 εξελίσσουν 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 προγραμματισμός 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 παράλ-

ληλών συσκευών, συμπεριλαμβανομένου μονάδων επεξεργασίας γραφικών, πολυπύρηνων κεντρικών μονάδων επεξεργασίας, κ.α

1.1.1 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.

1.1.2 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.

1.2 CUDA

1.2.1 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 1.1: 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 από το The Portland Group. Εκτός από 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 από τέτοια νήματα. Μια ομάδα από νήματα, που εκτελούνται παράλληλα, επικοινωνούν και συγχρονίζονται μεταξύ τους ονομάζεται block. Ο μέγιστος αριθμός των νημάτων σε ένα block είναι ένας περιορισμός που υπάρχει στην κάθε μονάδα γραφικής επεξεργασίας. Τέλος, μια ομάδα από blocks τα οποία έχουν την ίδια διάσταση και εκτελούνται από 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

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

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).

1.2.2 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 από τον χρήστη, επιτρέποντας μεγαλύτερο εύρος δεδομένων από 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.

1.2.3 Περιορισμοί

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

1.2.4 Μνήμη

Κοινή μνήμη και συγχρονισμός

Ο μεταγλωττιστής CUDA C μεταχειρίζεται τις μεταβλητές στην κοινή μνήμη με διαφορετικό τρόπο από ότι τις τυπικές μεταβλητές. Δημιουργεί ένα αντίγραφο για κάθε block που εκτελείται στην CPU. Κάθε νήμα σε αυτό το

block μοιράζεται την μνήμη, αλλά τα νήματα δεν μπορούν να προσπελάσουν και να επεξεργαστούν το αντίγραφο της μεταβλητής που φαίνεται στα άλλα blocks.

Αυτό παρέχει ένα καλό τρόπο με τον οποίο τα νήματα μέσα σε ένα block μπορούν να επικοινωνούν και να συνεργάζονται στους υπολογισμούς. Επιπλέον, τα buffers κοινής μνήμης βρίσκονται πάνω στην GPU, με αυτόν τον τρόπο, η καθυστέρηση στην προσπέλαση της κοινής μνήμης είναι πολύ μικρότερη από ότι στα τυπικά buffers, καθιστώντας την κοινή μνήμη πολύ αποδοτική.

Η επικοινωνία μεταξύ των νημάτων, αν και πολύ ενδιαφέρουσα, χρειάζεται έναν μηχανισμό για τον συγχρονισμό της. Για παράδειγμα, αν το νήμα A γράφει μια τιμή στην κοινή μνήμη και θέλουμε το νήμα B να κάνει κάτι με αυτήν την τιμή, δεν μπορούμε να ξεκινήσουμε το νήμα B έως ότου γνωρίζουμε ότι η εγγραφή από το νήμα A ολοκληρώθηκε. Χωρίς τον συγχρονισμό, θα είχαμε έναν αγώνα δρόμου όπου το σωστό αποτέλεσμα της εκτέλεσης θα εξαρτάται από μη ντετερμινιστικά χαρακτηριστικά του υλικού.

Σταθερή μνήμη

Έχουμε αναλύσει το πως οι μοντέρνες GPUs είναι εφοδιασμένες με τεράστιες δυνατότητες υπολογιστικής δύναμης. Το υπολογιστικό πλεονέκτημα που έχουν οι μονάδες επεξεργασίας γραφικών βοήθησε στην ανάπτυξη του προγραμματισμού γενικού σκοπού. Με εκατοντάδες αριθμητικές μονάδες στην GPU, συνήθως ο περιορισμός δεν είναι η αριθμητική απόδοση, αλλά το εύρος ζώνης της μνήμης. Υπάρχουν τόσες πολλές ALUs στους επεξεργαστές γραφικών, όπου πολλές φορές δεν προλαβαίνουμε να μεταφέρουμε τα δεδομένα αρκετά γρήγορα ώστε να διατηρήσουμε έναν υψηλό ρυθμό υπολογισμού.

Η γλώσσα CUDA παρέχει άλλον έναν τύπο μνήμης γνωστή ως σταθερή μνήμη. Όπως φαίνεται από το όνομα, χρησιμοποιούμε την σταθερή μνήμη για δεδομένα που δεν αλλάζουν κατά την διάρκεια μιας εκτέλεσης πυρήνα. Το υλικό NVIDIA παρέχει 64KB σταθερής μνήμης που χρησιμοποιεί με διαφορετικό τρόπο από ότι την γενική μνήμη. Σε μερικές περιπτώσεις, η χρήση της σταθερής μνήμης αντί της γενικής μνήμης μειώνει το εύρος ζώνης της μνήμης.

Δηλώνοντας την μνήμη σαν σταθερή, περιορίζουμε την χρήση της σε ανάγνωσης μόνο. Λόγω αυτού του περιορισμού, περιμένουμε να κερδίσουμε κάτι από αυτήν την διαδικασία. Η χρήση σταθερής μνήμης μπορεί να μας διαφυλάξει εύρος μνήμης σε σχέση με την ανάγνωση των δεδομένων από την γενική μνήμη. Υπάρχουν δύο λόγοι γιατί η ανάγνωση από την σταθερή μνήμη των 64KB μπορεί να διαφυλάξει εύρος μνήμης:

- Μια ανάγνωση από την σταθερή μνήμη μπορεί να αναμεταδοθεί σε κοινά νήματα, σώζοντας μας έως και 15 αναγνώσεις.
- Η σταθερή μνήμη είναι cached, έτσι επόμενες αναγνώσεις της ίδιας διεύθυνσης δεν θα δημιουργήσουν επιπλέον κίνηση στην μνήμη.

Για να εξηγήσουμε αυτήν την δήλωση και το τι είναι τα κοινά νήματα, σκεφτόμαστε την περίπτωση της ύφανσης. Στην ύφανση, το στημόνι αναφέρεται σε ένα σύνολο από νήματα, που υφαίνονται μαζί σε ένα ύφασμα.

Στην αρχιτεκτονική CUDA, το σθημόνι αναφέρεται σε μια συλλογή από 32 νήματα τα οποία υφαίνονται μεταξύ τους, και εκτελούνται αμφίδρομα. Σε κάθε γραμμή του κώδικα, κάθε νήμα από το σθημόνι εκτελεί την ίδια διαδικασία, σε διαφορετικά δεδομένα. Όσον αφορά τον χειρισμό της σταθερής μνήμης, το υλικό NVIDIA μπορεί να αναμεταδώσει μια απλή ανάγνωση σε μια ομάδα από 16 νήματα: τα μισά από τα 32 νήματα που βρίσκονται στο σθημόνι. Αν κάθε νήμα ζητάει δεδομένα από την ίδια διεύθυνση της σταθερής μνήμης, η μονάδα επεξεργασίας γραφικών θα εκτελέσει μόνο ένα αίτημα ανάγνωσης και θα αναμεταδώσει τα δεδομένα σε όλα τα νήματα. Αν η ανάγνωση γίνεται από την σταθερή μνήμη για μεγάλο όγκο πληροφοριών, θα δημιουργηθεί κίνηση που αντιστοιχεί μόνο στο 1/16 της κίνησης μνήμης που θα χρειαζόταν για την γενική μνήμη.

Τα πλεονεκτήματα όμως δεν σταματάνε εκεί. Επειδή έχουμε αποφασίσει να αφήσουμε άθικτη την μνήμη, το υλικό μπορεί να αποθηκεύσει τα σταθερά δεδομένα στην GPU. Έτσι μετά από την πρώτη ανάγνωση από την διεύθυνση της σταθερής μνήμης, όλες οι επόμενες αναγνώσεις δεν θα δημιουργήσουν επιπλέον κίνηση στην μνήμη. Δυστυχώς, μπορεί να υπάρξουν στιγμές που η απόδοση να μειώνεται λόγω της σταθερής μνήμης. Η αναμετάδοση μέρους από το σθημόνι είναι ένα δίκωπο μαχαίρι. Αν και μπορεί να επιταχύνει την απόδοση όταν τα 16 νήματα διαβάζουν την ίδια διεύθυνση, μπορεί αντίστοιχα να μειώσει την απόδοση όταν τα 16 νήματα διαβάζουν διαφορετικές διευθύνσεις. Για παράδειγμα, αν τα 16 νήματα χρειάζονται διαφορετικά δεδομένα από την σταθερή μνήμη, οι 16 αναγνώσεις θα γίνονταν με την σειρά, και θα χρειαζόταν 16 φορές περισσότερο χρόνο για να εκτελέσουν την εντολή. Αν διάβαζαν από την γενική μνήμη, η εντολή θα δινόταν άμεσα. Σε αυτήν την περίπτωση, η ανάγνωση από την σταθερή μνήμη θα ήταν πιο αργή από το να γινόταν ανάγνωση της γενικής μνήμης.

1.2.5 CUDA streams

Εξηγήσαμε το πως ο παράλληλος προγραμματισμός δεδομένων σε μια GPU μπορεί να δώσει εντυπωσιακά αποτελέσματα σε σχέση με την εκτέλεση σε CPU. Όμως υπάρχει ακόμα ένας τύπος παράλληλου υπολογισμού που μπορούμε να εκμεταλλευτούμε σε μια μονάδα επεξεργασίας γραφικών NVIDIA. Ο παραλληλισμός αυτός μοιάζει με αυτόν που συμβαίνει στα πολυ-νηματικά προγράμματα CPU. Αντί να εκτελούμε την ίδια διαδικασία σε πολλά στοιχεία δεδομένων όπως στον παραλληλισμό δεδομένων, ο παραλληλισμός έργων περιλαμβάνει παράλληλη εκτέλεση δύο ή περισσότερων έργων.

Σαν έργο μπορούμε να θεωρήσουμε μεγάλο αριθμό πραγμάτων. Για παράδειγμα, μια εφαρμογή μπορεί να εκτελεί δύο έργα: επανασχεδίαση του γραφικού περιβάλλοντος με ένα νήμα, και μεταφόρτωση μιας αναβάθμισης μέσω δικτύου με κάποιο άλλο νήμα. Αυτά τα έργα εκτελούνται παράλληλα, και ας μην έχουν τίποτε κοινό. Αν και ο παραλληλισμός έργων δεν είναι τόσο ευέλικτος όσο στις CPUs, συνεχίζει να μας προσφέρει ευκαιρίες για να αποκτήσουμε περισσότερη ταχύτητα από τις βασισμένες σε GPU εφαρμογές μας.

Τα CUDA streams, παίζουν μεγάλο ρόλο στην επιτάχυνση των εφαρμογών μας. Ένα CUDA stream αντιπροσωπεύει μια λίστα από GPU διεργασίες

που θα εκτελεσθούν με συγκεκριμένη σειρά. Οι διεργασίες μπορούν να περιλαμβάνουν εκτελέσεις πυρήνων, αντιγραφές μνήμης, και εκκινήσεις/τερματισμούς συμβάντων ενός stream. Η σειρά εισαγωγής των διεργασιών ορίζει την σειρά εκτέλεσής τους, με ευκαιρίες για αυτά τα έργα να εκτελεστούν παράλληλα.

1.3 OpenCL

1.3.1 Εισαγωγή

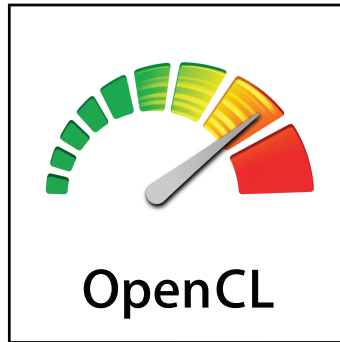
Το πρώτο GPGPU framework δημιουργήθηκε από την NVIDIA και ήταν το CUDA. Το CUDA παρείχε στους χρήστες ένα προγραμματιστικό περιβάλλον σε C like γλώσσα για την GPU. Όμως ήταν κλειστού κώδικα και μπορεί να τρέχει μόνο σε NVIDIA κάρτες γραφικών. Λόγω της μεγάλης δημοτικότητας του CUDA, η ανάγκη για ένα ανοιχτό πρότυπο αρχιτεκτονικής που θα υποστηρίζει διάφορα είδη συσκευών από 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 πολυπύρηνιο 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 από το μή-κερδοσκοπικό τεχνολογικό συνεταιρισμό Khronos Group. Έχει υιοθετηθεί από πολλές μεγάλες εταιρίες όπως η Apple, Intel, Qualcomm, AMD, Nvidia, Samsung, ARM Holdings.

1.3.2 Επισκόπηση

Το OpenCL ορίζει μια διεπαφή προγραμματισμού εφαρμογών με την οποία επιτρέπει στα προγράμματα που τρέχουν στον οικοδεσπότη να εκτε-



Σχήμα 1.2: Λογότυπο OpenCL

λέσουν πυρήνες στην συσκευή υπολογισμού, και να διαχειριστούν την μνήμη 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 από 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, εγγράφημη 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.

1.3.3 Ιστορία

Το OpenCL δημιουργήθηκε αρχικά από 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 από 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 από το Khronos Group και αποτελεί την τελευταία έκδοση του OpenCL.

1.3.4 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 από 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 από έναν μεταγλωττιστή runtime, μέσω κατάλληλου προγράμματος. Αυτό επιτρέπει στα

προγράμματα να εκμεταλλεύονται όλες τις συσκευές ενός συστήματος με ένα σετ φορητών υπολογιστικών πυρήνων.

1.3.5 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 από 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.

1.3.6 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 από έναν Web Browser, ενεργοποιώντας σημαντική επιτάχυνση των εφαρμογών όπως επεξεργασία βίντεο και εικόνας, και ανώτερης εξομοίωσης φυσικής για παιχνίδια WebGL. Το WebCL έχει αναπτυχθεί 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 1.3: 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

1.4 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]

1.4.1 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 από εξωτερικές μεταβλητές που εισάγει το πρόγραμμα το οποίο καλεί τον shader. Τα shader χρησιμοποιούνται πολύ στην κινηματογραφική επεξεργασία, στις εικόνες που αποδίδονται από τον υπολογιστή, αλλά και σε παιχνίδια υπολογιστών, για να παράγουν ένα μεγάλο αριθμό από εφέ. Εκτός από τα απλά μοντέλα φωτισμού, μερικά από 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 από 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 εισήχθησαν 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, αλλά 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 από 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.
- Tessellation 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.

1.4.2 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 συχνότητα 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 λειτουργία. Πιο σημαντικό από όλα, τα shader υπολογισμού δεν έχουν εισόδους καθορισμένες από 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.

1.4.3 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 από τον χρήστη όταν επικαλείται την λειτουργία υπολογισμού. Ο χώρος αυτών των ομάδων είναι τρισδιάστατος, οπότε έχει ένα αριθμό από ομάδες "X", "Y", "Z". Κάθε ένας από 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 από 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.

1.4.4 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_y, 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 ότι 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 από `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 από την `GL_MAX_COMPUTE_SHADER_STORAGE_SIZE` που αναφέρεται σε bytes. Η μικρότερη τιμή για το OpenGL είναι 32KB.

1.4.5 Υλοποίηση DirectX

Ένας shader υπολογισμού είναι μια κατάσταση shader υπολογισμού που εξαπλώνει το Microsoft Direct3D 11 πέρα από 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 από το DirectX 9 για την κατασκευή προγραμματιζόμενων τρισδιάστατων γραμμής σωλήνα. Μας δίνει την δυνατότητα να προγραμματίσουμε την γραμμή σωλήνα με τον συνδυασμό οδηγίων assembly, οδηγίων 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.

1.5 PathScale Enzo

Περισσότερα

2.0.1 CPUs vs GPUs

Οι GPUs έχουν δημιουργηθεί για πολύ συγκεκριμένες χρήσεις, για παράδειγμα η απεικόνιση γραφικών, πολλαπλασιασμούς πυκνού πλέγματος, απλά φίλτρα επεξεργασίας, κ.α. Είναι πολύ καλές στην διαχείριση μεγάλων καθυστερήσεων γιατί είναι σχεδιασμένες με τρόπο ώστε να ανέχονται καθυστερήσεις υφών, μια διαδικασία 1000+ κύκλων. Οι πυρήνες GPU διαθέτουν πολλά νήματα: όταν ένα νήμα εκτελέσει μια διεργασία υψηλής καθυστέρησης, (για παράδειγμα μια προσπέλαση μνήμης), αυτό το νήμα τίθεται σε αναμονή (ενώ τα υπόλοιπα νήματα συνεχίζουν να εργάζονται), έως ότου η διαδικασία τελειώσει. Αυτό επιτρέπει στις GPUs να διατηρούν τις μονάδες υπολογισμού απασχολημένες περισσότερο από ότι οι παραδοσιακοί πυρήνες.

Οι GPUs δεν αποδίδουν στον χειρισμό διακλαδώσεων, γιατί συνήθως δημιουργούν δεσμίδες από νήματα σε σημίονι, και τα αποστέλλουν στην γραμμή σωλήνα για λόγους οικονομίας εντολών προσπέλασης/αποκωδικοποίησης. Αν τα νήματα συναντήσουν κάποια διακλάδωση, μπορεί να αποκλίνουν, π.χ 2 νήματα σε ένα σημίονι 8 νημάτων μπορεί να ακολουθήσουν την διακλάδωση, ενώ τα άλλα 6 μπορεί να μην την ακολουθήσουν. Τώρα το σημίονι έχει διαιρεθεί σε δύο σημίονια μεγέθους 2 και 6, τα οποία δεν θα εκτελούνται αποδοτικά. Το σημίονι των 2 νημάτων θα εκτελείται με 25% απόδοση, και το σημίονι των 6 νημάτων θα εκτελείται με 75% απόδοση. Μπορούμε να φανταστούμε ότι αν μια μονάδα επεξεργασίας συνεχίσει να συναντά επιπλέον διακλαδώσεις, η απόδοση τους συνεχίζει να μειώνεται. Για αυτόν τον λόγο, οι GPUs δεν είναι καλή επιλογή για τον χειρισμό διακλαδώσεων και ο κώδικας που περιέχει αυτές θα πρέπει να εκτελείται στις CPUs.

Γενικότερα, ο κώδικας που δεν θα πρέπει να εκτελείται σε GPUs είναι κώδικας με λίγο παραλληλισμό ή κώδικας με πολλές διακλαδώσεις ή συγχρονισμό, για παράδειγμα βάσεις δεδομένων, λειτουργικά συστήματα, αλγόριθμοι γραφημάτων, κ.α.

Διαφορές προγραμματιστικού μοντέλου

Οι πιο σημαντικές διαφορές στο προγραμματιστικό μοντέλο των GPUs είναι ότι δεν υποστηρίζουν διακοπές και εξαιρέσεις. Εκτός από αυτό, δεν υπάρχουν μεγάλες διαφορές μεταξύ της CUDA, OpenCL, και C. Επίσης όπως αναφέραμε, ο κώδικας θα πρέπει να γράφεται με όσο το δυνατόν λιγότερες διακλαδώσεις και συνεχή επικοινωνία μεταξύ των νημάτων.

Πολλά προβλήματα στον πραγματικό κόσμο έχουν πολλές διακλαδώσεις και παρατυπίες. Αλγόριθμοι γραφημάτων, λειτουργικά συστήματα, web browsers, κ.α. Ακόμα και τα γραφικά χρησιμοποιούν όλο και περισσότερο διακλαδώσεις και διαδικασίες γενικού σκοπού, αναγκάζοντας έτσι τις GPUs να γίνονται όλο και περισσότερο προγραμματιζόμενες.

2.1 Συμπλέγματα GPU

2.1.1 VirtualCL

Επισκόπηση

Η πλατφόρμα VirtualCL είναι ένα περικάλυμμα για το OpenCL που επιτρέπει τις περισσότερες εφαρμογές να εκμεταλλεύονται με διαφάνεια πολλές OpenCL συσκευές σε ένα σύμπλεγμα, σαν αυτές να βρίσκονταν σε έναν προσωπικό υπολογιστή. Με την πλατφόρμα VirtualCL, οι απομακρυσμένοι κόμβοι εκτελούν διεργασίες για λογαριασμό των προγραμμάτων του ξενιστή.

Το VCL είναι ευέλικτο. Οι εφαρμογές μπορούν να δημιουργήσουν περιεχόμενο OpenCL που αποτελείται από συσκευές από διάφορους κόμβους, ή πολλά περιεχόμενα, κάθε ένα αποτελούμενο από τις συσκευές διαφορετικού κόμβου. Ή οποιονδήποτε συνδυασμό από τα παραπάνω. Άλλες εφαρμογές μπορούν να διαχωριστούν σε διάφορες ανεξάρτητες διεργασίες και νήματα, κάθε μία από τις οποίες εκτελείται σε διαφορετικό σύνολο συσκευών, ενώ ταυτόχρονα χρησιμοποιούν την κοινή μνήμη του συστήματος.

Πιο εξελιγμένες εφαρμογές μπορούν να επιλέγουν τις συσκευές που θα εκτελούν τις διεργασίες τους, όμως το VCL επιτρέπει μεταβλητές συστήματος στις οποίες ορίζονται πολιτικές για τον διαμοιρασμό των συσκευών. Η αρχική ρύθμιση, είναι κάθε περιεχόμενο που δημιουργείται να περιέχει όλες τις συσκευές ενός κόμβου. Το VCL αποτελείται από τρία μέρη, την βιβλιοθήκη VCL, τον μεσίτη, και τον back-end δαίμονα.

Βιβλιοθήκη

Η βιβλιοθήκη VCL είναι μια βιτρίνα συμπλέγματος για τις εφαρμογές OpenCL. Όταν συνδεθεί με τις εφαρμογές OpenCL, επιτρέπει πρόσβαση με διαφάνεια σε συσκευές OpenCL του συμπλέγματος, κρύβοντας την πραγματική τοποθεσία των συσκευών από τις εφαρμογές που τις καλούν. Η βιβλιοθήκη VCL είναι σχεδιασμένη να εκτελείται με τις περισσότερες εφαρμογές χωρίς επιπλέον παρεμβάσεις, έτσι μπορούμε να επιλέξουμε τον τρόπο που θα χρησιμοποιούνται οι συσκευές στο σύμπλεγμα, μέσω μεταβλητών συστήματος. Η βιβλιοθήκη VCL υποστηρίζει απόλυτα πολυ-νηματικό προγραμματισμό και παρέχει ασφάλεια νημάτων.

Η VCL βιβλιοθήκη χρησιμοποιεί διάφορους αλγόριθμους βελτιστοποίησης. Για παράδειγμα, λόγω της καθυστέρησης δικτύου, η βιβλιοθήκη προσπαθεί να βελτιστοποιήσει την απόδοση της επικοινωνίας διατηρώντας μια ανεξάρτητη βάση δεδομένων από αντικείμενα OpenCL και εκτελεί όσες λειτουργίες είναι δυνατόν στον υπολογιστή-ξενιστή, ώστε να ελαττώσει τον αριθμό των επικοινωνιών στο ελάχιστο.

Μεσίτης

Ο μεσίτης είναι μια λειτουργία δαίμονα που εκτελείται σε κάθε υπολογιστή-ξενιστή όπου οι χρήστες μπορούν να εκτελέσουν τις OpenCL εφαρμογές τους. Ο μεσίτης συνδέεται με την βιβλιοθήκη VCL μέσω υποδοχών UNIX. Η ευθύνη του περιέχει:

1. Παρακολούθηση της λειτουργικότητας και διαθεσιμότητας των συσκευών στο σύμπλεγμα.
2. Αναφορά αυτών των συσκευών στις εφαρμογές που τις ζητάνε.
3. Έξυπνη κατανομή των συσκευών για τις εφαρμογές OpenCL όταν γίνεται δημιουργία περιεχομένου, για παράδειγμα η προσπάθεια να ταιριάζει ο αριθμός των συσκευών με τον αριθμό των κόμβων που έχουν σύνολο τον αριθμό που ζητήθηκε από την εφαρμογή.
4. Πιστοποίηση αυθεντικότητας, δρομολόγηση, και εξασφάλιση της ποιότητας των μηνυμάτων μεταξύ των εφαρμογών.

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, συμβάλλει στην ανάλυση και κατηγοριοποίηση των βιολογικών διαδρόμων και δικτύων τα οποία είναι σημαντικό κομμάτι της συστημικής βιολογίας. Στην Δομική βιολογία, συμβάλλει στην εξομίωση 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 προγραμματιστική 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 που εκτελούνται από τους εθελοντές του, αποτελούν μερικούς από τους πιο συνήθεις τύπους εξομοιώσεων που εκτελούνται από 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 από 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 αυτό το πρόβλημα και έχουμε καταφέρει να δείξουμε ότι τα πρώτα "ψαλίδια" κόβονται από 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 από 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 διαφορετικά 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 Ψυχαγωγία

3.4.1 Εισαγωγή

Μια μηχανή φυσικής είναι ένα πρόγραμμα υπολογιστή που παρέχει εξομοίωση συγκεκριμένων συστημάτων φυσικής, όπως δυναμική άκαμπτων σωμάτων, ανίχνευση σύγκρουσης, δυναμική υγρών, για χρήση σε πεδία όπως τα γραφικά υπολογιστών, παιχνίδια, κινούμενα σχέδια, ταινίες. Μια από τις κύριες χρήσεις τους είναι στα παιχνίδια υπολογιστών, στην οποία περίπτωση η εξομοίωση γίνεται σε πραγματικό χρόνο. Ο όρος χρησιμοποιείται γενικότερα για να περιγράψει οποιοδήποτε σύστημα λογισμικού που εξομοιώνει φυσικά φαινόμενα, όπως επιστημονικές εξομοιώσεις υψηλής απόδοσης.

Οι μηχανές φυσικής έχουν χρησιμοποιηθεί αρκετά στους υπερ-υπολογιστές από την δεκαετία του '80 για να εκτελέσουν μοντελοποίηση δυναμικών υγρών, όπου αναθέτουμε διανύσματα ισχύος σε σωματίδια, για να δείξουμε την κυκλοφορία. Λόγω των υψηλών απαιτήσεων σε ταχύτητα και ακρίβεια, ειδικοί επεξεργαστές δημιουργήθηκαν που είναι γνωστοί ως επεξεργαστές διανυσμάτων για να επιταχύνουν τους υπολογισμούς. Οι τεχνικές μπορούν να χρησιμοποιηθούν για να μοντελοποιήσουν πρότυπα καιρού για την πρόβλεψη καιρού, δεδομένα σήραγγας αέρα για σχεδιασμό αεροπλάνων

και υποβρυχίων, και ανάλυση θερμικής απόδοσης για καλύτερο σχεδιασμό ψηκτρών για επεξεργαστές. Φυσικά μεγάλο ρόλο παίζει η ακρίβεια των υπολογισμών, αφού μικρές αποκλίσεις μπορούν να αλλάξουν δραστικά τα αποτελέσματα των υπολογισμών. Οι κατασκευαστές ελαστικών χρησιμοποιούν εξομοιώσεις φυσικής για να μελετήσουν πώς οι καινούριοι τύποι ελαστικών θα αποδίδουν σε συνθήκες βρεγμένου και στεγνού οδοστρώματος, χρησιμοποιώντας καινούρια υλικά και κάτω από διαφορετικές συνθήκες βάρους.

Υπάρχουν γενικά δύο τύποι μηχανών φυσικής. Οι πραγματικού χρόνου, και οι υψηλής ακρίβειας. Οι υψηλής ακρίβειας απαιτούν περισσότερη υπολογιστική δύναμη για να υπολογίσουν φυσικά φαινόμενα με ακρίβεια και χρησιμοποιούνται συνήθως από επιστήμονες αλλά και σε κινούμενα σχέδια. Οι πραγματικού χρόνου - χρησιμοποιούνται σε παιχνίδια υπολογιστών και σε άλλες μορφές διαδραστικού υπολογισμού - χρησιμοποιούν απλοποιημένους υπολογισμούς με μειωμένη ακρίβεια ώστε να επιτρέπουν στο παιχνίδι να αντιδράει σε αποδεκτό ρυθμό για την εμπειρία χρήσης.

3.4.2 Παιχνίδια

Γενικά

Στα περισσότερα παιχνίδια, η ταχύτητα των επεξεργασιών και η εμπειρία χρήσης είναι πιο σημαντικά από την ακρίβεια της εξομοίωσης. Αυτό μας οδηγεί σε σχεδιασμούς μηχανών φυσικής που παράγουν αποτελέσματα σε πραγματικό χρόνο αλλά αντιγράφουν φυσικά φαινόμενα μόνο για απλές περιπτώσεις. Τις περισσότερες φορές, η εξομοίωση είναι σχεδιασμένη να παρέχει μια φαινομενικά σωστή εκτίμηση, παρά απόλυτη ακρίβεια. Όμως μερικές μηχανές, απαιτούν μεγαλύτερη ακρίβεια σε σκληρές μάχης ή σε παιχνίδια τύπου παζλ. Οι κινήσεις χαρακτηρίων στο παρελθόν χρησιμοποιούσαν φυσική άκαμπτων σωμάτων γιατί είναι γρηγορότερο και πιο εύκολο να υπολογιστεί, όμως τα τελευταία χρόνια τα παιχνίδια και οι ταινίες έχουν αρχίσει να χρησιμοποιούν φυσική μαλακών σωμάτων. Αυτού του τύπου η εξομοίωση χρησιμοποιείται επίσης για εφέ σωματιδίων, κίνηση υγρών και υφασμάτων. Μια μορφή εξομοίωσης δυναμικής υγρών χρησιμοποιείται για να εξομοιώσει νερό και άλλα υγρά αλλά και την ροή της φωτιάς και του καπνού στον αέρα.



Σχήμα 3.4: Λογότυπο της μηχανή φυσικής havok

Ανίχνευση σύγκρουσης

Η ανίχνευση σύγκρουσης συνήθως αναφέρεται στο υπολογιστικό πρόβλημα της ανίχνευσης της διασταύρωσης δύο η περισσότερων αντικειμένων. Αν και το θέμα έχει σχέση περισσότερο με την χρήση του στα παιχνίδια και

σε άλλες εξομοιώσεις φυσικής, έχει και χρήσεις στην ρομποτική. Εκτός από την ανίχνευση του αν δύο αντικείμενα έχουν συγκρουστεί, τα συστήματα ανίχνευσης μπορούν να υπολογίσουν τον χρόνο της σύγκρουσης (Time Of Impact), και να αναφέρουν ένα σύνολο από σημεία διασταύρωσης. Η αντίδραση σύγκρουσης είναι η εξομοίωση του τι συμβαίνει όταν ανιχνευθεί μια σύγκρουση.

3.4.3 Υλοποιήσεις

Μονάδα Επεξεργασίας Φυσικής

Η μονάδα επεξεργασίας φυσικής (PPU) είναι ένας μικροεπεξεργαστής αποκλειστικά σχεδιασμένος για να χειρίζεται τους υπολογισμούς φυσικής, ειδικά σε μηχανές φυσικής των παιχνιδιών υπολογιστών. Η ιδέα είναι ότι αυτοί οι ειδικοί επεξεργαστές ελαφρύνουν το φόρτο εργασίας των CPUs, όπως μια κάρτα γραφικών εκτελεί υπολογισμούς γραφικών. Ο όρος αρχικά δημιουργήθηκε από την εταιρία Ageia για να περιγράψει τους επεξεργαστές PhysX στους καταναλωτές. Η NVIDIA απέκτησε την Ageia Technologies το 2008 και συνεχίζει να αναπτύσσει την πλατφόρμα PhysX και στο υλικό αλλά και στο λογισμικό. Από την έκδοση 2.8.3, η υποστήριξη για κάρτες PPU σταμάτησε, και δεν κατασκευάζονται πλέον.

Υπολογισμοί γενικής χρήσης σε GPUs

Η επιτάχυνση υλικού για υπολογισμούς φυσικής χρησιμοποιείται πλέον από τις GPUs που υποστηρίζουν υπολογισμό γενικής χρήσης. Η εκτέλεση φυσικών υπολογισμών σε GPUs είναι συνήθως αρκετά πιο γρήγορη από ότι σε μια CPU, έτσι η απόδοση των παιχνιδιών βελτιώνεται και η ροή εικόνας μπορεί να είναι πολύ πιο γρήγορη. Όμως η χρήση υπολογισμών φυσικής σε ένα παιχνίδι δημιουργεί επιπλέον φόρτο στην GPU. Έτσι, η χρήση ξεχωριστής μονάδας επεξεργασίας γραφικών για εκτελέσεις υπολογισμών φυσικής μπορεί να αποδώσει τα βέλτιστα αποτελέσματα. Το PhysX εκτελείται γρήγορα και αποδίδει μεγαλύτερο ρεαλισμό όταν εκτελείται στην GPU, αποφέροντας 10-20 φορές περισσότερα εφέ και οπτική πιστότητα από ότι οι υπολογισμοί φυσικής που εκτελούνται σε μια κεντρική μονάδα επεξεργασίας τελευταίας τεχνολογίας. Το PhysX χρησιμοποιεί ετερογενή υπολογισμό για να αποδώσει την καλύτερη εμπειρία χρήσης. Καθώς το παιχνίδι εκτελείται, το σύστημα PhysX εκτελεί μέρη της τεχνολογίας στην CPU αλλά και άλλα μέρη στην GPU. Αυτό γίνεται ώστε να χρησιμοποιείται αποδοτικά το υλικό του υπολογιστή ώστε να παρέχουν την καλύτερη δυνατή εμπειρία στον χρήστη. Το πιο σημαντικό, είναι ότι η τεχνολογία PhysX μπορεί να κλιμακώνεται με την χρήση GPU, σε αντίθεση με άλλες ανταγωνιστικές υλοποιήσεις φυσικής.

Σύγκριση

Οι πιο σημαντικές μηχανές γραφικών που χρησιμοποιούνται σήμερα είναι οι παρακάτω:

- PhysX - Είναι μια μηχανή φυσικής πραγματικού χρόνου, από την NVIDIA.

Είναι κλειστού κώδικα και χρησιμοποιείται σε πολλά παιχνίδια υπολογιστών και κονσολών. Υποστηρίζει μεγάλο αριθμό συσκευών.

- Havok - Είναι μια μηχανή φυσικής που χρησιμοποιείται σε πολλά παιχνίδια υπολογιστών.
- ODE - Είναι μια μηχανή φυσικής που υποστηρίζει κυρίως ανίχνευση συγκρούσεων, δυναμικές άκαμπτων σωμάτων. Αποτελεί ανοιχτό και ελεύθερο λογισμικό. Έχει χρησιμοποιηθεί σε πολλά παιχνίδια και εφαρμογές. Αποτελεί δημοφιλής επιλογή για εφαρμογές εξομίωσης ρομποτικής.
- Newton Game Dynamics - Είναι μια μηχανή φυσικής ανοιχτού κώδικα που εξομιάνει άκαμπτα σώματα σε παιχνίδια και άλλες εφαρμογές πραγματικού χρόνου.
- Bullet - Είναι μια μηχανή φυσικής που εξομιάνει ανίχνευση συγκρούσεων, δυναμική άκαμπτων και μαλακών σωμάτων. Χρησιμοποιείται σε παιχνίδια υπολογιστών αλλά και για οπτικά εφέ σε ταινίες. Η βιβλιοθήκη της bullet physics είναι ελεύθερη και ανοιχτού κώδικα κάτω από την άδεια zlib.

3.5 Μετατροπή

3.5.1 Εισαγωγή

Η μετατροπή είναι η άμεση μετατροπή ψηφιακής κωδικοποίησης, όπως για παράδειγμα σε αρχεία μιας ταινίας(π.χ .mp4, avi), αρχεία ήχου(.wav .mp3), ή κωδικοποίηση χαρακτήρων. (π.χ UTF-8, ISO/IEC 8859). Αυτή γίνεται σε περιπτώσεις που η συσκευή στόχος δεν υποστηρίζει την μορφή του αρχείου ή χρειαζόμαστε μικρότερο μέγεθος αρχείου, ή για να μετατρέψουμε αρχεία παλαιότερης μορφής σε μια πιο σύγχρονη για καλύτερη υποστήριξη σε μελλοντικές εφαρμογές.

Η μετατροπή χρησιμοποιείται συχνά στα λογισμικά προβολής βίντεο για να ελαττώσουμε το μέγεθος του αρχείου βίντεο. Μια διαδικασία που γίνεται συχνά είναι η μετατροπή από αρχεία MPEG-2(DVD) σε αρχεία μορφής MPEG-4, που ενσωματώνει σύγχρονους αλγόριθμους για καλύτερη ποιότητα εικόνας σε συνδυασμό με μικρότερο μέγεθος αρχείου.



Σχήμα 3.5: H264, το πιο δημοφιλές πρότυπο συμπίεσης βίντεο

3.5.2 Μειονεκτήματα

Το μεγαλύτερο μειονέκτημα της μετατροπής σε απωλεστικές μορφές αρχείου είναι η μειωμένη ποιότητα. Τα τεχνουργήματα συμπίεσης συσσωρεύο-

νται, οπότε κάθε διαδικασία μετατροπής δημιουργεί μια βαθμιαία απώλεια ποιότητας, που είναι γνωστή ως ψηφιακή απώλεια. Για αυτόν τον λόγο, η μετατροπή συνήθως δεν συνίσταται εκτός και αν δεν μπορούμε να την αποφύγουμε.

3.5.3 Βίντεο

Η συμπίεση βίντεο χρησιμοποιεί σύγχρονες τεχνικές για να μειώσει πλεονασμούς στα δεδομένα βίντεο. Οι περισσότεροι αλγόριθμοι συμπίεσης βίντεο συνδυάζουν συμπίεση εικόνας και προσωρινή αποζημίωση κίνησης. Ο ήχος κωδικοποιείται παράλληλα με διαφορετικούς αλγόριθμους συμπίεσης άλλα συνήθως συνδυάζεται σε ένα πακέτο όταν ολοκληρωθεί η διαδικασία.

Οι περισσότεροι αλγόριθμοι συμπίεσης βίντεο χρησιμοποιούν απωλεστική συμπίεση, καθώς το ασυμπίεστο βίντεο απαιτεί πολύ μεγάλους ρυθμούς μετάδοσης. Αν και οι περισσότεροι αλγόριθμοι έχουν ένα παράγοντα συμπίεσης 3, μια τυπική συμπίεση βίντεο MPEG-4 μπορεί να έχει παράγοντα συμπίεσης από 20 έως και 200. Όπως σε όλες τις απωλεστικές διαδικασίες συμπίεσης, υπάρχει ένα δίλημμα μεταξύ της ποιότητας του βίντεο, το κόστος της επεξεργασίας της συμπίεσης και της αποσυμπίεσης, και των απαιτήσεων του συστήματος. Υπερβολικά συμπιεσμένο βίντεο μπορεί να δημιουργήσει οπτικά τεχνουργήματα.

Συνήθως ο τρόπος με τον οποίο λειτουργεί ένας αλγόριθμος συμπίεσης βίντεο είναι με ομάδες γειτονικών εικονοστοιχείων(pixel), που ονομάζονται blocks. Αυτές οι ομάδες από pixels, συγκρίνονται μεταξύ τους από μια εικόνα στην επόμενη, και ο αλγόριθμος αποστέλλει μόνο τις αλλαγές μεταξύ αυτών των block. Σε περιοχές του βίντεο που υπάρχει μεγαλύτερη κίνηση, ο αλγόριθμος πρέπει να συμπίεσει περισσότερα δεδομένα για να προλάβει τον μεγαλύτερο αριθμό εικονοστοιχείων που αλλάζουν. Συνήθως σε σκηνές με φωτιά, εκρήξεις, καπνούς, το αποτέλεσμα της συμπίεσης έχει μεγαλύτερη απώλεια ποιότητας, η αύξηση του ρυθμού μετάδοσης.

Μερικοί από τους σύγχρονους αλγόριθμους μετατροπής είναι οι παρακάτω

- Lagarith - Είναι ένας μη απωλεστικός αλγόριθμος ανοιχτού κώδικα που δίνει έμφαση στην ταχύτητα, στην υποστήριξη διαφόρων χώρου χρωμάτων(YV12,RGB,YUY2). Είναι ιδανικός για επεξεργασία και αποθήκευση αρχείων. Αν και υπάρχουν κάποιοι καλύτεροι μη απωλεστικοί αλγόριθμοι συμπίεσης, ο lagarith είναι ο πιο γρήγορος και έτσι έχει κερδίσει την υποστήριξη της κοινότητας.
- VP9 - Είναι ένα ελεύθερο πρότυπο ανοιχτού κώδικα που αναπτύσσεται από την Google. Είναι ο διάδοχος του VP8. Σκοπός του είναι να μειώσει περισσότερο τον χώρο που απαιτείται από το βίντεο διατηρώντας την ίδια ποιότητα.
- H.264 - Είναι ένα πρότυπο συμπίεσης βίντεο, που είναι ίσως το πιο επιτυχημένο για την καταγραφή, συμπίεση, και αναμετάδοση περιεχομένου βίντεο. Χρησιμοποιείται για αναμετάδοση ψηφιακού σήματος τηλεόρασης, δορυφόρων, και από υπηρεσίες αναμετάδοσης στο διαδίκτυο.
- H.265 - Είναι ένα πρότυπο συμπίεσης βίντεο, διάδοχος του επιτυχημένου H.264. Σκοπός του είναι ο διπλασιασμός της συμπίεσης διατηρώντας την

ίδια ποιότητα. Υποστηρίζει αναλύσεις έως και 8192x4320.

- Daala - Είναι μια τεχνολογία συμπίεσης από το ίδρυμα Xiph.Org. Χρησιμοποιεί περιτύλιξη μεταμορφώσεων για να μειώσει τα οπτικά τεχνουργήματα. Ο στόχος είναι η απόδοση του να ξεπεράσει τις δυνατότητες του VP9 και του H.265.

3.5.4 Συμπίεση δεδομένων γενετικής

Οι γενετικοί αλγόριθμοι συμπίεσης είναι η τελευταία γενιά μη απωλεστικών αλγορίθμων για συμπίεση δεδομένων (συνήθως αλληλουχίες nucleotides) χρησιμοποιώντας συμβατικούς αλγορίθμους συμπίεσης αλλά και γενετικούς αλγόριθμους βελτιστοποιημένους στο συγκεκριμένο τύπο δεδομένων. Το 2012, μια ομάδα επιστημόνων από το John Hopkins University ανακοίνωσε έναν αλγόριθμο συμπίεσης γενετικής (HapZipper), ο οποίος χρησιμοποιεί HapMap δεδομένα για να πετύχει συμπίεση 95% μείωση στο μέγεθος αρχείου, πετυχαίνοντας καλύτερη συμπίεση σε πολύ καλύτερο χρόνο από ότι οι γνωστοί μη απωλεστικοί αλγόριθμοι. Άλλοι γενετικοί αλγόριθμοι συμπίεσης π.χ GenomeZip πετυχαίνουν μεγαλύτερη συμπίεση καταφέροντας αποθήκευση 6 δισεκατομμυρίων ανθρώπινων ζευγαριών γονιδιώματος σε 2.5 megabyte.

3.6 Γεωπληροφορική

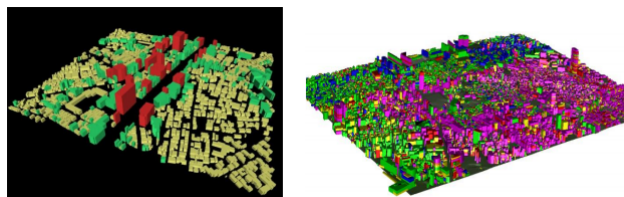
3.6.1 Γεωγραφικά πληροφοριακά συστήματα

3.6.2 Ωκεανοί

3.6.3 Σεισμική δραστηριότητα

Εισαγωγή

Η ενσωματωμένη εξομοίωση σεισμικής δραστηριότητας είναι μια ανώτερη τεχνολογία για προσδιορισμό και απεικόνιση της δομικής βλάβης που δημιουργείται σε ένα σενάριο σεισμικής δραστηριότητας. Στο IES, όλα τα κτήρια που βρίσκονται στην περιοχή δοκιμής απεικονίζονται σαν δομικά μοντέλα, και η δομική ζημιά μπορεί να προσδιοριστεί με γραμμική και μη-γραμμική δυναμική δομική ανάλυση. Επειδή υπάρχουν πολλά κτήρια σε μια περιοχή δοκιμής, και κάθε κτήριο έχει διαφορετικά χαρακτηριστικά, ο υπολογισμός υψηλής ανάλυσης είναι απαραίτητος ώστε να εκτελεστεί η ανάλυση σε σχετικά μικρό χρόνο.



Σχήμα 3.6: Ενσωματωμένη εξομοίωση σεισμικής δραστηριότητας

Ένας καλός υποψήφιος για τον IES, είναι το OBASAN (Object Based Structural Analysis). Σε μια εξομοίωση σεισμικής δραστηριότητας, κάθε κτήριο σε μια περιοχή δοκιμών μπορεί να παρασταθεί ως δομικό μοντέλο για ανάλυση του κάθε αντικειμένου. Το γεωγραφικό πληροφοριακό σύστημα παρέχει δεδομένα δομικών σχημάτων στην μορφή των στοιχείων και πολυγώνων, και άλλα συστήματα βάσεων δεδομένων παρέχουν τύπους κτηρίων (π.χ δομή οπλισμένου σκυροδέματος, δομή χάλυβα, ξύλινη δομή, κ.α) Βασισμένο στον τρόπο σχεδιασμού της Ιαπωνίας, η ανάλυση μοντέλων των κτηρίων στον IES δημιουργούνται αυτόματα από τις διαστάσεις και τον τύπο του κάθε κτηρίου, και τότε το OBASAN μπορεί να χρησιμοποιηθεί για να αναλύσει αυτά τα μοντέλα δοκιμής ανάλυσης. Η δοκιμή ζημιά σε όλα τα κτήρια μιας περιοχής δοκιμής μπορεί να προβλεφθεί και η ολική ζημιά σε μια περιοχή πόλης μπορεί να απεικονιστεί από αυτήν την εξομοίωση σεισμού.

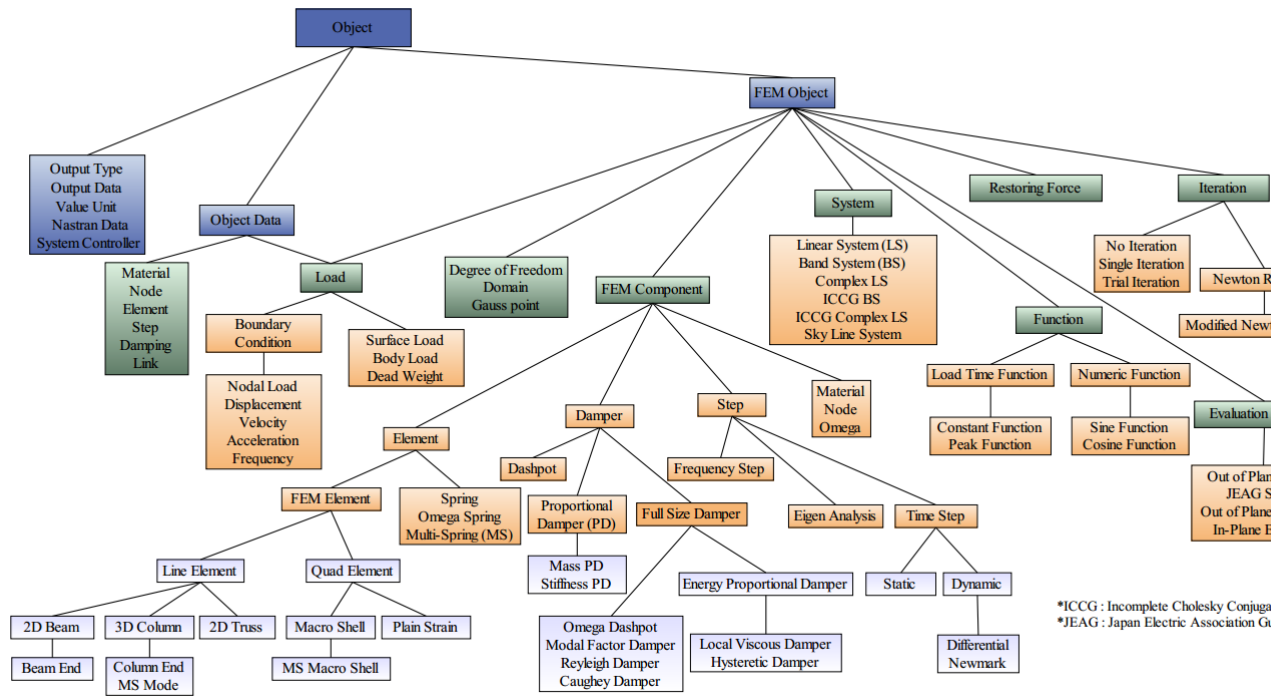
Τα πιο πολύπλοκα δομικά μοντέλα μπορούν να αυξήσουν την αξιοπιστία του προσδιορισμού δομικής βλάβης, π.χ τα μοντέλα μπορούν να χρησιμοποιηθούν για δυναμική ανάλυση δομικής απόκρισης. Μοντέλα δομικών αντικειμένων, μπορούν να χρησιμοποιηθούν για να δώσουν ένα πιο αξιόπιστο αποτέλεσμα.

OBASAN

Το OBASAN, είναι μια τεχνολογία που δημιουργήθηκε για την ανάλυση των δομικών μοντέλων ώστε να δέχεται τα αποτελέσματα των αντικειμένων (αξονική δύναμη, παραμόρφωση, καταπόνηση) και τα αποτελέσματα των κόμβων (π.χ ταχύτητα, επιτάχυνση, ισχύς) κάθε κτηρίου σε μια εξομοίωση σεισμικής δραστηριότητας, και έτσι το OBASAN μπορεί να παρέχει διάφορους τύπους αποτελεσμάτων δομικής ανάλυσης. Υπάρχουν δυνατότητες για χρήση υπολογισμού υψηλής απόδοσης ώστε να αυξηθεί η αποδοτικότητα και η δυνατότητα του συστήματος εξομοίωσης σεισμών (IES).

Το OBASAN αναπτύχθηκε με αντικειμενοστραφή προγραμματισμό σε γλώσσα C++. Η γλώσσα C++ μπορεί να επεκταθεί ώστε να χρησιμοποιεί παράλληλο προγραμματισμό στην GPU. Όπως φαίνεται στο διάγραμμα, η αρχιτεκτονική του OBASAN έχει οργανωθεί συστηματικά και είναι εύκολο να κατανοηθεί. Λόγω του ότι υπάρχουν πολλά κτήρια σε μια εξομοίωση σεισμικής δραστηριότητας σε μια περιοχή δοκιμών, ο IES χρησιμοποιεί την αρχιτεκτονική OpenMPI για να ενεργοποιήσει παράλληλο υπολογισμό στην κεντρική μονάδα επεξεργασίας (CPU). Στην CPU, ένα υποσύνολο (π.χ 10) κτηρίων μπορεί να εκτελεστεί με τον μέγιστο αριθμό νημάτων σε έναν παράλληλο υπολογισμό. Όπως και στον IES όπου η αρχιτεκτονική OpenMPI χρησιμοποιείται για τον παράλληλο υπολογισμό των κτηρίων σε μια περιοχή δοκιμών, στο OBASAN χρησιμοποιείται η αρχιτεκτονική CUDA για την παράλληλη επεξεργασία της δυναμικής ανάλυσης δομικής απόκρισης (DSRA).

Λόγω του ότι η DSRA έχει να κάνει με πιο πολύπλοκα μοντέλα, η δομική ανάλυση των μοντέλων πρέπει να παρέχει μεγάλο αριθμό παραμέτρων, που εκφράζονται με μαθηματικές εξισώσεις που περιέχουν μεγάλους πίνακες. Για ένα τόσο προχωρημένο υπολογιστικό έργο, η τεχνική HPC είναι η συνηθέστερη προσέγγιση για την επίλυση των μαθηματικών εξισώσεων σε μικρό χρόνο. Στο HPC, η τεχνολογία GPGPU είναι σχεδιασμένη ώστε να



Σχήμα 3.7: OBASAN

αντιμετωπίζει μεγάλο αριθμό δεδομένων και μεγάλο αριθμό υπολογισμών σε μαθηματικές εξισώσεις. Στο GPGPU, η αρχιτεκτονική CUDA μπορεί να επιλεχτεί για να λύσει αυτές τις μαθηματικές εξισώσεις της DSR. Το κάθε βήμα της ανάλυσης της DSR, απαιτεί πολλαπλασιασμούς πινάκων για την δομική ανάλυση.

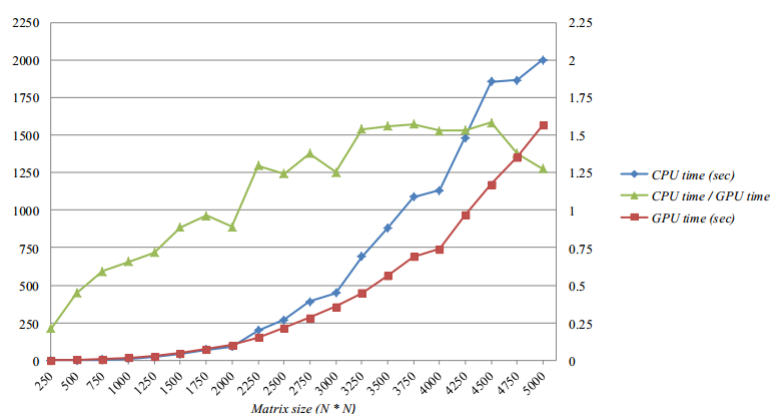
Στο παρακάτω σχήμα μπορούμε να δούμε ένα παράδειγμα χρόνου εκτέλεσης πολλαπλασιασμού πινάκων. Οι πίνακες A,B και C υποθέτουμε ότι είναι τετράγωνοι πίνακες και το μέγεθος τους είναι $N * N$. Ο κώδικας C++ του πολλαπλασιασμού πινάκων εκτελείται σε μια CPU (Intel Xeon), ενώ οι εντολές CUDA σε μια GPU(Tesla M2050).

Matrix size ($N * N$)	CPU time (sec)	GPU time (sec)	Matrix size ($N * N$)	CPU time (sec)	GPU time (sec)
250 * 250	0.1343	0.0006	2750 * 2750	389.9368	0.2829
500 * 500	1.2111	0.0027	3000 * 3000	447.8122	0.3573
750 * 750	4.2088	0.0071	3250 * 3250	692.1258	0.4498
1000 * 1000	10.3524	0.0157	3500 * 3500	878.4347	0.5626
1250 * 1250	20.8688	0.0290	3750 * 3750	1086.8954	0.6909
1500 * 1500	41.5084	0.0468	4000 * 4000	1133.5968	0.7412
1750 * 1750	71.0964	0.0737	4250 * 4250	1484.2185	0.9687
2000 * 2000	91.2003	0.1026	4500 * 4500	1855.4903	1.1724
2250 * 2250	198.0169	0.1525	4750 * 4750	1866.9938	1.3547
2500 * 2500	269.6148	0.2171	5000 * 5000	2000.5736	1.5680

Σχήμα 3.8: GPU and CPU matrix multiplication times

Στην DRSA, το μέγεθος πίνακα ορίζεται απο τον συνολικό αριθμό των μοιρών της ελευθερίας (DOF) σε ένα κτήριο, οπότε το μέγεθος του πίνακα εξαρτάται απο τις διαστάσεις του κάθε κτηρίου. Τα μικρότερα κτήρια όπως τα σπίτια αναπαρίστανται απο μικρότερους πίνακες και τα μεγαλύτερα κτήρια αναπαρίστανται απο μεγαλύτερους πίνακες. Όπως μπορούμε να δούμε στον πίνακα, οι GPUs είναι πολύ πιο γρήγορες απο τις CPUs σε

κάθε περίπτωση πολλαπλασιασμού πινάκων. Από τα δεδομένα του πίνακα, μπορούμε να απεικονίσουμε τα αποτελέσματα σε γράφημα. Στο παρακάτω γράφημα βλέπουμε την σχέση που έχει ο χρόνος εκτέλεσης με το μέγεθος του πίνακα, τόσο στην CPU όσο και στην GPU. Οι μεγαλύτεροι πίνακες χρειάζονται περισσότερο χρόνο για να εκτελέσουν τον υπολογισμό πινάκων. Όσο μεγαλύτερο το μέγεθος του πίνακα, τόσο μεγαλύτερο το πλεονέκτημα της GPU έναντι της CPU. Για ένα πίνακα $2000 * 2000$ στοιχείων, οι GPUs είναι περίπου 1000 φορές γρηγορότερες από τις CPUs. Για αυτό ο προγραμματισμός γενικού σκοπού είναι καλή επιλογή για την δυναμική ανάλυση δομικής απόκρισης.



Σχήμα 3.9: GPU and CPU matrix multiplication times

Τα παραπάνω αποτελέσματα, είναι μια απλή τεχνική CUDA. Ο προγραμματισμός υψηλής απόδοσης της DSRA μπορεί να αυξηθεί με την χρήση βιβλιοθηκών CUDA. Οι CUDA βιβλιοθήκες, π.χ CUBLAS(Γραμμική Άλγεβρα), CUFFT(Μετασχηματισμός Fast Fourier), μπορούν να λύσουν τις μαθηματικές εξισώσεις της DSRA. Γενικότερα, όλες οι βιβλιοθήκες CUDA είναι βελτιστοποιημένες για εκτέλεση υψηλής απόδοσης. Επιπλέον, το OBASAN με την χρήση CUDA μπορεί να επιτύχει καλή κλιμάκωση στον IES.

Συμπεράσματα

Για να μειώσουν την κοινωνική και οικονομική επίπτωση των σεισμών, οι ερευνητές προσπαθούν να εφαρμόσουν καινούριες τεχνολογίες και τα τελευταία δεδομένα στον IES. Η αριθμητική εξομοίωση χρησιμοποιείται για να προσδιορίσει πιθανές ζημιές σε σενάρια σεισμών. Σε μια εξομοίωση σεισμών, η σωστή πρόβλεψη της ζημιάς στην κοινωνία είναι πολύ σημαντική στην διαχείριση κρίσεων. Αυτή η προσέγγιση στο μαθηματικό πρόβλημα με την χρήση τεχνολογιών προγραμματισμού υψηλής απόδοσης και CUDA, μπορεί να εφαρμοστεί σε διάφορα πεδία της μηχανικής.

Με την χρήση των προσδιορισμών, ένα πιο σταθερό πλάνο μπορεί να χρησιμοποιηθεί για κατασκευή καινούριων κτηρίων για να εμποδίσει δομικές ζημιές σε σενάρια σεισμών. Επιπλέον, σχέδια για εκκένωση κτηρίων και μεταφορά ατόμων μπορούν να δημιουργηθούν για να αναλύσουν τις εξόδους κινδύνου, κάτι που μπορεί να προστατέψει και να σώσει ανθρώπινες ζωές.

3.7 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 3.10: Αστροφυσική

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

Προσδιορισμός του προβλήματος και διεθνής εμπειρία

Μεθοδολογία και Εφαρμογή

5.1 Εργαλεία που χρησιμοποιήθηκαν

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.

5.1.1 X_YTeX

Το X_YTeX 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 X_YTeX 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 ή ένα στρογγυλό γράμμα όπως το "ο" (protrusion), και ο χωρισμός γραμμών (hyphenation). Αν και το X_YTeX είναι υποδεέστερο του L^ATeX 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 X_YTeX

5.1.2 Texmaker

Το Texmaker είναι ένα επεξεργαστής κειμένου για L^ATeX και X_YL^ATeX. Είναι cross-platform ανοιχτού κώδικα με ενσωματωμένο PDF Viewer. Το Texmaker είναι μια εξ ολοκλήρου 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 L^AT_EX και μαθηματικά σύμβολα μπορούν να χρησιμοποιηθούν με ένα click. Ο Texmaker βρίσκει αυτόματα τα λάθη και προειδοποιήσεις και τα εμφανίζει στο αρχείο καταγραφής.

5.1.3 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, αλλά ο έλεγχος αναθεωρήσεων βρίσκεται ενσωματωμένος και μέσα σε διάφορους τύπους λογισμικού όπως οι επεξεργαστές κειμένου, αλλά 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.

5.1.4 Github

Το Github είναι μια διαδικτυακή υπηρεσία αποθηκευτικού χώρου Git, που επιτρέπει πλήρη κατανεμημένο έλεγχο 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 5.1: Λογότυπο Github

5.2 Άδεια χρήσης

5.2.1 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 ή 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 αντιγράφει, 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 χρήσης λογισμικού εντάσσονται σε μια από τις παρακάτω 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 ή 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 5.2: 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 εμμένει 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 5.3: Λογότυπο Free Software Foundation

5.2.2 Επιλογή


Μετά από εκτενή αναζήτηση και έρευνα πάνω στις άδειες χρήσεις λογισμικού, 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 από 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, προσαρμοσης, 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 L^AT_EX, αλλά ταυτόχρονα είναι και συμβατή με την άδεια χρήσης του πηγαίου κώδικα.

Το περιεχόμενο του παρόντος κειμένου υπάγεται σε Άδεια Χρήσης Creative Commons Attribution 4.0 <http://creativecommons.org/licenses/by-sa/4.0/> dsadsa dsadsa

Βιβλιογραφία

- [1] Open Source Initiative. *The Open Source Definition (Annotated)*. Online; accessed 7-September-2014. 2014. URL: <http://opensource.org/docs/osd> (see p. 41).
- [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>.
- [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.