Κ23α - Ανάπτυξη Λογισμικού Για Πληροφοριακά Συστήματα

Χειμερινό Εξάμηνο 2024 2025

Άσκηση 3 – Παράδοση: Κυριακή 12 Ιανουαρίου 2025

Στο τρίτο μέρος της εργασίας θα ασχοληθούμε με την βελτιστοποίηση της απόδοσης του κώδικα που δημιουργήθηκε στα προηγούμενα τμήματα της άσκησης. Αυτό θα επιτευχθεί τόσο με την χρήση παραλληλίας όσο και με τη βελτιστοποίηση επιμέρους λειτουργιών.

Βελτιστοποιήσεις

Στο 3ο παραδοτέο μπορείτε να δοκιμάσετε και να υλοποιήσετε βελτιστοποιήσεις που μειώνουν τον χρόνο εκτέλεσης ή/και τη μνήμη που καταναλώνεται, τον χώρο αποθήκευσης που απαιτείται και εν γένει βελτιώνουν την απόδοση σε σχέση με την αρχική υλοποίηση. Έχετε τη δυνατότητα να υλοποιήσετε δικές σας ιδέες ή/και ιδέες που βρίσκετε στη βιβλιογραφία. Σημαντική παράμετρος είναι η μέτρηση της αποτελεσματικότητας των σχετικών τροποποιήσεων.
Ορισμένες από τις προτεινόμενες βελτιστοποιήσεις περιγράφονται παρακάτω.

Αρχικοποίηση γράφων με τυχαίες ακμές

Στους αλγορίθμους FilteredVamana και StitchedVamana περιγράφεται διαδικασία με αρχικά κενό γράφο, στον οποίο σταδιακά προστίθενται κορυφές και ακμές. Στην περίπτωση του dataset που περιέχει ένα μόνο φίλτρο, θα δημιουργηθούν ανεξάρτητοι υπογράφοι. Δοκιμάστε να αρχικοποιήσετε τους γράφους με τυχαίες ακμές μεταξύ κορυφών και παρατηρήστε αν αυτό οδηγεί σε διασύνδεση των ανεξαρτητων υπογράφων (και αντίστοιχα στην απόδοση στα unfiltered ερωτήματα).

Αρχικοποίηση medoid στον Vamana με τυχαία σημεία

Ο αρχικός αλγόριθμος Vamana υπολόγιζε στην αρχή το medoid του συνόλου των σημείων, το οποίο θα είναι και το σημείο από το οποίο ξεκινούν όλα τα ερωτήματα. Δοκιμάστε να αντικαταστήσετε αυτό το βήμα με την επιλογή ενός τυχαίου σημείου. Παρατηρήστε την απόδοση του αλγορίθμου (τόσο σε χρόνο δημιουργίας όσο και σε χρόνο αναζήτησης).

Μία ενδιάμεση κατάσταση θα ήταν να επιλεγεί ένα τυχαίο υποσύνολο σημείων και να υπολογιστεί το medoid σε αυτό το υποσύνολο.

Παράλληλη Εκτέλεση

Η παράλληλη εκτέλεση μπορεί να πραγματοποιηθεί είτε χρησιμοποιώντας pthreads και τους αντίστοιχους μηχανισμούς συγχρονισμού (mutexes, condition variables, barriers, semaphores, κλπ.) είτε

χρησιμοποιώντας ένα πλαίσιο υψηλότερου επιπέδου (π.χ. std::thread στη C++, OpenMP). Μπορείτε να επιλέξετε με ποιον τρόπο θα υλοποιήσετε την παράλληλη εκτέλεση.

Παραλληλοποίηση

Η παραλληλοποίηση του προγράμματός, μπορεί να γίνει σε πολλά επίπεδα. Επισημαίνονται κάποια από τα jobs που μπορεί να εκτελεστούν:

- Υπολογισμός αποστάσεων (και πιθανή προσωρινή αποθήκευσή τους)
- Υπολογισμός ανεξάρτητων υπογράφων (σε StitchedVamana, αλλά και σε FilteredVamana αν δεν υπάρχουν ακμές μεταξύ των υπογράφων)

Σημαντικό τμήμα του σχεδιασμού είναι το μέγεθος της κάθε εργασίας, ώστε να είναι αρκετά μεγάλη για να μην επιβαρύνεται από την ακριβή έναρξη/τερματισμό των εργασιών/threads, αλλά και τον χρονοπρογραμματισμό τους και αρκετά μικρή ώστε να μπορεί να διαμοιράζεται το επεξεργαστικό φορτίο αποδοτικά σε διαφορετικούς επεξεργαστές, χωρίς να πρέπει να μένουν αναξιοποίητοι κάποιοι για λόγους συγχρονισμού.

Τελική Αναφορά

Στη τελική αναφορά θα παρουσιάσετε μια σύνοψη ολόκληρη της εφαρμογής που υλοποιήθηκε. Μπορείτε να αναφέρετε πράγματα που παρατηρήσατε κατά την μοντελοποίηση/υλοποίηση της εφαρμογής σας, με αποτέλεσμα να σας οδηγήσουν σε συγκεκριμένες σχεδιαστικές επιλογές που βελτίωσαν την εφαρμογή σας σε επίπεδο χρόνου, μνήμης, κτλ.
Στην αναφορά πρέπει ακόμη να παρουσιάσετε ένα σύνολο από πειράματα, τα οποία θα δείχνουν το χρόνο εκτέλεσης για τις επιλογές των δομών που θα επιλέξετε. Επειδή δεν μπορείτε να κάνετε πειράματα που να δείχνουν την επίδραση κάθε παραμέτρου, θα πρέπει να επιλέξετε εσείς ποια παράμετρο θα παρουσιάζετε κάθε φορά.. Για παράδειγμα μπορείτε να αναφέρετε ή/και να παρουσιάσετε με διαγράμματα τον χρόνο εκτέλεσης, κατανάλωση μνήμης, παράλληλης εκτέλεσης κ.α.
Τέλος, είναι απαραίτητο να παρουσιάσετε τις δομές που σας έδωσαν τους καλύτερους χρόνους εκτέλεσης για τα datasets και τον τελικό χρόνο/μνήμη, μαζί με τις προδιαγραφές του μηχανήματος που εκτελέσατε τα πειράματα. Η αναφορά δεν πρέπει να ξεπερνά τις 20 σελίδες.

Παράδοση εργασίας

Η εργασία είναι ομαδική, **2 ή 3 ατόμων**.
Γλώσσα υλοποίησης: C / C++
Περιβάλλον υλοποίησης: Linux (gcc >= 9.4+).

Παραδοτέα: Η παράδοση της εργασίας θα γίνει με βάση το τελευταιο commit πριν την προθεσμία υποβολής στο git repository σας. **Η χρήση git είναι υποχρεωτική**.
Στο αρχείο README.md θα αναφέρονται τα εξής:

- Ονοματεπώνυμα και ΑΜ των μελών της ομάδας

- Αναφορά στο ποιο μέλος της ομάδας ασχολήθηκε με ποιο αντικείμενο

Επιπλέον, εκτός από τον πηγαίο κώδικα, θα παραδώσετε μια σύντομη αναφορά, με τις σχεδιαστικές σας επιλογές καθώς και να εφαρμόσετε ελέγχους ως προς την ορθότητα του λογισμικού με τη χρήση ανάλογων βιβλιοθηκών (Software testing). Η ορθότητα τυχόν μεταβολών θα ελέγχεται με αυτοματοποιημένο τρόπο σε κάθε commit/push μέσω github actions.

Προδιαγραφές κώδικα: σύμφωνα με τον προγραμματιστικό διαγωνισμό SIGMOD 2024
https://transactional.blog/sigmod-contest/2024 (περιέχει reference implementation, datasets)

Αναφορές

1. Suhas Jayaram Subramanya, Devvrit, Rohan Kadekodi, Ravishankar Krishaswamy, and Harsha Vardhan Simhadri. 2019. DiskANN: fast accurate billion-point nearest neighbor search on a single node. Proceedings of the 33rd International Conference on Neural Information Processing Systems. Curran Associates Inc., Red Hook, NY, USA, Article 1233, 13766–13776. URL https://dl.acm.org/doi/abs/10.5555/3454287.3455520
1. Siddharth Gollapudi, Neel Karia, Varun Sivashankar, Ravishankar Krishnaswamy, Nikit Begwani, Swapnil Raz, Yiyong Lin, Yin Zhang, Neelam Mahapatro, Premkumar Srinivasan, Amit Singh, and Harsha Vardhan Simhadri. 2023. Filtered-DiskANN: Graph Algorithms for Approximate Nearest Neighbor Search with Filters. In Proceedings of the ACM Web Conference 2023 (WWW '23). Association for Computing Machinery, New York, NY, USA, 3406–3416. https://doi.org/10.1145/3543507.3583552