P. Hadjidoukas, E. Dermatas, E. Gallopoulos

Set 1 - MPI and OpenMP

Issued: November 21, 2024

Question 1: Hybrid Programming Model and Parallel I/O

a) Υλοποίηστε την κλήση MPI_Exscan του μοντέλου προγραμματισμού MPI χρησιμοποιώντας point-to-point επικοινωνίες (send/recv). Θεωρήστε πως έχετε P διεργασίες και πως η υλοποίηση της MPI_Exscan θα λειτουργεί για ακέραιες τιμές, δηλαδή μεταβλητές τύπου int. Μπορείτε να ονομάσετε τη νέα συνάρτηση MPI_Exscan _pt2pt.

b) Επεκτείνετε την προηγούμενη υλοποίηση ώστε να είναι κατάλληλη για το υβριδικό μοντέλο προγραμματισμού (MPI + OpenMP). Μια εφαρμογή εκτελείται με P διεργασίες και T νήματα σε κάθε διεργασία. Κάθε νήμα της εφαρμογής εκτελεί τον ίδιο κώδικα, εφαρμόζοντας το προγραμματιστικό παράδειγμα SPMD (Single Program Multiple Data). Ολα τα νήματα καλούν ταυτόχρονα την νέα συνάρτηση (MPI_ Exscan_omp) με κάθε νήμα να χρησιμοποιεί μια δική του ακεραία τιμή. Η υλοποίηση της συνάρτηση εκμεταλλεύεται και τα δύο μοντέλα προγραμματισμού (MPI, OpenMP).

c) Επεκτείνοντας και αξιοποιώντας τον κώδικα του προγούμενου ερωτήματος υλοποιήστε το παρακάτω σενάριο: Κάθε νήμα της υβριδικής εφαρμογής δεσμεύει και αρχικοποιεί, με συγκεκριμένη τιμή ενός ιδιωτικού randon seed, ένα μητρώο πραγματικών αριθμών διάστασης $NxNxN$. Στη συνέχεια, χρησιμοποιώντας MPI I/O η εφαρμογή θα γράφει τα μητρώα σε ένα δυαδικό αρχείο. Η φάση εγγραφής θα ενεργοποιείται ταυτόχρονα από όλα τα νήματα της εφαρμογής. Για τον υπολογισμό της σωστής θέσης εγγραφής κάθε μητρώου, χρησιμοποιήστε τη συνάρτηση που υλοποιήσατε στο προηγούμενο ερώτημα, έστω και αν το μέγεθος των δεδομένων είναι το ίδιο για κάθε νήμα. Επίσης, θα πρέπει να επιβεβαιώσετε προγραμματιστικά, στην ίδια εφαρμογή ή σε διαφορετική (παράλληλη ή σειριακή), πως τα δεδομένα στο δυαδικό αρχείο είναι σωστά. Αυτό προυποθέτει πως η εφαρμογή διαβάζει τα δεδομένα από το αρχείο και τα συγκρίνει με τα αρχικά (βλ. αρχικοποίηση κάθε μητρώου με συγκεκριμένες τυχαίες τιμές).

d) Επαναλάβετε το προηγούμενο ερώτημα χρησιμοποιώντας έναν συμπιεστή δεδομένων. Πριν τη διαδικασία εγγραφής, κάθε νήμα θα συμπιέζει το τριδιάστατο μητρώο του και το αποτέλεσμα της συμπίεσης, το οποίο πιθανότατα έχει διαφορετικό μέγεθος για κάθε μητρώο, θα γράφεται στο δυαδικό αρχείο.

   Η συμπίεση του μητρώου μπορεί να πραγματοποιηθεί με κάποια τυπική βιβλιοθήκη συμπίεσης δεδομένων, όπως οι ZLIB και ZSTD, ή με κάποια βιβλιοθήκη συμπίεσης επιστημονικών δεδομένων, όπως οι ZFP και SZ.

Question 2: Parallel Parametric Search in Machine Learning

Εισαγωγή

Η παραμετρική αναζήτηση στη μηχανική μάθηση είναι μια διαδικασία που στοχεύει στη βελτίωση της απόδοσης ενός μοντέλου με τη βελτιστοποίηση των παραμέτρων του. Οι παράμετροι αυτές είναι συνήθως γνωστές ως υπερπαράμετροι και επηρεάζουν άμεσα την απόδοση του μοντέλου. Υπάρχουν διάφορες μέθοδοι για την υλοποίηση της παραμετρικής αναζήτησης, όπως η αναζήτηση πλέγματος (grid search), η τυχαία αναζήτηση (random search), και πιο προχωρημένες τεχνικές όπως η Bayesian optimization.

Η αναζήτηση πλέγματος είναι μια εξαντλητική διαδικασία που δοκιμάζει όλους τους δυνατούς συνδυασμούς των υπερπαραμέτρων σε ένα προκαθορισμένο πλέγμα. Για παράδειγμα, αν έχουμε δύο υπερπαραμέτρους με τρεις και τέσσερις δυνατές τιμές η κάθε μία, η αναζήτηση πλέγματος θα δοκιμάσει όλους τους 3 x 4 = 12 συνδυασμούς.

Η τυχαία αναζήτηση επιλέγει τυχαίους συνδυασμούς υπερπαραμέτρων μέσα σε έναν προκαθορισμένο χώρο. Σε αντίθεση με την αναζήτηση πλέγματος, η τυχαία αναζήτηση δεν ερευνά όλους τους δυνατούς συνδυασμούς, αλλά μια προκαθορισμένη ποσότητα αυτών. Αυτό μπορεί να είναι πιο αποδοτικό σε μεγάλο χώρο παραμέτρων. Η Bayesian optimization είναι μια πιο εξελιγμένη τεχνική που χρησιμοποιεί ένα στατιστικό μοντέλο για να προβλέψει ποιοι συνδυασμοί υπερπαραμέτρων είναι πιθανότερο να αποδώσουν καλύτερα. Στηρίζεται σε προηγούμενα αποτελέσματα και προσαρμόζεται δυναμικά καθώς η διαδικασία προχωρά.

```
1  from sklearn.neural_network import MLPClassifier
2  from sklearn.model_selection import ParameterGrid
3  from sklearn.datasets import make_classification
4  from sklearn.metrics import accuracy_score
5  from sklearn.model_selection import train_test_split
6
7  X, y = make_classification(n_samples=10000, random_state=42, n_features=2,
8      n_informative=2, n_redundant=0, class_sep=0.8)
9
10 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33,
11     random_state=42)
12
13
14 params = [{'mlp_layer1': [16, 32],
15             'mlp_layer2': [16, 32],
16             'mlp_layer3': [16, 32]}]
17
18 pg = ParameterGrid(params)
19
20 results = []
21 for i, p in enumerate(pg):
22     print(p)
23     l1 = p['mlp_layer1']
24     l2 = p['mlp_layer2']
25     l3 = p['mlp_layer3']
26     m = MLPClassifier(hidden_layer_sizes=(l1, l2, l3))
27     m.fit(X_train, y_train)
28     y_pred = m.predict(X_test)
29     ac = accuracy_score(y_pred, y_test)
30     print(i, ac)
31     results.append((i, p, ac))
32
33 for r in results:
34     print(r)
```

Εφαρμογή

Για να γίνει παραμετρική αναζήτηση, πρέπει πρώτα να καθοριστούν οι υπερπαράμετροι που θα βελτιστοποιηθούν. Για παράδειγμα, σε ένα Random Forest Classifier, οι υπερπαράμετροι

μπορεί να περιλαμβάνουν τον αριθμό των δέντρων (n_estimators), το μέγιστο βάθος των δέντρων (max_depth), και το κλάσμα των χαρακτηριστικών που χρησιμοποιούνται για τον σχηματισμό κάθε δέντρου (max_features). Αντίστοιχα, σε ένα MLP Classifier, δηλαδή σε ένα νευρωνικό δίκτυο, τυπικές υπερπαράμετροι αποτελούν ο αριθμός των επιπέδων, ο αριθμός των νευρώνων ανά επίπεδο, ο αλγόριθμος βελτιστοποίησης των βαρών και ο ρυθμός μάθησης.

Παράλληλη Παραμετρική Αναζήτηση
Βασικός στόχος της άσκησης είναι η παραλληλοποίηση της παραμετρικής αναζήτησης σε προβλήματα μηχανικής μάθησης, όπως αυτό που φαίνεται στον παραπάνω σειριακό Python κώδικα. Η επεξεργασία κάθε ομάδας παραμέτρων μπορεί να εκτελεστεί ανεξάρτητα, επομένως ο χρόνος της αναζήτησης μπορεί να μειωθεί με την αξιοποίηση του διαθέσιμου παραλληλισμού επιπέδου εργασιών (task-level parallelism). Η παραπάνω αξιοποίηση στην Python, σε έναν υπολογιστικό κόμβο, πρέπει να γίνει με τις παρακάτω μεθόδους

- multiprocessing pool
- mpi futures
- με υλοποίηση του μοντέλου master-worker χρησιμοποιώντας MPI, αντίστοιχη με αυτή που είδαμε στη διάλεξη του μαθήματος.

Μετρήσεις
Αφού ολοκληρώσετε τη παράλληλη έκδοση της εφαρμογής και βεβαιωθείτε για τη σωστή λειτουργία της, θα μετρήσετε την απόδοσή του ώστε να δείξετε τη σωστή αξιοποίηση του διαθέσιμου πολυεπεξεργαστικού υλικού. Θα πρέπει να μετρήσετε το χρόνο εκτέλεσης της εφαρμογής, για διάφορους αριθμούς επεξεργαστικών στοιχείων (διεργασιών) ώστε να υπολογίσετε τη χρονοβελτίωση (speedup) της εφαρμογής. Μπορείτε να προσαρμόσετε τα δεδομένα του προβλήματος στον αρχικό σειριακό κώδικα (π.χ. μέγεθος δεδομένων, αριθμός επιπέδων και νευρώνων στο δίκτυο) σύμφωνα με τις ανάγκες σας.

Question 3: OpenMP Tasking

Στον παρακάτω κώδικα, η συνάρτηση initialize θέτει τιμές σε ένα διάνυσμα A διάστασης N χρησιμοποιώντας την thread-safe και χρονοβόρα συνάρτηση work, που είναι υλοποιημένη κάπου αλλού.

```
1  extern double work(int i);
2  void initialize(double *A, int N)
3  {
4      #pragma omp parallel for schedule(dynamic, 2)
5      for (int i = 0; i < N; i++) {
6          A[i] = work(i);
7      }
8  }
```

a) Εξηγήστε πώς διαμοιράζονται οι επαναλήψεις του βρόγχου στα νήματα.

b) Γράψτε μία ισοδύναμη υλοποίηση του παράλληλου κώδικα που θα βασίζεται στο μοντέλο εργασιών (tasks) του OpenMP.

Παραδοτέα

Κώδικας (σε αρχείο zip) και αναφορά (σε αρχείο pdf) με ανάλυση του προβλήματος, εξήγηση της λύσης, και παρουσίαση των αποτελεσμάτων.