Αναφορά εργασίας 3 : Εκτέλεση μοντέλου Ising με χρήση της GPU μέσω CUDA

ΤΗΜΜΥ ΑΠΘ Παράλληλα και Διανεμημένα Συστήματα

Νικοπολιτίδης Εμμανουήλ-Γεώργιος

ΑΕΜ:9474

Κώδικας : https://drive.google.com/drive/folders/1lz2pFBLhx5IXhvHS0RTYW0EN_IdW1NiR?usp=sharing

Εισαγωγή

Σκοπός είναι η υλοποίηση του μοντέλου Ising με διάφορες προγραμματιστικές μεθόδους. Σύμφωνα με το μοντέλο αυτό εκκινώντας από ενάν πίνακα με στοιχεία 1 και -1 υπολογίζουμε την νέα τιμή κάθε θέσης με βάση την σχέση:

$$G[i,j]=\text{sign}(G[i-1,j] + G[i,j-1] + G[i,j] + G[i+1,j] + G[i,j+1])$$

Η συγκεκριμένη διαδικάσια επαναλαμβάνεται για Κ επαναλήψεις.

Στην εργασία προσεγγίζουμε το παραπάνω μοντέλο με σειριακό τρόπο αλλά και με παράλληλο προγραμματισμό σε GPU με χρήση CUDA και συγκρίνουμε τα αποτελέσματα.

Sequential

Η απλούστερη μορφή του κώδικα όπου μέσω μιας συνάρτησης sign υπολογίζουμε σειριακά το άθροισμα του κάθε σημείου και των τεσσάρων γειτόνων του σε σχήμα σταυρού και κατόπιν αποφασίζουμε για την τιμή του σημείου στην επόμενη επανάληψη (1 αν το άθροισμα είναι θετικό, -1 αν το άθροισμα είναι αρνητικό) . Πρέπει να σημειωθεί εδώ πως ,όπως και στις επόμενες εκδοχές του κώδικα , ο πίνακας θεωρείται πως έχει μορφή τόρου και άρα το προηγούμενο του 0 (γραμμή, στήλη) είναι το N-1 και το επόμενο του N-1 το 0.

Version 1

Εδώ κάνουμε μια πρώτη προσπάθεια παραλληλοποίησης του σειριακού κώδικα αξιοποιώντας την CUDA. Χρησιμοποιούμε ένα thread από την GPU για την εκτέλεση των υπολογισμών κάθε σημείου επομένως έχουμε μεγάλο αριθμό threads (ίσο με τα στοιχεία του πίνακα).

Version 2

Αυτή την φορά κάθε thread της GPU κάνεις τους υπολογισμούς ενός b*b μέρους του μεγάλου πίνακα. Έτσι έχουμε λιγότερα threads αλλά με περισσότερη δουλειά.

Version 3

Τέλος η τρίτη εκδοχή είναι πανομοιότυπη με την δεύτερη με την διαφορά πως προτού τα ξεκινήσουμε τους απαραίτητους υπολογισμούς για το μοντέλο Ising τα threads κάθε block αντιγράφουν (μόνο)τα στοιχεία του αρχικού πίνακα που του αντιστοιχούν σε ένα πίνακα shared μνήμης ώστε οι μετέπειτα προσπελάσεις να είναι εξαιρετικά γρηγορότερες. Σημειώνεται πως με στόχο τον περιορισμό της πολυπλοκότητας στην shared μνήμη του κάθε block περνάνε μόνο τα στοιχεία του πίνακα για τα οποία θα κάνει υπολογισμούς ενώ τα στοιχεία γειτωνικών block που είναι απαραίτητα για τις πράξεις γίνονται προσβάσιμα μέσω global μνήμης . Επειδή οι προσπελάσεις της shared μνήμης είναι εξαιρετικά περισσότερες από αυτές της global η απόδοση του προγράμματος πρακτικά δεν επηρεάζεται.(Υποσημείωση 1)

Αποτελέσματα

Παρακάτω παρατίθενται πίνακας με τους χρόνους όλων των versions για διάφορες παραμέτρους(λόγω των πολλών μεταβλητών θεωρήθηκε ότι τα διαγράμματα δεν θα εξυπηρετούσαν την κατανόηση των αποτελεσμάτων).

#	N,M	K	b	B.v1	B.v2	B.v3	Tserial	Tv.1	Tv.2	Tv.3
1	60	41001	10	6	36	6	2.99	0.17	0.75	1
2	60	41001	6	10	10	10	3.08	0.15	0.39	0.43
3	100	41001	10	100	5	10	9.5	0.4	5.8	3.9
4	1000	303	5	10000	1000	200	6.28	0.021	0.048	0.042
5	1000	3031	5	10000	1000	200	63.34	0.20	0.28	0.20
6	1000	303	10	10000	1000	100	7.16	0.01	0.28	0.23
7	2000	301	10	4000	4000	4000	25.27	0.11	0.21	0.19
8	2000	301	20	4000	2000	2000	25.23	0.1	0.20	0.14
9	4000	301	4	16e4	4e4	4e4	100.84	0.29	0.33	0.39
10	4000	301	40	16e4	4e4	4e4	101.15	0.27	1.24	2.8
11	10000	31	10	1e6	1e5	1e5	65.48	0.34	0.5	0.42
12	10000	3001	10	1e6	1e5	1e5	-	13.17	35.14	36.07

*Χρόνοι σε sec.

Συμπεράσματα

- Για μικρούς πίνακες παρατηρούμε πως η εκδοχή v1 είναι εξαιρετικά γρηγορότερη καθώς κάθε thread επωμίζεται μικρό όγκο εργασίας σε αντίθεση με τις άλλες υλοποιήσεις όπου η δουλειά πολλαπλασιάζεται. Επίσης για λίγα στοιχεία φαίνεται πως η μεταφορά των δεδομένων από την global στην shared memory στην πραγματικότητα κοστίζει χρόνο(αφού το v2 τρέχει πιο γρήγορα από το v3).

- Για μεσαίους πίνακες αυξάνοντας τις διαμερίσεις b ο χρόνος φαίνεται να αυξάνεται στις 2 τελευταίες παράλληλες εκδοχές πράγμα που υποδεικνύει πως ακόμα η κλήση ενός thread ανά σημείο είναι η ενδεδειγμένη λύση.
- Για πίνακες άνω της διάστασης του 100 φαίνονται τα πραγματικά οφέλη της παραλληλοποίησης αφού ο χρόνος γίνεται πραγματικά ελάχιστος σε σχέση με τον σειριακό κώδικα.
- Εδώ (N=2000) παρατηρούμε πως αυξάνοντας το b οι χρόνοι των v3 και v2 πέφτουν και έτσι καταλαβαίνουμε πως πλέον το άνοιγα υπερβολικού αριθμού threads αρχίζει να μην είναι βιώσιμο.
- Για πίνακες των 4000 όμως η αύξηση του b οδηγεί σε αντίθετα αποτελέσματα με τους χρόνους των v2 και v3 να αυξάνονται σημαντικά.
- Τέλος για πίνακες με 100 εκατομμύρια στοιχεία η παραλληλοποίηση αποδίδει μεν εντυπωσιακά αλλά και πάλι το v1 παραμένει το τάχιστο όλων. Το συγκεκριμένο παράδοξο αναζητά εξήγηση στην υποσημείωση 2.
- Γενικότερα το v3 είναι γρηγορότερο από το v2 χάρης στην αξιοποίηση της μνήμης shared η οποία επιτρέπει προσπελάσεις των στοιχείων ακόμα και 100 φορές ταχύτερα .
- Όσο αναφορά το K βλέπουμε πως η επίδραση του είναι απλά αναλογική στα αποτελέσματα

Υποσημειώσεις

1. Στην υλοποίηση του v3 έχει γίνει ένα σκόπιμο trade-off μεταξύ της αξιοποίησης της μνήμης shared και της πολυπλοκότητας του κώδικα. Επιλέχθηκε να αντιγραφούν στην shared μόνο τα στοιχεία εκείνα που βρίσκονται σε θέσεις όπου το εκάστοτε block είναι υπεύθυνο για τον υπολογισμό του μοντέλου Ising . Τα υπόλοιπα στοιχεία σε σχήμα κορνίζας που ουσιαστικά ανήκουν σε άλλα block προσπελαύνονται μέσω της global.
2. Το σύστημα στο οποίο εκτελέστηκαν τα πειράματα διέθετε 50kBytes shared memory/block ενώ κάθε integer καταλαμβάνει 4 Bytes. Εύκολα προκύπτει πως στην shared μνήμη μπορούν να αποθηκευτούν πίνακες μέγιστων διαστάσεων 110*110. Εικάζεται ,λοιπόν, πως το μέγεθος αυτό είναι αρκετά μικρό για να μπορέσει να επιταχύνει την διαδικασία σε σχέση με το v1 όπου ναι μεν εκκινούνται πολλά threads αλλά έχουν ελάχιστο φόρτο εργασίας. Μία πιθανή λύση θα ήταν η χρήση boolean αντί για integer.