21/6/2020

Γιώργος Ντάκος(1059569), Παναγιώτης Συριόπουλος(1059664), Χρήστος Μπουλαφέντης(1059612), Μάριος Παπαβασιλείου(1059649)

ΟΜΆΔΑ 39

**ΠΑΡΑΛΛΗΛΗ ΕΠΕΞΕΡΓΑΣΙΑ**

ΑΝΑΦΟΡΆ PROJECT

ΑΝΑΦΟΡΆ

STATIC:

#pragma omp parallel num\_threads(n) shared(vertices,arcs,sorted, graph, size,vertices\_size,arcs\_size) private(id, threads, a, v):

Σε αυτό το μπλοκ κώδικα έχουμε δηλώσει ποιες μεταβλητές θα είναι κοινόχρηστες και θα προσπελαστούν από όλα τα νήματα και είναι αυτές που βρίσκονται εντός της εντολής shared. Από την άλλη πλευρά οι μεταβλητές που βρίσκονται μέσα στη private μπορούν να προσπελαστούν από την στοίβα του κάθε νήματος ιδιωτικά. Τώρα με την εντολή #pragma omp parallel δημιουργούμε το μπλοκ κώδικα που θέλουμε να παραλληλοποιήσουμε.

#pragma omp for schedule(static):

Με την εντολή #pragma omp for δεν ξεκινάει κάποιο νήμα αφού έχουν ξεκινήσει είδη από την #pragma omp parallel.Αυτό που κάνει είναι να χρησιμοποιεί οποία νήματα έχουν ήδη διακλαδιστεί από το block parallel στο οποίο περικλείεται. Δεν υπάρχει λόγος για #pragma omp barrier αφού είναι υπονοούμενο στο τέλος του βρόγχου και αν βάζαμε κάποιο barrier θα είχαμε σαν αποτέλεσμα πιο αργή εκτέλεση του αλγορίθμου. Όπως βλέπουμε έχουμε ορίσει τον χρονοπρογραμματισμό του βρόχου με τον στατικό τρόπο [schedule(static)].Με αυτό τον τρόπο το σύστημα αναθέτει εκ περιτροπής σε κάθε νήμα μερίδια επαναλήψεων που περιλαμβάνουν chunksize επαναλήψεις. Εμείς έχουμε παραλείψει το chunksize οπότε η τιμή του ορίζεται αυτόματα από το σύστημα σύμφωνα με το κλάσμα (συνολικό πλήθος/πλήθος νημάτων)

#pragma omp atomic

Με αυτή την εντολή προστατεύουμε κρίσιμα τμήματα που αποτελούνται από μια μοναδική εντολή ανάθεσης τιμής στο πρόγραμμα μας. Φυσικά θα μπορούσαμε να χρησιμοποιήσουμε την critical αλλά με την atomic στις συγκεκριμένες εντολές του προγράμματος έχουμε καλύτερες αποδόσεις από ότι θα είχαμε με critical.

DYNAMIC:

Τα κομμάτια που παραλληλοποιούμε και που ορίζουμε ως κρίσιμα σημεία είναι τα ιδιά με αυτά που έχουμε ορίσει και στον κώδικα με χρονοδιάγραμμα static. Το μόνο που αλλάζει τώρα σε αυτή την περίπτωση είναι ότι αντί για static χρονοπρογραμματισμό του βρόγχου χρησιμοποιούμε dynamic χρονοπρογραμματισμό. Σε ένα δυναμικό χρονοδιάγραμμα οι επαναλήψεις χωρίζονται πάλι σε μερίδια μεγέθους chunksize διαδοχικών επαναλήψεων. Κάθε νήμα εκτελεί ένα τέτοιο μερίδιο και αφού το ολοκληρώσει, ζητάει να του ανατεθεί άλλο ένα από το σύστημα χρόνου εκτέλεσης. Τώρα ανάλογα με το πλήθος τον κόμβων ορίζουμε και το chunksize που θα ανατίθεται σε κάθε νήμα. Όπως παρατηρήσαμε για διάφορες τιμές του chunksize περνάμε άλλοτε καλυτέρους χρόνους και άλλοτε χειρότερους κάτι το οποίο είναι προφανές.

TASK:

#pragma omp task

Η εντολή task δημιουργεί ένα νέο έργο το οποίο είναι υπεύθυνο για την πρώτη πολύγραμμη του κώδικα.

#pragma omp taskwait

Τώρα η taskwait υποχρεώνει το έργο που δημιούργησε το νέο έργο να περιμένει την ολοκλήρωση του έργου αυτού.

#pragma omp single nowait

Η εντολή single καθορίζει πως το αντίστοιχο μπλοκ κώδικα θα εκτελεστεί από ένα μόνο νήμα της τρέχουσας ομάδας και όχι απαραίτητα από το master νήμα. Τώρα με την εντολή nowait τα νήματα δεν χρειάζονται να περιμένουν στο νοητό φράγμα στο τέλος της δομής single. Άμα αντιμετωπίζαμε θέματα κατά την εκτέλεση του κώδικα σαφώς δεν θα βάζαμε nowait αλλά θα είχαμε μείωση στην απόδοση του αλγορίθμου.

|  |  |  |  |
| --- | --- | --- | --- |
| -Ο0 | 1-Thread | 2-Threads | 4-Threads |
| Static | 0.013679 | 0.0135215 | 0.013958 |
| Dynamic | 0.14198 | 0.13526 | 0.14170 |
| Task | 0.013794 | 0.014003 | 0.014337 |

Πίνακας : Για 2618 ακμές και 227 κόμβους χωρίς βελτιστοποίηση.

|  |  |  |  |
| --- | --- | --- | --- |
| -Ο3 | 1-Thread | 2-Threads | 4-Threads |
| Static | 0.004251 | 0.004406 | 0.005318 |
| Dynamic | 0.004343 | 0.004422 | 0.00457 |
| Task | 0.004460 | 0.004871 | 0.004712 |

Πίνακας 2: Για 2618 ακμές και 227 κόμβους με βελτιστοποίηση.

|  |  |  |  |
| --- | --- | --- | --- |
| -Ο0 | 1-Thread | 2-Threads | 4-Threads |
| Static | 5.090227 | 5.080218 | 5.118972 |
| Dynamic | 5.126633 | 5.109917 | 5.102919 |
| Task | 5.221702 | 5.209572 | 5.201007 |

Πίνακας 3: Για 59593 ακμές και 1098 κόμβους χωρίς βελτιστοποίηση.

|  |  |  |  |
| --- | --- | --- | --- |
| -Ο3 | 1-Thread | 2-Threads | 4-Threads |
| Static | 1.468483 | 1.463482 | 1.472576 |
| Dynamic | 1.467967 | 1.462833 | 1.464514 |
| Task | 1.484815 | 1.472186 | 1.466497 |

Πίνακας 4: Για 59593 ακμές και 1098 κόμβους με βελτιστοποίηση.

|  |  |  |  |
| --- | --- | --- | --- |
| -Ο0 | 1-Thread | 2-Threads | 4-Threads |
| Static | 63.247868 | 63.508591 | 63.780889 |
| Dynamic | 63.194975 | 63.360479 | 63.331730 |
| Task | 64.065698 | 64.152243 | 64.391550 |

Πίνακας 5: Για 214403 ακμές και 2073 κόμβους χωρίς βελτιστοποίηση.

|  |  |  |  |
| --- | --- | --- | --- |
| -Ο3 | 1-Thread | 2-Threads | 4-Threads |
| Static | 17.744845 | 16.764479 | 16.532518 |
| Dynamic | 17.709013 | 17.743910 | 18.062798 |
| Task | 17.714154 | 17.749936 | 17.751725 |

Πίνακας 6 :Για 214403 ακμές και 2073 κόμβους με βελτιστοποίηση.