

Παράλληλος Προγραμματισμός 2018

Προγραμματιστική Εργασία #1

Ονοματεπώνυμο: Ηλιάδης Νικόλαος

ΑΜ: Π2014009

Στον πρώτο matrix1.c έχουμε προσπέλαση του πίνακα κατά γραμμή.

Μέσω την συνάρτησης get_walltime έχουμε υπολογισμό του χρόνου εκτέλεσης της κάθε προσπέλασης.

Έχουμε δυναμική δέσμευση μνήμης μέσω της malloc.

Αρχικά αρχικοποιούμε τον πίνακα με τιμές = 0,0 (.0 γιατί έχουμε πίνακα double)

Έπειτα ξεκινάμε τη εκτέλεση της παραπάνω συνάρτησης ώστε να ξεκινήσει η μέτρηση του χρόνου και εκτελούμε τη λούπα του πίνακα μας `Array[j*NCOLS + i]`. Έπειτα ξανακαλούμε τη συνάρτηση με όρισμα (end) ώστε να τερματιστεί η μέτρηση του χρόνου.

Τέλος έχουμε τύπωση των αποτελεσμάτων. Τόσο του χρόνου που πέρασε όσο και των προσπελάσεων ανά δευτερόλεπτο.

Όσον αφορά το matrix2.c έχουμε τη εκτέλεση του ίδιου ακριβώς κώδικα με τη μόνη διαφορά τώρα ότι κάνουμε προσπέλαση του πίνακα κατά στήλη. Αυτό γίνεται στην λούπα `Array[i*NCOLS + j]`.

Έγιναν μετρήσεις για `NROWS = 100, 1000, 10000, 100000, 1000000` τόσο του χρόνου που πέρασε (tim) όσο και των προσπελάσεων ανά δευτερόλεπτο (Megaccses) κρατώντας σταθερό τον αριθμό των `NCOLS` σε 100 σε κάθε run.

Έγιναν 5 runs σε κάθε εκδοχή του πειράματος και παρακάτω παραθέτω τους πίνακες αποτελεσμάτων.

Πίνακες Αποτελεσμάτων

Κατά στήλη

Calculation for 100 Rows and 100 Columns

Run	Time-elapsed	Megaccses
1 st Try	0.000059	434.724184
2 nd Try	0.000045	200.184547
3 rd Try	0.000034	590.217872
4 th Try	0.000073	270.783685
5 th Try	0.000072	290.637781
Average	0.000044	357.291779

Calculation for 1000 Rows and 100 Columns

Run	Time-elapsed	Megacces
1 st Try	0.000505	458.782217
2 nd Try	0.000408	489.745148
3 rd Try	0.000463	581.256458
4 th Try	0.000789	298.454556
5 th Try	0.000757	541.787521
Average	0.000584	474.005180

Calculation for 10000 Rows and 100 Columns

Run	Time-elapsed	Megacces
1 st Try	0.007053	283.571361
2 nd Try	0.006580	100.456857
3 rd Try	0.008558	201.143859
4 th Try	0.003581	159.482448
5 th Try	0.006868	183.787217
Average	0.006528	185.688348

Calculation for 100000 Rows and 100 Columns

Run	Time-elapsed	Megacces
1 st Try	0.048103	415.628821
2 nd Try	0.082456	318.456455
3 rd Try	0.045454	345.154825
4 th Try	0.038155	348.527457
5 th Try	0.047231	358.872367
Average	0.052279	357.327985

Calculation for 1000000 Rows and 100 Columns

Run	Time-elapsed	Megacces
1 st Try	0.357823	575.628821
2 nd Try	0.313455	587.456878
3 rd Try	0.294545	650.175248
4 th Try	0.328765	580.545689
5 th Try	0.345948	550.578767
Average	0.328107	588.8770806

Κατά γραμμή

Calculation for 100 Rows and 100 Columns

Run	Time-elapsed	Megacces
1 st Try	0.000043	466.505478
2 nd Try	0.000042	468.706944
3 rd Try	0.000109	508.074724
4 th Try	0.000098	405.789703
5 th Try	0.000055	451.312087
Average	0.0000694	460.077787

Calculation for 1000 Rows and 100 Columns

Run	Time-elapsed	Megacces
1 st Try	0.000431	465.000443
2 nd Try	0.000588	454.667100
3 rd Try	0.000671	508.087245
4 th Try	0.000752	499.458721
5 th Try	0.000458	458.087082
Average	0.000580	477.060118

Calculation for 10000 Rows and 100 Columns

Run	Time-elapsed	Megacces
1 st Try	0.004389	455.976953
2 nd Try	0.003508	454.667100
3 rd Try	0.004812	508.087245
4 th Try	0.005822	499.458721
5 th Try	0.006801	458.087082
Average	0.000580	475.255420

Calculation for 100000 Rows and 100 Columns

Run	Time-elapsed	Megacces
1 st Try	0.090303	221.476721
2 nd Try	0.044750	446.927622
3 rd Try	0.052241	305.657822
4 th Try	0.045122	380.784351
5 th Try	0.078254	408.157731
Average	0.062134	352.600849

Calculation for 1000000 Rows and 100 Columns

Run	Time-elapsed	Megacces
1 st Try	0.487823	446.582121
2 nd Try	0.469231	452.878021
3 rd Try	0.398912	402.782241
4 th Try	0.494545	405.034836
5 th Try	0.487324	441.809341
Average	0.467567	429.817312

Συμπεράσματα

Όταν έχουμε ακολουθιακή προσπέλαση του πίνακα έχουμε πιο γρήγορη εκτέλεση καθώς τα αποτελέσματα βρίσκονται στην cache memory διότι όπως γνωρίζουμε η cache memory είναι πολύ πιο γρήγορη από τη main memory.

Όσον αφορά την προσπέλαση κατά γραμμή όταν δηλαδή δεν έχουμε ακολουθιακή προσπέλαση τα data του πίνακα βρίσκονται στην main memory και όχι στην cache για αυτό τον λόγο έχουμε τη εν λόγω καθυστέρηση που παρουσιάστηκε στους παραπάνω πίνακες.