Παράλληλη Επεξεργασία
Εαρινό Εξάμηνο 2025-2026

Εργασία
"Παράλληλη Προεπεξεργασία Μεγάλου Όγκου Δεδομένων"

Τελική Ημερομηνία Παράδοσης: Θα ανακοινωθεί

1. Εισαγωγή

Η προεπεξεργασία δεδομένων αποτελεί ένα από τα σημαντικότερα στάδια σε εφαρμογές μηχανικής μάθησης, επιστημονικών υπολογισμών και ανάλυσης μεγάλων δεδομένων. Πριν την εκπαίδευση ενός μοντέλου ή την εφαρμογή ενός αλγορίθμου ανάλυσης, τα δεδομένα συχνά πρέπει να καθαριστούν, να μετασχηματιστούν και να κανονικοποιηθούν, ώστε τα χαρακτηριστικά τους να βρίσκονται σε κατάλληλη αριθμητική κλίμακα.

Σε μικρά σύνολα δεδομένων, τέτοιου είδους μετασχηματισμοί μπορούν να υλοποιηθούν εύκολα με βιβλιοθήκες όπως NumPy, pandas ή scikit-learn. Ωστόσο, όταν ο πίνακας δεδομένων είναι πολύ μεγάλος, η επεξεργασία του γίνεται υπολογιστικά απαιτητική. Επιπλέον, όταν το μέγεθος των δεδομένων υπερβαίνει τη διαθέσιμη κύρια μνήμη, η απόδοση δεν εξαρτάται μόνο από την ταχύτητα του επεξεργαστή, αλλά και από το κόστος ανάγνωσης και εγγραφής από το αποθηκευτικό σύστημα.

Στην παρούσα εργασία ζητείται η υλοποίηση ενός συστήματος παράλληλης προεπεξεργασίας μεγάλων αριθμητικών δεδομένων. Το πρόγραμμα θα διαβάζει έναν μεγάλο πίνακα από δυαδικό αρχείο, θα υπολογίζει στατιστικά ανά στήλη και θα εφαρμόζει μετασχηματισμούς κανονικοποίησης, όπως StandardScaler και MinMaxScaler.

Μηχανισμοί που θα χρησιμοποιηθούν για την αποδοτική προεπεξεργασία των δεδομένων είναι οι εξής:

- παραλληλισμός σε επίπεδο εντολών μέσω SIMD
- πολυνηματικός προγραμματισμός με OpenMP
- κατανεμημένος προγραμματισμός με MPI
- επιτάχυνση με GPU μέσω CUDA
- αποδοτικό I/O
- out-of-core επεξεργασία μεγάλων αρχείων

2. Περιγραφή του προβλήματος

Δίνεται ένας μεγάλος πίνακας πραγματικών αριθμών:

$$X \in R^{(N \times D)}$$

όπου N είναι ο αριθμός των δειγμάτων και D είναι ο αριθμός των χαρακτηριστικών. Κάθε γραμμή αντιστοιχεί σε ένα δείγμα και κάθε στήλη αντιστοιχεί σε ένα χαρακτηριστικό. Ο πίνακας είναι αποθηκευμένος σε δυαδικό αρχείο, σε διάταξη row-major, δηλαδή τα στοιχεία αποθηκεύονται διαδοχικά ανά γραμμή:

$$x_{00}, x_{01}, \ldots, x_{0,D-1}, x_{10}, x_{11}, \ldots, x_{N-1,D-1}$$

Το πρόγραμμα πρέπει να διαβάζει το αρχείο εισόδου, να υπολογίζει στατιστικά ανά στήλη και να παράγει νέο δυαδικό αρχείο με τα κανονικοποιημένα δεδομένα. Η επεξεργασία πρέπει να υποστηρίζει μεγάλα αρχεία, τα οποία μπορεί να μην χωρούν εξ ολοκλήρου στη διαθέσιμη κύρια μνήμη.

2.1 Μορφή Δεδομένων Εισόδου

Το αρχείο εισόδου είναι raw binary αρχείο χωρίς επικεφαλίδα και χωρίς metadata. Αυτό σημαίνει ότι το πρόγραμμα δεν μπορεί να γνωρίζει από μόνο του τις διαστάσεις του πίνακα. Οι διαστάσεις δίνονται από τη γραμμή εντολών. Για την κύρια εκδοχή της εργασίας θεωρούμε ότι τα δεδομένα είναι τύπου double. Προαιρετικά, μπορεί να υποστηριχθεί και τύπος float. Το μέγεθος του αρχείου εισόδου σε bytes είναι: file_size = N · D · sizeof(double). Για παράδειγμα, αν N = 10^7 και D = 128, τότε το μέγεθος του αρχείου είναι: 10^7 · 128 · 8 = 10.24 GB
Άρα, ακόμη και σχετικά απλές διαστάσεις μπορούν να οδηγήσουν σε αρχεία πολλών GB.

2.2 Παραγωγή Δεδομένων

Τα δεδομένα μπορούν να παραχθούν με Python, χρησιμοποιώντας ενδεικτικά τη συνάρτηση make_regression της βιβλιοθήκης scikit-learn ή άλλη αντίστοιχη διαδικασία παραγωγής συνθετικών αριθμητικών δεδομένων. Τα παραγόμενα δεδομένα αποθηκεύονται σε raw binary μορφή, ώστε να μπορούν να διαβαστούν από το πρόγραμμα C/C++. Η παραγωγή δεδομένων είναι παραμετρική ως προς:

- τον αριθμό δειγμάτων N
- τον αριθμό χαρακτηριστικών D
- τον τύπο δεδομένων
- το όνομα του αρχείου εξόδου
- το seed της γεννήτριας τυχαίων αριθμών

Παράδειγμα εντολής παραγωγής δεδομένων:

```
python3 generate_data.py \
  --samples 10000000 \
  --features 128 \
  --output data_10M_128.bin \
  --dtype float64
```

2.3 Εκτέλεση Προγράμματος

Το πρόγραμμα πρέπει να εκτελείται από τη γραμμή εντολών ως εξής:

```
./scaler input.bin output.bin N D mode
```

Παράμετρος	**Περίγραφή**
input.bin	Αρχείο εισόδου
output.bin	Αρχείο εξόδου
N	Αριθμός γραμμών
D	Αριθμός στηλών
Mode	Τύπος μετασχηματισμού: standard ή minmax

Παραδείγματα εκτέλεσης:

```
./scaler data_10M_128.bin out_standard.bin 10000000 128 standard
```

```
./scaler data_10M_128.bin out_minmax.bin 10000000 128 minmax
```

Προαιρετικά, μπορεί να υποστηρίζεται επιπλέον παράμετρος block_rows, η οποία δηλώνει τον αριθμό γραμμών που διαβάζονται ανά block.

2.4 Υπολογισμός Στατιστικών

Για κάθε στήλη j, όπου 0 <= j < D, πρέπει να υπολογιστούν τα παρακάτω στατιστικά.

a. Μέση Τιμή: *μ_j = (1/N) · Σ_{i=0}^{N-1} x_ij*

b. Ελάχιστη Τιμή: *min_j = min_{0 <= i < N} x_ij*

c. Μέγιστη Τιμή: *max_j = max_{0 <= i < N} x_ij*

d. Διακύμανση: *σ_j^2 = (1/N) · Σ_{i=0}^{N-1} (x_ij - μ_j)^2*
Εναλλακτικά, μπορεί να χρησιμοποιηθεί η ισοδύναμη σχέση: σ_j^2 = (1/N) · Σ_{i=0}^{N-1} x_ij^2 - μ_j^2

e. Τυπική Απόκλιση: *σ_j = sqrt(σ_j^2)*
Θεωρούμε πως δεν υπάρχει περίπτωση τα στοιχεία μιας στήλης να είναι ίδια, δηλαδή σ_j = 0.

2.5 Μετασχηματισμοί Κανονικοποίησης

a. StandardScaler

$$x'_{ij} = (x_{ij} - \mu_j) / \sigma_j$$

Μετά τον μετασχηματισμό, κάθε στήλη πρέπει θεωρητικά να έχει μέση τιμή περίπου 0 και τυπική απόκλιση περίπου 1. Σε περίπτωση όπου σ_j = 0, πρέπει να αποφευχθεί η διαίρεση με το μηδέν.

b. MinMaxScaler

$$x'_{ij} = (x_{ij} - min_j) / (max_j - min_j)$$

Μετά τον μετασχηματισμό, οι τιμές κάθε στήλης πρέπει θεωρητικά να βρίσκονται στο διάστημα [0, 1]. Σε περίπτωση όπου max_j = min_j, πρέπει να αποφευχθεί η διαίρεση με το μηδέν.

3. Φάσεις Επεξεργασίας

Η υλοποίηση πρέπει να βασίζεται σε λογική δύο φάσεων. Δεν επιβάλλεται συγκεκριμένη υλοποίηση, όμως πρέπει να διασφαλιστεί ότι τα συνολικά στατιστικά είναι γνωστά πριν από την εφαρμογή του scaling.

3.1 Φάση Υπολογισμού Στατιστικών

Στην πρώτη φάση, το πρόγραμμα διαβάζει τα δεδομένα εισόδου και υπολογίζει τα απαιτούμενα στατιστικά ανά στήλη: άθροισμα, άθροισμα τετραγώνων, ελάχιστο, μέγιστο, μέση τιμή, διακύμανση και τυπική απόκλιση.

3.2 Φάση Μετασχηματισμού

Στη δεύτερη φάση, το πρόγραμμα εφαρμόζει τον επιλεγμένο μετασχηματισμό στα δεδομένα εισόδου και γράφει το αποτέλεσμα σε νέο αρχείο. Η φάση αυτή πρέπει να χρησιμοποιεί τα στατιστικά που υπολογίστηκαν στην πρώτη φάση.

3.3 Απαιτήσεις για τις δύο φάσεις

1. Το αρχείο εισόδου μπορεί να χρειαστεί να διαβαστεί περισσότερες από μία φορές.
2. Η εφαρμογή του scaling δεν μπορεί να γίνει σωστά πριν υπολογιστούν τα συνολικά στατιστικά.
3. Η προσέγγιση πρέπει να δουλεύει και όταν ο πίνακας δεν χωράει εξ ολοκλήρου στη μνήμη.
4. Η διαχείριση του αρχείου εξόδου πρέπει να διασφαλίζει ότι τα δεδομένα γράφονται στη σωστή σειρά.

3.4 Out-of-Core Επεξεργασία

Σε πραγματικά προβλήματα, το αρχείο εισόδου μπορεί να είναι μεγαλύτερο από τη διαθέσιμη μνήμη. Επομένως, δεν πρέπει να θεωρείται ότι ολόκληρος ο πίνακας χωράει στη RAM. Η εφαρμογή πρέπει να υποστηρίζει block-based επεξεργασία.

Αν το block περιέχει B γραμμές, τότε το μέγεθος του block σε bytes είναι:

$$block\_size = B \cdot D \cdot sizeof(double)$$

Η επιλογή του B πρέπει να γίνεται έτσι ώστε το block να χωράει άνετα στη μνήμη. Για παράδειγμα, αν B = 100000 και D = 128, τότε το μέγεθος block είναι περίπου 102.4 MB.

4. Υλοποίηση

4.1. Σειριακή Υλοποίηση Αναφοράς

Η σειριακή υλοποίηση είναι απαραίτητη για δύο λόγους: χρησιμοποιείται ως βάση ελέγχου ορθότητας και ως βάση υπολογισμού speedup. Η σειριακή έκδοση πρέπει να είναι σωστή, καθαρά γραμμένη και επαρκώς τεκμηριωμένη. Η σειριακή έκδοση πρέπει να υποστηρίζει:

- ανάγνωση raw binary αρχείου
- υπολογισμό στατιστικών ανά στήλη
- εφαρμογή StandardScaler
- εφαρμογή MinMaxScaler
- εγγραφή raw binary αρχείου εξόδου
- επεξεργασία μεγάλων αρχείων με blocks

4.2. SIMD

Η SIMD υλοποίηση στοχεύει στην αξιοποίηση vector instructions του επεξεργαστή, όπως AVX, AVX2 ή AVX-512. Η βασική ιδέα είναι ότι αντί να επεξεργαζόμαστε ένα στοιχείο κάθε φορά, επεξεργαζόμαστε πολλά στοιχεία ταυτόχρονα μέσα σε vector registers. Μπορεί να αξιοποιηθεί automatic vectorization από τον compiler, explicit SIMD intrinsics, κατάλληλη διάταξη loops και κατάλληλα compiler flags. Στην αναφορά πρέπει να εξηγηθεί:

1. Ποια τμήματα του αλγορίθμου επιχειρήθηκε να γίνουν vectorized.
2. Αν χρησιμοποιήθηκε automatic vectorization ή explicit intrinsics.
3. Ποια compiler flags χρησιμοποιήθηκαν.
4. Αν η SIMD υλοποίηση βελτίωσε την απόδοση.
5. Ποιοι παράγοντες περιόρισαν το speedup.

4.3. OpenMP

Η OpenMP έκδοση στοχεύει στην αξιοποίηση πολλών πυρήνων σε σύστημα κοινής μνήμης. Οι φοιτητές πρέπει να παραλληλοποιήσουν κατάλληλα τον υπολογισμό στατιστικών, την εφαρμογή του scaling και πιθανώς την επεξεργασία των blocks. Ιδιαίτερη προσοχή πρέπει να δοθεί στην αποφυγή race conditions, καθώς πολλά threads μπορεί να ενημερώνουν τα ίδια στατιστικά ανά στήλη. Η OpenMP υλοποίηση πρέπει να τεκμηριώνει:

1. Τον τρόπο κατανομής εργασίας στα threads.
2. Τη στρατηγική αποφυγής race conditions.
3. Τη χρήση ή μη χρήση reductions.
4. Τη συμπεριφορά για διαφορετικό αριθμό threads.
5. Την επίδραση του block size στην απόδοση.

Η μέτρηση της απόδοσης θα πρέπει να γίνει για διαφορετικό αριθμό threads, ενδεικτικά: 1, 2, 4, 8, 16

4.4. MPI

Η MPI έκδοση στοχεύει στην εκτέλεση του προγράμματος σε κατανεμημένη μνήμη, δηλαδή σε πολλαπλές διεργασίες που μπορεί να εκτελούνται στον ίδιο ή σε διαφορετικούς κόμβους. Η βασική ιδέα είναι ότι κάθε MPI διεργασία αναλαμβάνει ένα διαφορετικό τμήμα του πίνακα δεδομένων.

Οι φοιτητές πρέπει να σχεδιάσουν την κατανομή των δεδομένων στις διεργασίες, τον τρόπο ανάγνωσης του αρχείου εισόδου, τον τοπικό υπολογισμό στατιστικών, τη συγχώνευση των τοπικών στατιστικών, την εφαρμογή του scaling και την εγγραφή του αποτελέσματος. Στην αναφορά πρέπει να περιγράφεται:

- πώς κατανέμονται οι γραμμές στις MPI διεργασίες
- πώς υπολογίζονται τα συνολικά στατιστικά
- αν χρησιμοποιήθηκε MPI I/O ή κλασικό file I/O
- ποιο είναι το κόστος των συλλογικών επικοινωνιών
- αν εμφανίζεται bottleneck στο filesystem

Η μέτρηση της απόδοσης θα πρέπει να γίνει για διαφορετικό αριθμό MPI διεργασιών, ενδεικτικά: 1, 2, 4, 8, 16

4.5. CUDA Υλοποίηση

Η CUDA έκδοση στοχεύει στην επιτάχυνση των υπολογισμών μέσω GPU. Η GPU μπορεί να χρησιμοποιηθεί για υπολογισμό μερικών στατιστικών, εφαρμογή του scaling και επεξεργασία block-by-block όταν τα δεδομένα δεν χωρούν στη μνήμη της GPU. Πρέπει να αποφασιστεί ποια μέρη της εφαρμογής θα εκτελούνται στη CPU και ποια στη GPU. Στην CUDA υλοποίηση πρέπει να εξεταστούν το κόστος μεταφοράς δεδομένων CPU-GPU, το κόστος μεταφοράς δεδομένων GPU-CPU, η εκτέλεση kernels, η χρήση μνήμης GPU, η επίδραση του μεγέθους block και η περίπτωση όπου το dataset δεν χωράει ολόκληρο στη μνήμη της GPU.

5. Πειραματική Αξιολόγηση

Η πειραματική αξιολόγηση αποτελεί βασικό μέρος της εργασίας. Δεν αρκεί να υλοποιηθεί το πρόγραμμα. Πρέπει να μετρηθεί και να αναλυθεί η απόδοσή του για τουλάχιστον δύο διαφορετικά μεγέθη δεδομένων.

Ενδεικτικά Μεγέθη Δεδομένων

Περίπτωση	**N**	**D**	**Μέγεθος αρχείου (double)**
Small	1,000,000	32	~ 256 MB
Medium	5,000,000	64	~ 2.56 GB
Large	10,000,000	128	~ 10.24 GB
Very Large	50,000,000	128	~ 51.2 GB

Έλεγχος Ορθότητας

Η ορθότητα πρέπει να ελεγχθεί συγκρίνοντας το παραγόμενο αρχείο με reference αποτέλεσμα από Python/NumPy ή scikit-learn. Ο έλεγχος πρέπει να περιλαμβάνει σύγκριση της σειριακής έκδοσης με υλοποίηση αναφοράς και σύγκριση κάθε παράλληλης έκδοσης με τη σειριακή ή με την υλοποίηση αναφοράς. Σε κάθε περίπτωση να αναφερθούν τα max και mean absolute errors.

6. Παραδοτέα

Θα πρέπει να παραδώσετε τα ακόλουθα:

1. Ένα συμπιεσμένο αρχείο **ZIP** με τον **φάκελο που έχει τους κώδικες σας**.

Συγκεκριμένα, το αρχείο πρέπει να περιέχει:

   a. Πηγαίο κώδικα.
   b. Makefile
   c. Scripts παραγωγής δεδομένων. Δεν πρέπει να παραδοθούν μεγάλα binary αρχεία δεδομένων.
   d. Scripts εκτέλεσης πειραμάτων.
   e. Scripts ελέγχου ορθότητας.
   f. README με οδηγίες εκτέλεσης

2. Ένα αρχείο **PDF** με την **αναλυτική γραπτή αναφορά** για την εργασία σας.

Στο εξώφυλλο της γραπτής αναφοράς θα συμπεριλάβετε το όνομα (**επώνυμο, όνομα**) και τον **ΑΜ** όλων των μελών της ομάδας. Στην αναφορά θα περιγράφετε και τεκμηριώνετε αναλυτικά τις παράλληλες υλοποιήσεις και τις σχεδιαστικές αποφάσεις σας και θα παρουσιάσετε τα αποτελέσματά σας

Σχολιάστε τις διαφορές από την χρήση βελτιστοποιήσεων στο χρόνο εκτέλεσης της εφαρμογής και (κυρίως) στην χρονοβελτίωση. Γενικότερα, δώστε ιδιαίτερο βάρος στην αναφορά σε αυτά που κάνατε εσείς. Η διαμόρφωση της αναφοράς σας θα πρέπει να είναι παρόμοια με εκείνη που έχει χρησιμοποιηθεί για την παρουσίαση των λύσεων των ασκήσεων του μαθήματος. Η αναφορά θα πρέπει να περιλαμβάνει τουλάχιστον τις ακόλουθες ενότητες:

- Εισαγωγή
- Αναλυτική τεκμηρίωση παράλληλων υλοποιήσεων και τυχόν βελτιστοποιήσεων
- Αποτελέσματα
- Συμπεράσματα

Σε περίπτωση που η εργασία πραγματοποιηθεί σε διαφορετικό μηχάνημα από αυτό που θα σας δοθεί, τότε θα πρέπει να αναφερθούν οι πληροφορίες για τα εξής:

- Operating System
- CPU
- Cores
- RAM
- GPU
- Disk / filesystem
- Compiler
- MPI implementation
- CUDA version

7. Δήλωση ομάδας

Η εργασία θα πρέπει να γίνει σε ομάδες έως 2 ατόμων.

8. Παράδοση εργασίας

Η παράδοση της εργασίας θα γίνει από τον υπεύθυνο της κάθε ομάδας στο “E-class” του μαθήματος. Για λόγους δικαιοσύνης και ισοτιμίας, εκπρόθεσμες εργασίες δεν θα ληφθούν υπόψη και δεν θα αξιολογηθούν. **Κάθε ομάδα θα παραδώσει μια φορά μόνο την εργασία (όχι κάθε φοιτητής ξεχωριστά).**

8. Βαθμολογία

Η εργασία παραδίδεται ΜΟΝΟ κατά την εξεταστική Ιουνίου (υποβολές που θα γίνουν κατά την διάρκεια της εξεταστικής Σεπτεμβρίου ή της άτυπης εξεταστικής Φεβρουαρίου δεν θα γίνουν δεκτές).

Ο βαθμός της εργασίας διατηρείται μέχρι και την άτυπη εξεταστική Φεβρουαρίου 2027.

Αν απαιτείται, θα προστεθούν διευκρινίσεις, βοηθητικές πληροφορίες και απαντήσεις στο “E-Class”.

Καλή επιτυχία!