ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΑΡΑΛΛΗΛΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΣΕ ΠΡΟΒΛΗΜΑΤΑ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ

1η Εργαστηριακή Άσκηση

Ο αλγόριθμος των κ-μέσων “K-MEANS”

ΠΡΟΕΤΟΙΜΑΣΙΑ

Ο αλγόριθμος k-means ομαδοποιεί διανύσματα σε ένα προκαθορισμένο αριθμό κατηγοριών, βάσει μιας συνάρτησης απόστασης. Αποτελεί έναν από τους πλέον δημοφιλείς αλγόριθμος ομαδοποίησης διανυσμάτων.
Από ένα σύνολο N διανυσμάτων κάθε ένα του οποίου αποτελείται από Nv πραγματικούς αριθμούς, ομαδοποιεί τα διανύσματα σε Nc κατηγορίες ή ομάδες ελαχιστοποιώντας την απόσταση των διανυσμάτων κάθε κατηγορίας από το κέντρο της κάθε κατηγορίας. Για περισσότερες πληροφορίες δείτε εδώ
https://en.wikipedia.org/wiki/K-means_clustering
ή εδώ
https://www.youtube.com/watch?v=_aWzGGNrcic

Για όσους ενδιαφέρονται περισσότερο για τον αλγόριθμο και τις παραλλαγές του ας διαβάσουν αυτην την ιστοσελίδα
https://en.wikipedia.org/wiki/K-means_clustering

ΕΚΦΩΝΗΣΗ

Στα πλαίσια του εργαστηρίου θα υλοποιήσουμε την πλέον απλή μορφή του αλγόριθμου.

Αρχικά κατασκευάζουμε N τυχαία διανύσματα. Είναι αυτά που θα ομαδοποιήσουμε σε Nc κατηγορίες

Βήμα 0: Κατασκευή τυχαίων διανυσμάτων. Με την βοήθεια της rand() κατασκευάζουμε N διανύσματα διάστασης Nv το κάθε ένα: Vec[N][Nv]

Ο Αλγόριθμος Κ-μέσων
Βήμα 1: Αρχικοποίηση των κέντρων. Ορίζουμε τα κέντρα των Nc κατηγοριών επιλέγοντας τυχαία Nc διανυσματα απο τα N διαθέσιμα τα οποία θεωρούμε ότι είναι τα κέντρα των κατηγοριών Center[Nc][Nv]. Προσοχή θα πρέπει να δοθεί έτσι ώστε να μην επιλέξουμε ίδια κέντρα.
Βήμα 2: Εφόσον γνωρίζουμε τα κέντρα, υπολογίζουμε για κάθε ένα διάνυσμα του Vec[N][Nv] την ελάχιστη ευκλείδια απόσταση από τα κέντρα Center[Nc][Nv] και την πληροφορία αυτή την αποθηκεύουμε στον πίνακα Classes[N].

Βήμα 3: Εφόσον γνωρίζουμε από τις τιμές του πίνακα Classes[N] την πληροφορία της κατηγορίας κάθε διανύσματος Vec, υπολογίζουμε τα νέα κέντρα Center[Nc][Nv] από την μέση τιμή των διανυσμάτων Vec που ανήκουν στην κάθε κατηγορία.
Βήμα 4: Αν η μεταβολή της αθροιστικής απόστασης των διανυσμάτων από τα αντίστοιχα κέντρα είναι μικρότερη από ένα ποσό π.χ. 0.000001 ο k-means τερματίζει, διαφορετικά οι υπολογισμοί συνεχίζονται από το Βήμα 2.

Εργασία 1
Μελετήστε τον αλγόριθμο και αποφασίστε για ποιά είναι οι απαραίτητοι πίνακες που θα πρέπει να χρησιμοποιήσετε έτσι ώστε να αποθηκεύσετε την πληροφορία που χρειάζεται.
Στην συνέχεια αποφασίστε ποιές θα είναι οι βασικές συναρτήσεις που θα εκτελέσουν τους υπολογισμούς που χρειάζονται.

Εργασία 2
Υλοποιήστε το περιεχόμενο των συναρτήσεων και προσθέστε ότι επιπρόσθετη μνήμη ή συναρτήσεις χρειάζονται. Προσοχή να είσαστε φειδωλοί στην μνήμη και το μέγεθος του κώδικα που κατασκευάζετε.

Εργασία 3
Να χρησιμοποιήσετε τον debugger για να διορθώσετε τυχόν λάθη στον αρχικό κώδικα που φτιάξατε. Για κάθε αλλαγή που κάνετε να φτιάχνετε νεα έκδοση του προγράμματος. Προσοχή να δώσετε μικρές αριθμητικές τιμές στις διαστάσεις των πινάκων που θα χρησιμοποιήσετε έτσι ώστε να μπορείτε να βλέπετε μέσα από τον dubugger τις τιμές τους. Ετσι θα εντοπίσετε και ευκολότερα τα σφάλματά σας.

Εργασία 4
Να βάλετε ρεαλιστικές τιμές στις διαστάσεις των πινάκων σας
#define N 100000
#define Nv 1000
#define Nc 100
#define THR_KMEANS 0.000001
και να χρησιμοποιήσετε τον profiler του gcc (gprof) για να δείτε ποιές συναρτήσεις καταναλώνουν τον περισσότερο χρόνο. Μην βάλετε οδηγίες βελτιστοποίησης στον compiler.
Προσοχή το πρόγραμμα ενδεχωμένως να αργήσει να τελειώσει τους υπολογισμούς γρήγορα (ανάλογα με την υλοποίηση). Συνήθως σε συνήθη μηχανήματα i5,i7 της Intel ή ισοδύναμα της AMD να χρειάζεται 5-10 λεπτά.

Εργασία 5
Με βάσει τους υπολογισμούς του profiler προσπαθήσετε να βελτιώσετε τον χρόνο απόκρισης του προγράμματός σας κάνοντας αλλαγές και χρησιμοποιώντας τον οδηγό βελτιστοποίησης -O2.

Εργασία 6
Προσπαθείστε να εκμεταλευτείτε τις δυνατότητες παράλληλης επεξεργασίας SIMD (vector processing), για την αρχιτεκτονική του επεξεργαστή που έχετε διαθέσιμο στα πλαίσια του μαθήματος μέσω διαδικτύου. Δείτε τα χαρακτηριστικά του (cat */proc/*cpuinfo) και προσθέστε στον κώδικά σας τις παραμέτρους

```
// ************************************************************
#pragma GCC optimize("O3","unroll-loops","omit-frame-pointer","inline", "unsafe-math-optimizations")
#pragma GCC option("arch=native","tune=native","no-zero-upper")
//***********************************************************
```

Να μεταγλωττίσετε το πρόγραμμά σας ως εξής:

```
gcc myprogram.c -o myprogram -fopt-info-vec-optimized
```

για να σας πληροφορήσει ο μεταγλωττιστης ποιούς βρόχους παραλληλοποίησε. Εκτελέστε το πρόγραμμα για να δείτε τι βελτιώσεις στους χρόνους υπολογισμού πετύχατε.

Για να θεωρήσετε ότι έχετε κάνει καλή δουλειά θα πρέπει σε μηχανήματα με υπολογιστική ισχύ bogoMips >= 4000 να έχετε για 16 βήματα του αλγόριθμου k-means απόκριση μικρότερη του 1 min.
Αριστη δουλειά ~30 secs.

Επιτέλους τελειώσατε....

ΠΑΡΑΔΟΤΕΑ

- Για την εργασία 1 στείλτε τον κώδικα σε c ή c++ στον οποίο θα έχετε ορίσει την απαραίτητη μνήμη και τις βασικές συναρτήσεις του προγράμματος που θα υλοποιήσετε. Να βάλετε σχόλια!!!
- Για την εργασία 2 στείλτε τον κώδικα σε c ή c++ υλοποιήσατε. Να βάλετε σχόλια!!!
- Για την εργασία 3 να στείλτε τις διαφορετικές εκδόσεις του προγράμματός σας. Να βάλετε σχόλια!!!
- Για την εργασία 4. Να στείλετε την αναφορά του profiler για τους χρόνους που καταναλώνουν οι συναρτήσεις σας.
- Για την εργασία 5 στείλτε τον κώδικα σε c ή c++ που υλοποιήσετε. Να βάλετε σχόλια εκεί που κάνατε τις αλλαγές και στείλτε τους παλιούς (πριν τις αλλαγές) και τους νέους χρόνους εκτέλεσης προγράμματος (μετά τις αλλαγές). Σχολιάστε την προσπάθειά σας.
- Για την εργασία 6 στείλτε τον κώδικα σε c ή c++ που υλοποιήσετε. Να βάλετε σχόλια εκεί που κάνατε τις αλλαγές και στείλτε τους παλιούς (πριν τις αλλαγές) και τους νέους χρόνους εκτέλεσης προγράμματος (μετά τις αλλαγές). Σχολιάστε την προσπάθειά σας.