# Παράλληλα συστήματα - Χειμερινό ‘22

# Εργασία 2/16-2-2022

Ομάδα:

sdi2000064 Κωνσταντίνος Κανελλάκης

sdi2000212 Μιχάλης Χατζησπύρου

Άσκηση 2.3

1. **Εξετάστε αν ο εξωτερικός βρόχος του «κατά γραμμή» αλγορίθμου μπορεί να παραλληλοποιηθεί.**

Όπως φαίνεται στον κατά γραμμή αλγόριθμο το x[row] εξαρτάται άμεσα από το x[col]. Όπως παρατηρούμε στο εσωτερικό for η μεταβλητή col παίρνει τιμές στο διάστημα [row+1,n) το οποίο έχει σαν αποτέλεσμα την εξάρτηση του x[row] από στοιχείο άλλου νήματος, πράγμα που καθιστά αδύνατη την παραλληλοποίηση του εξωτερικού βρόχου.

2. **Εξετάστε αν ο εσωτερικός βρόχος του «κατά γραμμή» αλγορίθμου μπορεί να παραλληλοποιηθεί.**

Για τον εσωτερικό βρόχο παρατηρούμε ότι δεν υπάρχουν εξαρτήσεις μεταξύ των στοιχείων που χωρίζονται μεταξύ των νημάτων και άρα μπορεί να παραλληλοποιηθεί χρησιμοποιώντας reduction για το «x[row] −= A[row][col]*x[col];»

3. **Εξετάστε αν ο (δεύτερος) εξωτερικός βρόχος του «κατά στήλη» αλγορίθμου μπορεί να παραλληλοποιηθεί.**

Όμοια με το 1.

4. **Εξετάστε αν ο εσωτερικός βρόχος του «κατά στήλη» αλγορίθμου μπορεί να παραλληλοποιηθεί.**

Όμοια με το 2.

5. **Γράψτε ένα πρόγραμμα OpenMP για καθέναν από τους βρόχους που θεωρήσατε ότι μπορούν να παραλληλοποιηθούν. Ίσως σας φανεί χρήσιμη η οδηγία single –όταν ένα μπλοκ κώδικα εκτελείται παράλληλα και ένα υπομπλοκ στο εσωτερικό του πρέπει να εκτελεστεί από ένα μόνο νήμα, το υπομπλοκ μπορεί να προσδιοριστεί με μια οδηγία #pragma omp single. Τα νήματα της ομάδας που εκτελεί τον κώδικα θα μείνουν μπλοκαρισμένα στο τέλος του υπο-μπλοκ κώδικα μέχρι να φτάσουν σε αυτό το σημείο όλα τα νήματα.**

Για την υλοποίηση των αλγορίθμων χρησιμοποιήθηκαν αρκετές συναρτήσεις από το ερώτημα της άσκησης 2.2 με openMP. Τα ορίσματα του προγράμματος είναι τα εξής:

- Αριθμός νημάτων
- Διάσταση πίνακα
- Τύπος schedule
- Chunk size

- 0 για να χρησιμοποιηθεί ο αλγόριθμος των στηλών ή 1 για τον αλγόριθμο των γραμμών

Οι συναρτήσεις rows() και cols() υλοποιούν τον αλγόριθμο κατά γραμμή και κατά στήλη αντίστοιχα. Οι παραλληλοποιήσεις των βρόχων έγιναν σύμφωνα με τις απαντήσεις στα ερωτήματα 1 και 2. Είναι επίσης απαραίτητο, να σημειωθεί ότι η αντιστοιχία μεταξύ αριθμών και schedule type γίνεται ως εξής:

```
typedef enum omp_sched_t {
  // schedule kinds
  omp_sched_static = 0x1,
  omp_sched_dynamic = 0x2,
  omp_sched_guided = 0x3,
  omp_sched_auto = 0x4,
} omp_sched_t;
```

Το πρόγραμμα αφού τρέξει μία από τις δύο προαναφερθείσες υλοποιήσεις εμφανίζει τον χρόνο που έκανε για να εκτελεστεί και τερματίζει.

6. **Τροποποιήστε τον παράλληλο βρόχο σας χρησιμοποιώντας τον όρο schedule(runtime) και δοκιμάστε το πρόγραμμα με διάφορα χρονοδιαγράμμα-τα. Αν το άνω τριγωνικό σύστημα έχει 10.000 μεταβλητές, ποιο χρονοδιάγραμμα δίνει τις καλύτερες επιδόσεις;**

Για την παρουσίαση των αποτελεσμάτων χρησιμοποιήθηκαν 2 επιπλέον αρχεία script.sh και table.py. Το script.sh εκτελεί τις δύο υλοποιήσεις (κατά γραμμή και κατά στήλη) προκειμένου να προκύψει ο μέσος χρόνος εκτέλεσης της κάθε υλοποίησης για πίνακα διάστασης 10000 όπως ζητείται στην εκφώνηση, τρέχουμε 4 φορές το κάθε πρόγραμμα και λαμβάνουμε τον μέσο όρο του χρόνου εκτέλεσης. Συγκεκριμένα τρέχουν πειράματα για διαφορετικό αριθμό νημάτων 1,2,4,8 και διαφορετικά schedule type (static, dynamic, auto, guided) και chunk size(1,4,8,100). Ύστερα, εξάγει τα αποτελέσματα σε αρχείο txt το οποίο το λαμβάνει ως είσοδο το script table.py και τα αποθηκεύει σε έναν πίνακα 3 διαστάσεων. Τέλος, επεξεργάζεται τα αποτελέσματα, υπολογίζει ποιος συνδυασμός schedule type και chunk size είναι ο πιο αποδοτικός ανά αριθμό νημάτων και τα αποθηκεύει σε ένα αρχείο excel με τη μορφή πινάκων( παραδείγματα περιλαμβάνονται στον φάκελο της εργασίας).

Για την παρουσίαση των αποτελεσμάτων μορφοποιήσαμε το excel που λαμβάνουμε από το script table.py, προκειμένου να γίνεται πιο εύκολα η σύγκριση των αποτελεσμάτων. Τα μορφοποιημένα αποτελέσματα παραθέτονται παρακάτω:

| | rows | cols | schedule_t | chunk_size |
|---|---|---|---|---|
| Chunk size 1 | | | | |
| | 1 | 0.184774 | 0.489178 | static | 1 |
| | 1 | 0.881929 | 2.200078 | dynamic | 1 |
| | 1 | 0.183596 | 0.485824 | guided | 1 |
| | 1 | 0.184082 | 0.487898 | auto | - |
| Chunk size 1 | | | | |
| | 2 | 0.555703 | 0.850232 | static | 1 |
| | 2 | 1.694123 | 2.883395 | dynamic | 1 |
| | 2 | 0.11728 | 0.29703 | guided | 1 |
| | 2 | 0.116689 | 0.254967 | auto | - |
| Chunk size 1 | | | | |
| | 4 | 0.314115 | 0.493764 | static | 1 |
| | 4 | 1.720876 | 2.083763 | dynamic | 1 |
| | 4 | 0.100308 | 0.165869 | guided | 1 |
| | 4 | 0.084569 | 0.134114 | auto | - |
| Chunk size 1 | | | | |
| | 8 | 0.698149 | 0.828899 | static | 1 |
| | 8 | 2.052352 | 2.299858 | dynamic | 1 |
| | 8 | 0.457712 | 0.40017 | guided | 1 |
| | 8 | 0.449377 | 0.432602 | auto | - |

| | rows | cols | schedule_t | chunk_size |
|---|---|---|---|---|
| Chunk size 4 | | | | |
| | 1 | 0.18318 | 0.497932 | static | 4 |
| | 1 | 0.364346 | 0.872568 | dynamic | 4 |
| | 1 | 0.184899 | 0.497891 | guided | 4 |
| | 1 | 0.183849 | 0.497253 | auto | - |
| Chunk size 4 | | | | |
| | 2 | 0.210958 | 0.44225 | static | 4 |
| | 2 | 0.470109 | 0.834417 | dynamic | 4 |
| | 2 | 0.115541 | 0.29635 | guided | 4 |
| | 2 | 0.116542 | 0.257368 | auto | - |
| Chunk size 4 | | | | |
| | 4 | 0.134177 | 0.257389 | static | 4 |
| | 4 | 0.508505 | 0.68661 | dynamic | 4 |
| | 4 | 0.095338 | 0.154957 | guided | 4 |
| | 4 | 0.084594 | 0.134214 | auto | - |
| Chunk size 4 | | | | |
| | 8 | 0.504316 | 0.569326 | static | 4 |
| | 8 | 0.836819 | 0.904717 | dynamic | 4 |
| | 8 | 0.451516 | 0.393466 | guided | 4 |
| | 8 | 0.478941 | 0.466216 | auto | - |

| | rows | cols | schedule_t | chunk_size |
|---|---|---|---|---|
| Chunk size 8 | | | | |
| | 1 | 0.18447 | 0.502582 | static | 8 |
| | 1 | 0.281035 | 0.714752 | dynamic | 8 |
| | 1 | 0.186023 | 0.502662 | guided | 8 |
| | 1 | 0.184384 | 0.500628 | auto | - |
| Chunk size 8 | | | | |
| | 2 | 0.163797 | 0.364297 | static | 8 |
| | 2 | 0.261208 | 0.61309 | dynamic | 8 |
| | 2 | 0.115235 | 0.301648 | guided | 8 |
| | 2 | 0.11644 | 0.257271 | auto | - |

| | rows | cols | schedule_t | chunk_size |
|---|---|---|---|---|
| Chunk size 100 | | | | |
| | 1 | 0.18378 | 0.501404 | static | 100 |
| | 1 | 0.197291 | 0.517527 | dynamic | 100 |
| | 1 | 0.183533 | 0.503383 | guided | 100 |
| | 1 | 0.184544 | 0.502551 | auto | - |
| Chunk size 100 | | | | |
| | 2 | 0.124533 | 0.270065 | static | 100 |
| | 2 | 0.133012 | 0.292566 | dynamic | 100 |
| | 2 | 0.11292 | 0.291559 | guided | 100 |
| | 2 | 0.116228 | 0.25739 | auto | - |

Chunk size 8

| | | | | |
|---|---|---|---|---|
| 4 | 0.110142 | 0.216248 | static | 8 |
| 4 | 0.257585 | 0.470082 | dynamic | 8 |
| 4 | 0.092276 | 0.15251 | guided | 8 |
| 4 | 0.084487 | 0.134197 | auto | - |

Chunk size 8

| | | | | |
|---|---|---|---|---|
| 8 | 0.472837 | 0.582058 | static | 8 |
| 8 | 0.593429 | 0.688358 | dynamic | 8 |
| 8 | 0.445802 | 0.383825 | guided | 8 |
| 8 | 0.460221 | 0.428156 | auto | - |

Chunk size 100

| | | | | |
|---|---|---|---|---|
| 4 | 0.090947 | 0.143689 | static | 100 |
| 4 | 0.097177 | 0.161641 | dynamic | 100 |
| 4 | 0.082928 | 0.148955 | guided | 100 |
| 4 | 0.084355 | 0.133831 | auto | - |

Chunk size 100

| | | | | |
|---|---|---|---|---|
| 8 | 0.463012 | 0.542252 | static | 100 |
| 8 | 0.447325 | 0.378681 | dynamic | 100 |
| 8 | 0.426673 | 0.366042 | guided | 100 |
| 8 | 0.460796 | 0.44548 | auto | - |

Fastest schedule type and chunk size

| | | | | |
|---|---|---|---|---|
| 1 | 0.18318 | static | 4 | rows |
| 2 | 0.11292 | guided | 100 | rows |
| 4 | 0.082928 | guided | 100 | rows |
| 8 | 0.366042 | guided | 100 | cols |

Ο κώδικας αναπτύχθηκε και δοκιμάστηκε στο υπολογιστικό σύστημα linux. Τα scripts της bash επίσης έτρεξαν στα μηχανήματα λινουξ της σχολής ωστόσο τα python scripts έτρεξαν σε προσωπικό υπολογιστή με έκδοση 3.6.13 όπου εγκαταστάθηκαν και οι απαιτούμενες βιβλιοθήκες. Να σημειωθεί ότι οι πίνακες και τα δεδομένα που παρουσιάζονται πιο πάνω μπορεί να παρουσιάσουν μερικές μικρές αλλοιώσεις σε σχέση με την θεωρία (αυτά που δηλαδή θα περιμέναμε να βγουν) καθώς στο μηχάνημα που τρέχαμε τα προγράμματα συχνά ήταν και άλλοι συνδεδεμένοι. Συγκεκριμένα χρησιμοποιήθηκε το μηχάνημα με αριθμό 25 του οποίου τα χαρακτηρίστηκα είναι τα εξής:

Operating system: VERSION="18.04.6 LTS (Bionic Beaver)"

Architecture: x86_64

CPU op-mode(s): 32-bit, 64-bit

CPU(s): 4

Thread(s) per core: 1

Core(s) per socket: 4

Socket(s): 1

Model name: Intel(R) Core(TM) i5-6500 CPU @ 3.20GHz

Cache line size: 64

Compiler version: gcc version 7.5.0