Ελευθεριάδης Χαράλαμπος Παράλληλα και Διανεμημένα ΑΕΜ: 9257 | 6936981498
elefchar@ece.auth.gr

k-NN – Serial and MPI Implementation in C

Github Link: https://github.com/LambisElef/kNN_MPI

Περιγραφή του προβλήματος:

Ζητούμενο της εργασίας ήταν η δημιουργία ενός αλγορίθμου ο οποίος για κάθε σημείο ενός συνόλου X βρίσκει τους k πλησιέστερους γείτονες (σημεία) και τις αντίστοιχες αποστάσεις στο σύνολο αυτό.

Σημαντικό: Η υλοποίησή μου είναι της λογικής **Row Major**.

Σειριακή υλοποίηση:

Η συνάρτηση kNN() βρίσκει τους k πλησιέστερους γείτονες (σημεία) που ανήκουν σε ένα σύνολο X και τις αντίστοιχες αποστάσεις για κάθε σημείο του συνόλου Y. Υπάρχουν αναλυτικά σχόλια μέσα στον κώδικα που επεξηγούν επακριβώς τη συνάρτηση.

Για την ταξινόμηση των πλησιέστερων αποστάσεων χρησιμοποιήθηκε μια συνάρτηση quickSort() με τις βοηθητικές της partition() και swap().

MPI-σύγχρονη υλοποίηση:

Η συνάρτηση distrAllkNN() καλείται από όλες τις MPI διαδικασίες με διαφορετικό κομμάτι της διαμέρισης X του ολικού συνόλου και φροντίζει για την εύρεση των πλησιέστερων γειτόνων από όλα τα υποσύνολα X' που έχουν διαμοιραστεί στις MPI διαδικασίες και των αντίστοιχων αποστάσεων για κάθε σημείο του αρχικού συνόλου της X. Υπάρχουν αναλυτικά σχόλια μέσα στον κώδικα που επεξηγούν επακριβώς τη συνάρτηση.

Αυτό που αξίζει να σημειωθεί είναι ο τρόπος επικοινωνίας μεταξύ των MPI διεργασιών για ανταλλαγή των σημείων τους και υπολογισμό όλων των αποστάσεων. Οι διεργασίες οργανώνονται σε ένα «δαχτυλίδι». Έτσι μετά από κάθε υπολογισμό των αποστάσεων μεταξύ του αρχικού συνόλου X και ενός άλλου υποσυνόλου X' που δέχτηκαν από την ακριβώς προηγούμενη διεργασία, καλούνται να δώσουν το σύνολο σημείων X' που προηγουμένως έλαβαν στην αμέσως επόμενη διεργασία και να παραλάβουν ένα καινούργιο σύνολο σημείων από την αμέσως προηγούμενη διαδικασία. Ο υπολογισμός τελειώνει μετά από n-1 ανταλλαγές σημείων, δηλαδή όταν όλα τα σύνολα σημείων περάσουν από όλες τις MPI διεργασίες.

Επειδή εδώ χρησιμοποιείται σύγχρονη επικοινωνία με τις συναρτήσεις MPI_Send() και MPI_Recv(), μια διεργασία δε γίνεται να στέλνει και να λαμβάνει σημεία ταυτόχρονα. Έτσι, κάθε ανταλλαγή έχει δύο φάσεις: στην πρώτη φάση οι διεργασίες με ζυγό αριθμό ταυτότητας στέλνουν σημεία και οι διεργασίες με μονό αριθμό ταυτότητας λαμβάνουν, ενώ στην επόμενη φάση γίνεται το ανάποδο.

MPI-ασύγχρονη υλοποίηση:

Εδώ η συνάρτηση distrAllkNN() λειτουργεί με τον ίδιο τρόπο όπως η αντίστοιχη στη σύγχρονη υλοποίηση, με μόνη διαφορά το κομμάτι της επικοινωνίας μεταξύ των διεργασιών για ανταλλαγή των συνόλων σημείων τους. Εδώ χρησιμοποιείται ασύγχρονη επικοινωνία με τις συναρτήσεις

MPI_Isend() και MPI_IRecv(). Έτσι μια διεργασία μπορεί να στέλνει και να λαμβάνει πληροφορίες, καθώς και να συνεχίζει τους υπολογισμούς της. Έτσι, πριν από κάθε νέο υπολογισμό οι διεργασίες MPI φροντίζουν να έχουν ήδη αρχίσει την ανταλλαγή των απαραίτητων νέων σημείων πριν την εκκίνηση του προηγούμενο υπολογισμού. Σε περίπτωση που δεν έχουν προλάβει, αναγκάζονται να περιμένουν μέχρι να ολοκληρωθεί η προηγούμενη ανταλλαγή. Στόχος είναι να «κρυφτεί» το κόστος της επικοινωνίας.

Compile:

Για να κάνετε compile τον κώδικα ανοίγετε ένα terminal στον φάκελο που κατεβάσατε από το github link στο τέλος της αναφοράς και πληκτρολογείτε την εντολή:

- gcc knnring_sequential.c tester.c -O3 -lm -lopenblas -o knnring_sequential για τη σειριακή υλοποίηση, την
- mpicc knnring_sequential.c knnring_mpi_sync.c tester_mpi.c -O3 -lm -lopenblas -o knnring_mpi_sync για την MPI_σύγχρονη υλοποίηση και την
- mpicc knnring_sequential.c knnring_mpi_async.c tester_mpi.c -O3 -lm -lopenblas -o knnring_mpi_async για την MPI_ασύγχρονη υλοποίηση.

Επικύρωση ορθότητας:

Η εργασία **τρέχει κανονικά στους testers (προ 28/11 και τωρινούς)** τοπικά. Στο elearning τρέχει κανονικά το knnring_sequential κομμάτι, ωστόσο το knnring_mpi εμφανίζει segmentation fault χωρίς να μπορώ να βρω το λόγο που συμβαίνει αυτό.

Αποτελέσματα:

Το HPC IT AUTh ακόμη δεν έχει λύσει το πρόβλημά μου με Illegal Instruction, παρόλο που τους έχω ενημερώσει από το βράδυ της Τετάρτης 27/11. Έχω ψάξει και δοκιμάσει πάρα πολλά πράγματα, ωστόσο μάλλον έχω καταλήξει ότι είναι θέμα του openblas module του hpc/pdlabs. Περιμένω την απάντησή τους.

Αναμενόμενα αποτελέσματα είναι η γρηγορότερη εκτέλεση του knnring_mpi_async, καθώς κρύβει το κόστος της επικοινωνίας κατά τη διάρκεια των υπολογισμών. Θέλω να δοκιμάσω αν χρειάζεται ποτέ αναμονή για τα προηγούμενα δεδομένα και κατά πόσο το 1Gbit δίκτυο του cluster είναι bottleneck. Λογικά λόγω του 1Gbit δε θα φανεί εύκολα διαφορά, το κόστος επικοινωνίας θα είναι ήδη μικρό. Μένει να το επιβεβαιώσω με πραγματικά test και δεδομένα. Ίσως αρχίσουν να φαίνονται διαφορές αν αυξήσω κατά πολύ τα σημεία του X και τις διαστάσεις τους.

Τοπικά μετρήθηκε ο χρόνος επικοινωνίας και υπολογισμών. Ο πρώτος είναι πραγματικά ελάχιστος μπροστά στο δεύτερο, γι' αυτό και απουσιάζουν τα οποιαδήποτε αποτελέσματα.

UPDATE: Επιτέλους λειτουργεί κανονικά το cluster, αυτή τη φορά χρησιμοποιούμε τη βιβλιοθήκη netlib για το cblas. Οι δοκιμές έγιναν με πλησιέστερους-γείτονες-**k=10** και MPI ρυθμίσεις **4 nodes και 1 task ανά node**, έτσι ώστε οι επικοινωνίες να μην είναι εσωτερικές, αλλά να μεσολαβεί το Ethernet δίκτυο. Παρακάτω φαίνονται τα αποτελέσματα:

MPI - Σύγχρονη (δευτερόλεπτα)	Μέσος χρόνος επικοινωνίας (node2node)	Μέσος χρόνος υπολογισμών (ανά node)	Συνολικός χρόνος εκτέλεσης
N = 1000 D = 100	0.1911013	2.462013625	10.482959
N = 1000 D = 1000	0.26162675	4.7287698	19.845083
N = 1000 D = 2000	0.510929917	7.192621125	30.743295
N = 1000 D = 4000	0.871462417	12.7537	54.145562
N = 1000 D = 8000	1.525818417	23.16828188	98.099335
N = 1000 D = 10000	1.967815417	28.37235063	120.463613
N = 2000 D = 100	0.8259524	21.15492	87.937210
N = 2000 D = 1000	0.980878333	30.15249625	124.344328
N = 2000 D = 2000	1.429896917	40.04981938	165.446701

MPI - Ασύγχρονη (δευτερόλεπτα)	Μέσος χρόνος καθυστέρησης λόγω επικοινωνίας (node2node)	Μέσος χρόνος υπολογισμών (ανά node)	Συνολικός χρόνος εκτέλεσης
N = 1000 D = 100	0.09245075	2.527258188	10.591457
N = 1000 D = 1000	0.164443333	4.752693625	19.713571
N = 1000 D = 2000	0.267662167	7.208123313	30.052960
N = 1000 D = 4000	0.62166375	12.75198188	53.387306
N = 1000 D = 8000	0.97841175	23.19954938	96.641847
N = 1000 D = 10000	1.124576167	28.32866813	118.023109
N = 2000 D = 100	0.265974417	21.22394625	86.249514
N = 2000 D = 1000	0.664569333	30.29355813	124.122396
N = 2000 D = 2000	1.179834583	39.96056625	164.766493

Συμπέρασμα: Οι χρόνοι υπολογισμών στη σύγχρονη και ασύγχρονη υλοποίηση είναι σχεδόν ίδιοι, όπως αναμενόταν. Επίσης, από τα δεδομένα είναι ξεκάθαρο ότι **ο μέσος χρόνος καθυστέρησης λόγω επικοινωνίας είναι χαμηλότερος στην ασύγχρονη υλοποίηση του MPI**. Τα κέρδη στο συνολικό χρόνο εκτέλεσης είναι εμφανή μόνο για μεγάλες διαστάσεις, όμως και πάλι λόγω των εκτενών υπολογισμών και των σχετικά λίγων δεδομένων που απαιτούνται, το κέρδος δε γίνεται ποτέ μεγαλύτερο από 3% επί του συνολικού χρόνου εκτέλεσης του προγράμματος.