Παράλληλα Συστήματα

Εργασία 4 - CUDA, 2024

Γιάννης Ζερβάκης

A.M.: 1115201800048

Περιγραφή εργασίας

Στην εργασία αυτή, κληθήκαμε να παρακολουθήσουμε και να ολοκληρώσουμε ένα course της NVIDIA, του οποίου στόχος είναι η εξοικείωση με τον παράλληλο προγραματισμό σε C/C++ και CUDA. Το συγκεκριμένο σεμινάριο, χωρίζεται σε τρείς ενότητες. Η πρώτη αφορά τις βασικές παράλληλες λειτουργίες χρησιμοποιώντας CUDA : την έννοια του πυρήνα, των `grid`, `blocks` και `threads`, τον συγχρονισμό τους, τις ρυθμίσεις εκτέλεσης (`execution configuration`), διαχείριση μνήμης κ.α. Βάσει διαφόρων παραδειγμάτων, επιτεύθει κι η εκμάθηση των βασικών λειτουργιών. Στην δεύτερη ενότητα εξοικειωθήκαμε με το εργαλείο `nsys` το οποίο μας βοηθάει να απεικονίσουμε διάφορες μετρήσεις που έγιναν σε σχέση με τους χρόνους και τη μνήμη του προγράμματος, τόσο σε επίπεδο CPU όσο και σε GPU ενώ μέσω παραδειγμάτων εντρυφήσαμε πιο πολύ στο κομμάτι της μνήμης, αφού μελετήσαμε το πρόβλημα του page faulting και πως αυτό μπορούμε να το περιορίσουμε. Τέλος, στην τρίτη ενότητα, κληθήκαμε να χρησιμοποιήσουμε εργαλεία που σκοπό έχουν την οπτική απεικόνιση των μετρήσεων και κατάστασης του προγράμματος ενώ ασχοληθήκαμε και με το κομμάτι των streams. Καθ'όλο το σεμινάριο υπήρχαν παραδείγματα και ασκήσεις που στόχο είχαν την καλύτερη εκμάθηση των προαναφερθέντων εννοιών, ενώ στην τρίτη και τελευταία ενότητα μας ζητήθηκε να επιταγχύνουμε μια ήδη έτοιμη εφαρμογή που αφορά μια προσωμοίωση Ν σωματιδίων που προσωμοιώνει ένα δυναμικό σύστημα σωματιδιών.

Περιγραφή του προβλήματος

Στόχος του προβλήματος που ζητήθηκε να επιλύσουμε στα πλαίσια του course, ήταν η επιτάγχυνση ενός N-body προσωμοιωτή με τις γνώσεις και δεξιότητες που αναπτύξαμε κατά την παρακολούθησή του. Έτσι λοιπόν, μας δόθηκε μια απλά CPU-only εκδοχή της εφαρμογής, η οποία χρειαζόταν περίπου 10 δευτερόλεπτα να εκτελεσθεί (σύμφωνα με το test/assessment του περιβάλοντος της nvidia, στο οποίο τρέχει και το τελικό πρόγραμμα). Στον κώδικα αυτόν δεν υπάρχε παραλληλοποίηση.

Προσέγγιση

Όπως προαναφέρθηκε, το πρόγραμμα που δόθηκε να μετατρέψουμε και να επιταγχύνουμε ήταν το `nbody_slow.cu` που βρίσκεται στον κατάλογο αυτό, κι οι αλλαγές που έγιναν είναι λίγες μα στοχευμένες. Αρχικά, μετατρέπουμε την βασική συνάρτηση που υπολογίζει την βαρυτική επίδραση όλων των σωματιδίων πάνω σε όλα τα σωματίδια στο σύσημα σε πυρήνα, έτσι ώστε να είναι αυτή που θα εκτελεστεί παράλληλα. Αυτό έγινε με την προσθήκη στο πρωτότυπό της, της λέξης κλειδί `__global__`. Στη συνέχεια, στόχος μας είναι να κρατήσουμε την ίδια λειτουργία της συνάρτησης αυτής, αλλά πλέον να υπολογίζει τιμές όχι για όλα τα n-σωματίδια, αλλά κάθε πυρήνας να υπολογίζει μια ομάδα σωματιδίων κυκλικά ανάλογα με το πόσα blocks και threads θα εκτελεστεί. Έτσι λοιπόν, υπολογίζουμε αρχικά τις συντεταγμένες του thread που βρισκόμαστε `idx = blockIdx.x * blockDim.x + threadIdx.x` ενώ υπολογίζουμε και με τι

δρασκελισμό πρέπει ο πυρήνας που βρισκόμαστε να κινείται, έτσι να ισομοιραστεί το πλήθος των τιμών που πρέπει να υπολογιστούν, `stride = blockDim.x * gridDimx`. Η ουσιαστική λοιπόν μετατροπή είναι πως υπολογίζουμε τις τιμές μας στον πυρήνα που βρισκόμαστε από το idx του μέχρι και να εξαντλήσουμε το συνολικό πλήθος με βήμα το stride.

Αρκεί λοιπόν πλέον, να εκκινήσουμε τον πυρήνα που δημιουργήσαμε με σωστές ρυθμίσεις εκτέλεσης (`execution configuration`) ώστε να έχουμε το επιθυμητό αποτέλεσμα. Πριν όμως φάσουμε να αναφερθούμε σε αυτές, αλλάξαμε τον τρόπο δέσμευσης της μνήμης έτσι ώστε να είναι συμβατός και με τον πυρήνα. Δηλαδή η κλήση `malloc()` αντικαταστάθηκε με την `cudaMallocManaged()` (αντίστοιχα, χρησιμοποιήθηκε και η κλήση αποδέσμευσης `cudaFree()`). Στο σημείο αυτό, και δεδομένου ότι γνωρίζουμε εκ των προτέρων πως την μνήμη αυτή θα τη χρησιμοποιήσει η GPU και ο ο πυρήνας που δημιουργήσαμε, καλέσαμε την μέθοδο `cudaMemPrefetchAsync()` η οποία μεταφέρει ασύγχρονα ομάδες της μνήμης από την CPU στην GPU με σκοπό την αποφυγή του page faulting που έχει ως συνέπεια την επιτακτική μετανάστευση των δεδομένων. Με αυτόν τον τρόπο, κερδίσαμε τον χρόνο που θα χανόταν σε άλλη περίπτωση.

Στη συνέχεια, έρχεται η ώρα της επιλογής των ρυθμίσεων εκτέλεσης, δηλαδή την επιλογή του αριθμού των threads και blocks. Για αριθμό των blocks επιλέξαμε το 256, αφ'ενός γιατί είναι δύναμη του δύο, αφ'ετέρου γιατί είναι η πιο συνηθισμένη επιλογή που γινόταν και κατά τη διάρκεια του `course` στις υπόλοιπες ασκήσεις. Όσον αφορά τον αριθμό των threads, επιλέξαμε πολλαπλάσιο του αριθμού των SMs (`Streaming Multiprocessors`) τον οποίο ανλτήσαμε με την βοήθεια των εντολών `cudaGetDevice()` και `cudaDeviceGetAttribute()` οι οποίες επιστρέφουν το αναγνωριστικό της συσκευής καθώς και ένα συγκεκριμένο χαρακτηριστικό της αντίστοιχα. Η επιλογή αυτή έγινε για να ισομοιραστεί το μερίδιο της συνολικής δουλειάς άρτια, χωρίς προκύψουν threads που να πρέπει να περιμένουν για να εκτελεστούν εν τέλει σειριακά πάνω στην ίδια υπολογιστική μονάδα.

Έτσι λοιπόν, με όλες τις παραπάνω μετατροπές, πράγματι βελτιώσαμε σημαντικά τον χρόνο εκτέλεσης της εφαρμογής και πετύχαμε τους στόχους ολοκλήρωσης εκτέλεσης σε κάτω από 0.9 δευτερόλεπτα για 4096 σώματα και κάτω από 1.3 δευτερόλεπτα για 65536. Στην παρακάτω εικόνα φαίνονται ακριβώς τα αποτελέσματα από την αξιολόγηση του course :

```
[4]: run_assessment()

Running nbody simulator with 4096 bodies
-----------------------------------------

Application should run faster than 0.9s
Your application ran in: 0.1257s
Your application reports  20.453 Billion Interactions / second

Your results are correct

Running nbody simulator with 65536 bodies
-----------------------------------------

Application should run faster than 1.3s
Your application ran in: 0.2024s
Your application reports  368.676 Billion Interactions / second

Your results are correct

Congratulations! You passed the assessment!
See instructions below to generate a certificate, and see if you can accelerate the simulator even more!
```

Εκτέλεση και αρχεία

Όλες οι εκτελέσεις γίνανε στο περιβάλλον του σεμιναρίου το οποίο προσφέρει δικό του τρόπο αξιολόγησης (assessment). Τα αρχεία που περιέχονται στον κατάλογο αυτό είναι αρχικά τα αρχεία επικεφαλίδες που δόθηκαν, `files.h` και `timer.h`, τα οποία δηλώνουν συναρτήσεις που έχουν να κάνουν με την εγγραφή και ανάγνωση των τιμών από τα αρχεία και την μέτρηση του χρόνου αντίστοιχα. Επίσης υπάρχει το αρχείο `nbody_slow.cu` που είναι το αρχείο που αρχικά μας δόθηκε, χωρίς μετατροπές και επιτάχυνση. Τέλος υπάρχει το αρχείο `nbody_accelerated.cu`, το οποίο είναι το πλέον μετατρεμένο που περιέχει όλες αλλαγές που προαναφέρθηκαν και στοχεύει στην επιτάχυνση της εφαρμογής. Τέλος υπάρχει μια εικόνα `results.png` με τα αποτελέσματα από την αξιολόγηση της επιταγχυμένης εφαρμογής, καθώς κι ο κατάλογος `certifications/` που περιέχει ένα pdf αρχείο με τα πιστοποιητικά βεβαίωσης ολοκλήρωσης του σεμιναρίου και την παρακάτω εικόνα που εμφανίζεται μετά το πέρας του σεμιναρίου επίσης :