ΠΑΡΑΛΛΗΛΑ ΣΥΣΤΗΜΑΤΑ

ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
UNIVERSITY OF WEST ATTICA

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΕΡΓΑΣΙΑ 2Β.1

CUDA

ΣΤΟΙΧΕΙΑ ΦΟΙΤΗΤΗ / ΕΡΓΑΣΙΑΣ

ΟΝΟΜΑΤΕΠΩΝΥΜΟ : ΑΘΑΝΑΣΙΟΥ ΒΑΣΙΛΕΙΟΣ ΕΥΑΓΓΕΛΟΣ
ΑΡΙΘΜΟΣ ΜΗΤΡΩΟΥ : 19390005
ΕΞΑΜΗΝΟ ΦΟΙΤΗΤΗ : 11
ΠΡΟΓΡΑΜΜΑ ΣΠΟΥΔΩΝ : ΠΑΔΑ

ΥΠΕΥΘΥΝΟΣ ΕΡΓΑΣΤΗΡΙΟΥ: ΙΟΡΔΑΝΑΚΗΣ ΜΙΧΑΛΗΣ
ΥΠΕΥΘΥΝΟΣ ΘΕΩΡΙΑΣ: ΜΑΜΑΛΗΣ ΒΑΣΙΛΕΙΟΣ

ΠΑΡΑΛΛΗΛΑ ΣΥΣΤΗΜΑΤΑ

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΑΡΑΛΛΗΛΑ ΣΥΣΤΗΜΑΤΑ

1. Εισαγωγή

1.1 Σκοπός της άσκησης

Ο σκοπός της άσκησης είναι η χρήση της αρχιτεκτονικής CUDA για την επίλυση ενός προβλήματος υπολογισμών σε πίνακες, με αξιοποίηση παράλληλης επεξεργασίας.

1.2 Συνοπτική περιγραφή του προβλήματος που επιλύεται

Το πρόγραμμα δημιουργεί έναν τυχαίο 2Δ πίνακα ακέραιων αριθμών και εκτελεί τις παρακάτω πράξεις:

- Υπολογισμό μέσου όρου (calcAvg).
- Εύρεση μέγιστου στοιχείου (findMax).
- Δημιουργία πίνακα B βάσει των στοιχείων του αρχικού πίνακα (createB).
- Δημιουργία πίνακα C βάσει γειτονικών στοιχείων (createC).

2. Σχεδιασμός

2.1 Περιγραφή της προσέγγισης που ακολουθήθηκε

Η προσέγγιση περιλαμβάνει την παραλληλοποίηση βασικών λειτουργιών σε πίνακες χρησιμοποιώντας CUDA kernels. Οι υπολογισμοί κατανέμονται σε νήματα μέσω πλέγματος (grid) και μπλοκ (blocks).

2.2 Ανάλυση της λογικής και των μεθοδολογιών

Η λογική βασίζεται στα εξής:

- Χρήση παράλληλων reduction για υπολογισμό μέσου όρου και εύρεση μέγιστου.
- Χρήση shared memory για βελτίωση απόδοσης.
- Διαίρεση του πίνακα σε τμήματα για την κατανομή της δουλειάς σε threads.

2.3 Περιγραφή των δομών δεδομένων και των αλγορίθμων

2.3.1 Δομές δεδομένων και μεταβλητές

Οι βασικές δομές δεδομένων περιλαμβάνουν:

- Πίνακες εισόδου (d_A) και εξόδου (d_OutArr) στη μνήμη της συσκευής (device).
- Μεταβλητές για αποθήκευση μέσου όρου, μέγιστου και ελάχιστου στοιχείου.
- Δομές για τη δημιουργία CUDA events για μέτρηση χρόνου.

2.3.2 Πρόγραμμα γεννήτρια για παραγωγή 2Δ πινάκων

Η συνάρτηση create2DArray δημιουργεί τυχαίο πίνακα μεγέθους NxN, διασφαλίζοντας ότι το μέγιστο στοιχείο του είναι μεγαλύτερο από N·m, όπου m ο μέσος όρος.

2.3.3 calcAvg<<<>>>()

Η συνάρτηση **πυρήνα** υπολογίζει τον μέσο όρο όλων των στοιχείων του πίνακα χρησιμοποιώντας reduction και atomic εντολές.

2.3.4 findMax<<<>>>()

Η συνάρτηση **πυρήνα** εντοπίζει το μέγιστο στοιχείο του πίνακα με παράλληλο reduction και χρήση atomic εντολών.

2.3.5 createB<<<>>>()

Υπολογίζει τον πίνακα Bij=(m−Aij) / amax και βρίσκει το ελάχιστο στοιχείο του πίνακα.

2.3.6 createC<<<>>>()

Δημιουργεί τον πίνακα Cij=(Aij+Ai(j+1)+Ai(j−1)) / 3, λαμβάνοντας υπόψη τα όρια του πίνακα.

2.3.7 atomicMin

Η atomicMin είναι μια προσαρμοσμένη συνάρτηση που χρησιμοποιείται για τον υπολογισμό της ελάχιστης τιμής σε αριθμούς κινητής υποδιαστολής στη συσκευή (device). Η χρήση της είναι αναγκαία, καθώς η CUDA δεν παρέχει υποστήριξη για atomic εντολές σε αριθμούς κινητής υποδιαστολής (floats). Η υλοποίηση γίνεται με την atomicCAS

3. Υλοποίηση

3.1 Αναφορά στις βασικές λειτουργίες του κώδικα

Ο κώδικας περιλαμβάνει:

- Αρχικοποίηση δεδομένων και παραμέτρων CUDA.
- Δημιουργία και μεταφορά δεδομένων από host σε device.

- Εκτέλεση των παραπάνω CUDA kernels.
- Ανάκτηση και αποθήκευση αποτελεσμάτων.

3.2 Επεξήγηση παράλληλων τμημάτων του κώδικα

Η χρήση shared memory βελτιώνει την ταχύτητα των υπολογισμών, μειώνοντας την προσπέλαση στην παγκόσμια μνήμη.
Οι παράλληλες λειτουργίες διανέμουν το έργο σε threads και blocks.

3.3 Περιγραφή της επικοινωνίας και του συγχρονισμού μεταξύ νημάτων

Χρήση __syncthreads() για συγχρονισμό threads στο ίδιο μπλοκ.

Χρήση atomic λειτουργιών για ασφαλή τροποποίηση κοινών δεδομένων.

4. Δοκιμές και Αποτελέσματα

4.1 Αναφορά των συνθηκών εκτέλεσης

Η εκτέλεση γίνεται για διαφορετικά μεγέθη πίνακα N x N, αριθμούς νημάτων T ανά μπλοκ και μπλοκ ανά πλέγμα.

Η μεταγλώττιση του προγράμματος γίνεται μέσω command line σε περιβάλλον Linux, με τον compiler της NVIDIA **nvcc**.

```
nvcc -o cuda1 cuda1.cu
```

Η εκτέλεση του προγράμματος γίνεται μέσω command line σε περιβάλλον Linux και πρέπει ο χρήστης να περάσει παραμετρικά 2 αρχεία txt, ώστε να αποθηκευτούν αντίστοιχα ο πίνακας A και ο πίνακας B ή C. Ενδεικτική εντολή εκτέλεσης:

```
./cuda1 A.txt OutArr.txt
```

4.2 Παρουσίαση των αποτελεσμάτων σε μορφή κειμένου

Τα αποτελέσματα είναι αποθηκευμένα στον φάκελο Output και οι πίνακες A και B ή C εκάστως στους αντίστοιχους φακέλους. Για εξοικονόμηση χώρου δεν παρουσιάζονται όλα τα αποτελέσματα σε μορφή κειμένου στην παρούσα τεκμηρίωση.

Το πρόγραμμα απαιτεί από τον χρήστη να περάσει παραμετρικά 2 .txt αρχεία εξόδου με όνομα της επιλογής του, στα οποία θα αποθηκευτούν ο πίνακας A και ο B ή ο C. Σε περίπτωση που ο χρήστης δεν βάλει τον απαιτούμενο αριθμό παραμέτρων, το πρόγραμμα τερματίζεται και εμφανίζεται χαρακτηριστικό μήνυμα

4.2.1 Output_no_args.txt

```
Usage: ./cuda1 A.txt OutArr.txt
```

Για να ελέγξουμε την ορθότητα του αλγορίθμου δοκιμάσαμε για αρχή έναν πίνακα μικρού μεγέθους, ώστε να επιβεβαιώσουμε ότι επιτυγχάνεται η ταξινόμηση.

4.2.2 Output8B.txt

```
--------------- Device Properties ---------------
Device name           : NVIDIA TITAN RTX
Max threads per block : 1024
Max block dimensions  : 1024 x 1024 x 64
Max grid dimensions   : 2147483647 x 65535 x 65535
-------------------------------------------------
--------------- Input Parameters ---------------
Matrix size        : 8 x 8
Blocks per Grid    : 2
Threads per Block  : 4
-------------------------------------------------
The array A has been stored in file A/A8B.txt
Average: 64.41
Time for the kernel calcAvg<<<>>>(): 0.204736 ms
Max: 624
Time for the kernel findMax<<<>>>(): 0.015552 ms
The array B has been stored in file OutArr/OutArr8B.txt
Min: -0.0522
Time for the kernel createB<<<>>>(): 0.015040 ms
```

4.2.3 Output8C.txt

```
--------------- Device Properties ---------------
Device name           : NVIDIA TITAN RTX
Max threads per block : 1024
Max block dimensions  : 1024 x 1024 x 64
Max grid dimensions   : 2147483647 x 65535 x 65535
-------------------------------------------------
```

```
--------------- Input Parameters ---------------
Matrix size        : 8 x 8
Blocks per Grid    : 2
Threads per Block  : 4
------------------------------------------------
The array A has been stored in file A/A8C.txt
Average: 66.98
Time for the kernel calcAvg<<<>>>(): 0.207712 ms
Max: 444
Time for the kernel findMax<<<>>>(): 0.014592 ms
The array C has been stored in file OutArr/OutArr8C.txt
Time for the kernel createC<<<>>>(): 0.012992 ms
```

4.2.4 Output512.txt

```
--------------- Device Properties ---------------
Device name           : NVIDIA TITAN RTX
Max threads per block : 1024
Max block dimensions  : 1024 x 1024 x 64
Max grid dimensions   : 2147483647 x 65535 x 65535
-------------------------------------------------
--------------- Input Parameters ---------------
Matrix size        : 512 x 512
Blocks per Grid    : 32
Threads per Block  : 16
------------------------------------------------
The array A has been stored in file A/A512.txt
Average: 3.12
Time for the kernel calcAvg<<<>>>(): 0.136576 ms
Max: 100
Time for the kernel findMax<<<>>>(): 0.016704 ms
The array C has been stored in file OutArr/OutArr512.txt
Time for the kernel createC<<<>>>(): 0.016576 ms
```

4.2.5 Output1024.txt

```
--------------- Device Properties ---------------
Device name           : NVIDIA TITAN RTX
Max threads per block : 1024
Max block dimensions  : 1024 x 1024 x 64
Max grid dimensions   : 2147483647 x 65535 x 65535
-------------------------------------------------
--------------- Input Parameters ---------------
Matrix size        : 1024 x 1024
Blocks per Grid    : 32
Threads per Block  : 32
------------------------------------------------
The array A has been stored in file A/A1024.txt
Average: 1.52
```

```
Time for the kernel calcAvg<<<>>>(): 36.310913 ms
Max: 51241
Time for the kernel findMax<<<>>>(): 0.059424 ms
The array B has been stored in file OutArr/OutArr1024.txt
Min: -0.0019
Time for the kernel createB<<<>>>(): 0.072832 ms
```

4.2.6 Output10000.txt

```
--------------- Device Properties ---------------
Device name           : NVIDIA TITAN RTX
Max threads per block : 1024
Max block dimensions  : 1024 x 1024 x 64
Max grid dimensions   : 2147483647 x 65535 x 65535
-------------------------------------------------
--------------- Input Parameters ---------------
Matrix size        : 10000 x 10000
Blocks per Grid    : 100
Threads per Block  : 100
-------------------------------------------------
The array A has been stored in file A/A10000.txt
Average: 0.00
Time for the kernel calcAvg<<<>>>(): 29.475456 ms
Max: 0
Time for the kernel findMax<<<>>>(): 0.014784 ms
The array C has been stored in file OutArr/OutArr10000.txt
Time for the kernel createC<<<>>>(): 0.011424 ms
```

4.2.7 Output20000.txt

```
--------------- Device Properties ---------------
Device name           : NVIDIA TITAN RTX
Max threads per block : 1024
Max block dimensions  : 1024 x 1024 x 64
Max grid dimensions   : 2147483647 x 65535 x 65535
-------------------------------------------------
--------------- Input Parameters ---------------
Matrix size        : 20000 x 20000
Blocks per Grid    : 162
Threads per Block  : 124
-------------------------------------------------
The array A has been stored in file A/A20000.txt
Average: 0.00
Time for the kernel calcAvg<<<>>>(): 39.388447 ms
Max: 0
Time for the kernel findMax<<<>>>(): 0.015104 ms
The array C has been stored in file OutArr/OutArr20000.txt
Time for the kernel createC<<<>>>(): 0.011424 ms
```

4.3 Ανάλυση της αποδοτικότητας

4.3.1 Χρόνοι εκτέλεσης του παράλληλου αλγορίθμου

Οι χρόνοι εκτέλεσης αυξάνονται γραμμικά με το μέγεθος του πίνακα για το calcAvg, όπως αναμένεται λόγω του υπολογισμού που διασχίζει όλα τα στοιχεία. Η απόδοση για τα findMax και createC παραμένει σταθερή λόγω της χρήσης βελτιστοποιημένων παράλληλων μεθόδων.

Μέγεθος Πίνακα	**calcAvg (ms)**	**findMax (ms)**	**createB / createC (ms)**
8 x 8	0.204736	0.015552	0.015040 (B)
512 x 512	0.136576	0.016704	0.016576 (C)
1024 x 1024	36.310913	0.059424	0.072832 (B)
10000 x 10000	29.475456	0.014784	0.011424 (C)
20000 x 20000	39.388447	0.015104	0.011424 (C)

4.3.3 Παρατηρήσεις

Οι χρόνοι εκτέλεσης για το calcAvg παρουσιάζουν αυξητική τάση για μεγαλύτερα μεγέθη πινάκων, όπως αναμενόταν.

Η χρήση atomic εντολών στους πυρήνες (findMax, createB) εξασφαλίζει σταθερούς χρόνους ανεξάρτητα από το μέγεθος του πίνακα.
Η απόδοση του πυρήνα createC παραμένει εξαιρετικά υψηλή ακόμη και για μεγάλους πίνακες, χάρη στην παράλληλη επεξεργασία.

5. Προβλήματα και Αντιμετώπιση

5.1 Αναφορά προβλημάτων

1. **Πρόβλημα Δέσμευσης Μνήμης για Μεγάλα NNN:**
   Κατά την εκτέλεση για μεγάλους πίνακες (π.χ., 512×512), παρουσιάστηκε αποτυχία κατά τη μεταφορά δεδομένων από τη μνήμη της συσκευής (device) στη μνήμη του host:
   - **Σφάλμα:** CUDA Error --> cudaMemcpy(h_avg, d_avg, sizeof(float), cudaMemcpyDeviceToHost) failed.cudamemcpy_error
   - **Αιτία:** Η μη επαρκής μνήμη για την αποθήκευση μεγάλων πινάκων στη συσκευή ή η κακή διαχείριση μνήμης μεταξύ host και device.
2. **Πρόβλημα Χρήσης Shared Memory:**
   Κατά τη μεταγλώττιση του κώδικα, οι πυρήνες calcAvg, findMax και createB εμφάνισαν σφάλματα λόγω υπέρβασης της μέγιστης ποσότητας κοινής μνήμης που διατίθεται ανά μπλοκ:
   - **Σφάλμα:**

vbnet
Copy code
ptxas error: Entry function '_Z7createBPiPfS0_S_S0_' uses too much shared data (0x400000 bytes, 0x18000 max)

shared_mem_err

- **Αιτία:** Ο ορισμός μεγάλων πινάκων shared memory για κάθε μπλοκ υπερβαίνει τη μέγιστη επιτρεπόμενη χωρητικότητα της συσκευής.

3. **Θέματα στον Συγχρονισμό Νημάτων:**
   Κατά την εκτέλεση παράλληλων λειτουργιών με χρήση shared memory, παρατηρήθηκαν δυσκολίες στον σωστό συγχρονισμό των νημάτων:
   - Τα __syncthreads() δεν χρησιμοποιήθηκαν επαρκώς ή σωστά, προκαλώντας ασυνεπή αποτελέσματα σε ορισμένες περιπτώσεις.

5.2 Λύσεις που δοκιμάστηκαν και εφαρμόστηκαν

Για το Πρόβλημα Δέσμευσης Μνήμης:

- **Βελτιστοποίηση Μεταφοράς Δεδομένων:**
  - Οι μεταφορές δεδομένων από το host στη συσκευή και το αντίστροφο διασπάστηκαν σε μικρότερα τμήματα. Έτσι, μειώθηκε η πιθανότητα υπέρβασης της διαθέσιμης μνήμης.
- **Έλεγχος Μεγέθους Πινάκων:**
  - Εφαρμόστηκε δυναμικός έλεγχος για να διασφαλιστεί ότι το N δεν ξεπερνά τα όρια της συσκευής.

- **Για Θέματα Συγχρονισμού Νημάτων:**

  - **Προσθήκη `__syncthreads()`:**
    - Εξασφαλίστηκε ότι όλα τα νήματα του μπλοκ έχουν ολοκληρώσει τη φόρτωση δεδομένων στην shared memory πριν την έναρξη των υπολογισμών.

ΠΑΡΑΛΛΗΛΑ ΣΥΣΤΗΜΑΤΑ

Σας ευχαριστώ για την προσοχή σας.