Παράλληλος Προγραμματισμός 2018
Προγραμματιστική Εργασία #1

Ονοματεπώνυμο: Δημήτριος Κατσαρός
ΑΜ: Π2014003

Στην πρώτη εκδοχή του κώδικά μου (matrix1.c) το πρόγραμμα κάνει προσπέλαση του πίνακα table ανά στήλη.
Πιο συγκεκριμένα αφού γίνεται η δυναμική δέσμευση μνήμης κάνω ένα "ζέσταμα" στον επεξεργαστή δίνοντας τιμές στον πίνακα.
Στην συνέχεια αφού καλείται η συνάρτηση getwalltime ξεκινάει και η προσπέλαση του πίνακα.
Η πράξη που γίνεται εκεί γίνεται καθαρά για να μην υπάρξει απαλοιφή των loops.
Οι μεταβλητές που χρησιμοποιώ μέσα στον πίνακα table είναι j*NCOLS+i. Έτσι στην ουσία κάθε φορά που γίνεται η επανάληψη του j ο πίνακας βλέπει την πρώτη τιμή της επόμενης γραμμής.
Αυτό αλλάζει όταν ολοκληρώνεται για πρώτη φορά το loop του j και αυξάνεται το i κατά 1. Τότε ο πίνακας αρχίζει να βλέπει τον δεύτερο κελί από κάθε στήλη και συνεχίζει έτσι μέχρι να τελειώσει και το loop για το i.
Στην συνέχεια καλείται πάλι η getwalltime για να πάρουμε πάλι τον χρόνο έτσι ώστε να δούμε πόσο ακριβώς κράτησε η διαδικασία προσπέλασης.
Στην συνέχεια του κώδικα γίνεται έλεχγος για να δω αν τα στοιχεία του πίνακα είναι έτσι όπως θα έπρεπε να είναι.
Τέλος εμφανίζω και τυπώνω τον χρόνο και τα Maccess στην οθόνη και σε ένα αρχείο result.csv και κάνω clear την μνήμη από τον πίνακα.

Στην δεύτερη εκδοχή του κώδικα (matrix2.c) γίνονται ακριβώς τα ίδια πράγματα με την διαφορά ότι η προσπέλαση του πίνακα γίνεται ανά γραμμή.
Πιο συγκεκριμένα ο compiler προσπελαύνει τα στοιχεία του πίνακα σε σειρά χωρίς να προσπερνάει κανένα στοιχείο.
Γίνεται χρήση των μεταβλητών table[i*NCOLS+j]. Το i φτάνει μέχρι και τον αριθμώ των γραμμών ενώ το j μέχρι τον αριθμό των στηλών.
Έτσι αρχικά i*NCOLS=0 οπότε το j αυξάνεται κατά 1 και ο πίνακας προσπελαύνει τις τιμές του πίνακα κανονικά.
Όταν το i γίνει ένα τότε ο πίνακας θα δείχνει το 1ο στοιχείο της δεύτερης γραμμής ενώ θα έχει ήδη προσπελαστεί η πρώτη γραμμή, και θα συνεχίσει την προσπέλαση με τον ίδιο τρόπο μέχρι να τελειώσει ο πίνακας.

Έτρεξα τους κώδικες για τιμές NROWS=100,1000,10000,100000. Δοκίμασα να τρέξω δέκα φορές την κάθε περίπτωση και να πάρω για σύγκριση την μέση τιμή κάθε περίπτωσης. Παρακάτω παραθέτω τους πίνακες με τα αποτελέσματα.

Πίνακες αποτελεσμάτων

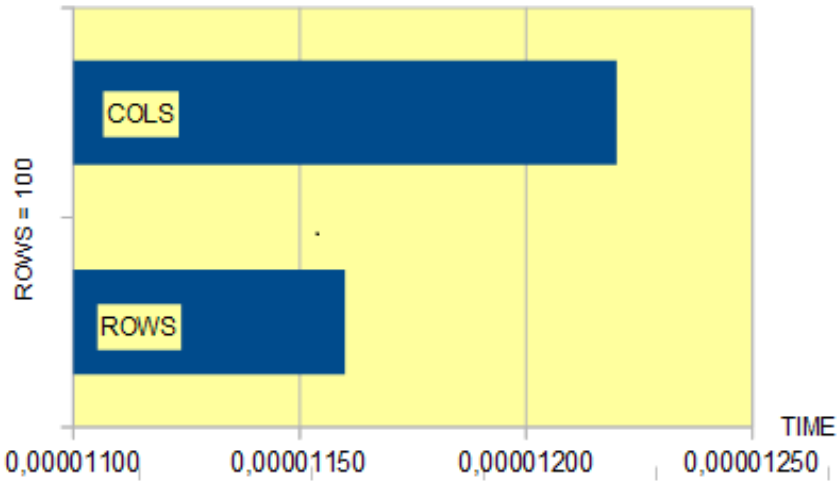
Χρόνοι για την προσπέλαση ανά γραμμή			
Rows = 100	**Rows = 1000**	**Rows = 10000**	**Rows = 100000**
0.000015	0.000224	0.002595	0.01896
0.000017	0.000163	0.002509	0.01719
0.000024	0.000112	0.001645	0.021336
0.000006	0.000198	0.001879	0.017338
0.000006	0.000222	0.001361	0.021109
0.000016	0.000171	0.00145	0.018336
0.000006	0.000107	0.001307	0.019953
0.000013	0.000244	0.002493	0.017232
0.000007	0.000172	0.001755	0.017139
0.000006	0.000245	0.002537	0.019879
Avg Time	**Avg Time**	**Avg Time**	**Avg Time**
0.00001160	0.000186	0.001953	0.018847

Maccess για την προσπέλαση ανά γραμμή			
Rows = 100	**Rows = 1000**	**Rows = 10000**	**Rows = 100000**
1331.525079	893.355485	770.728409	1054.852372
1181.494085	1226.404678	797.09312	1163.468516
838.8608	1784.810213	1215.74029	937.379372
3355.4432	1010.675663	1064.409085	1153.533092
3355.4432	901.032009	1469.365563	947.469194
1252.031045	1169.959275	1379.251562	1090.747006
3355.4432	1868.28686	1530.209413	1002.366887
1553.445926	819.2	802.200249	1160.635342
2796.202667	1163.468516	1139.601685	1166.915854
3226.387692	816.012451	788.328916	1006.081627
Avg Maccess	**Avg Maccess**	**Avg Maccess**	**Avg Maccess**
2224.627689	1165.320515	1095.692829	1068.344926

Χρόνοι για την προσπέλαση ανά στήλη			
Rows = 100	**Rows = 1000**	**Rows = 10000**	**Rows = 100000**
0.000026	0.000443	0.01046	0.102798
0.000025	0.000822	0.009966	0.099441
0.000015	0.000849	0.008958	0.09692
0.000008	0.000244	0.010114	0.097637
0.000007	0.000346	0.009074	0.093624
0.000006	0.000822	0.005963	0.091985
0.00001	0.000627	0.010017	0.096478
0.000006	0.00033	0.009972	0.096491
0.000013	0.000698	0.007602	0.094246
0.000006	0.000731	0.004634	0.096662
Avg Time	**Avg Time**	**Avg Time**	**Avg Time**
0.00001220	0.000591	0.008676	0.096628

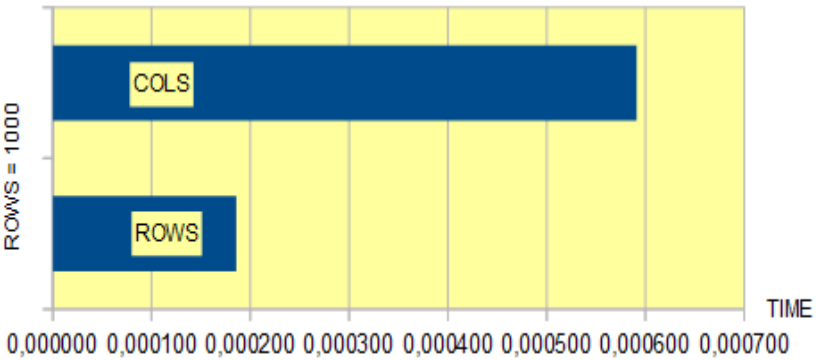
Maccess για την προσπέλαση ανά γραμμή			
Rows = 100	**Rows = 1000**	**Rows = 10000**	**Rows = 100000**
769.597064	451.485899	191.206419	194.556343
798.915048	243.289095	200.679601	201.124181
1331.525079	235.568885	223.261597	206.355729
2542.002424	820.000782	197.746588	204.840508
2796.202667	578.125982	220.410626	213.620177
3355.4432	243.289095	335.410156	217.426784
1997.287619	318.958479	199.662208	207.301178
3355.4432	605.675668	200.559652	207.273518
1525.201455	286.496175	263.089478	212.210799
3355.4432	273.601044	431.601564	206.906445
Avg Maccess	**Avg Maccess**	**Avg Maccess**	**Avg Maccess**
2182.706096	405.649110	322.949297	207.161566

Διαγράμματα

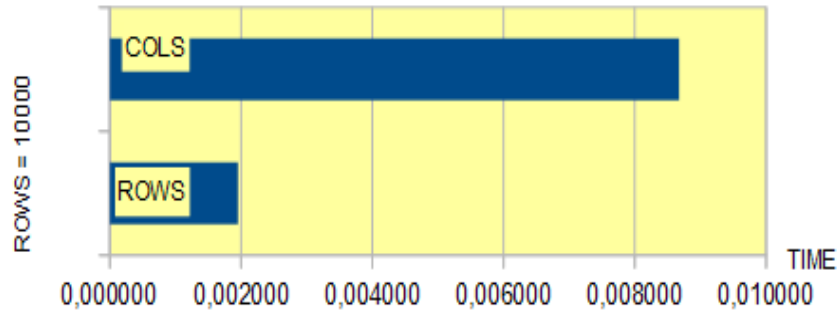


Διάγραμμα μέσου χρόνου μεταξύ προσπέλαση στήλης και γραμμής για ROWS=100

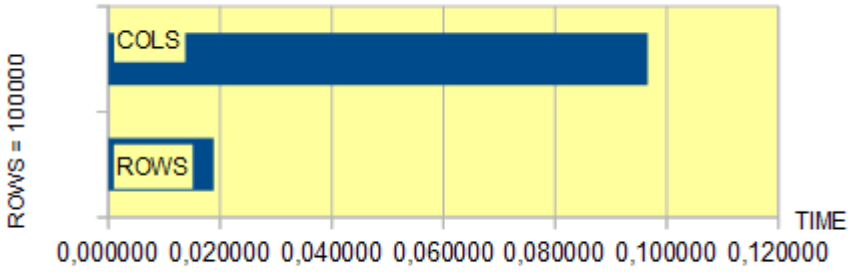
Διάγραμμα μέσου χρόνου μεταξύ προσπέλασης στήλης και γραμμής για ROWS=1.000



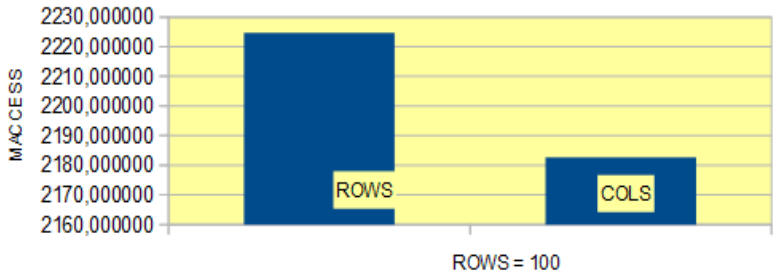
Διάγραμμα μέσου χρόνου μεταξύ προσπέλασης στήλης και γραμμής για ROWS=1.000



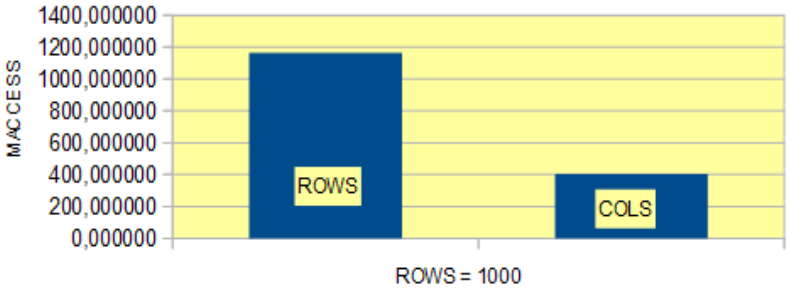
Διάγραμμα μέσου χρόνου μεταξύ προσπέλασης στήλης και γραμμής για ROWS= 10.000



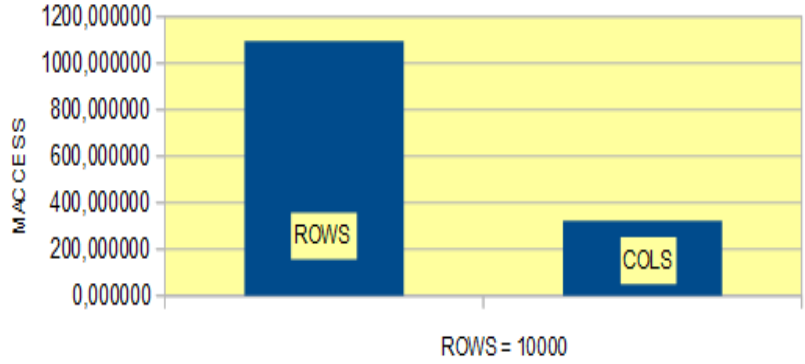
Διάγραμμα μέσου χρόνου μεταξύ προσπέλασης στήλης και γραμμής για ROWS = 100.000



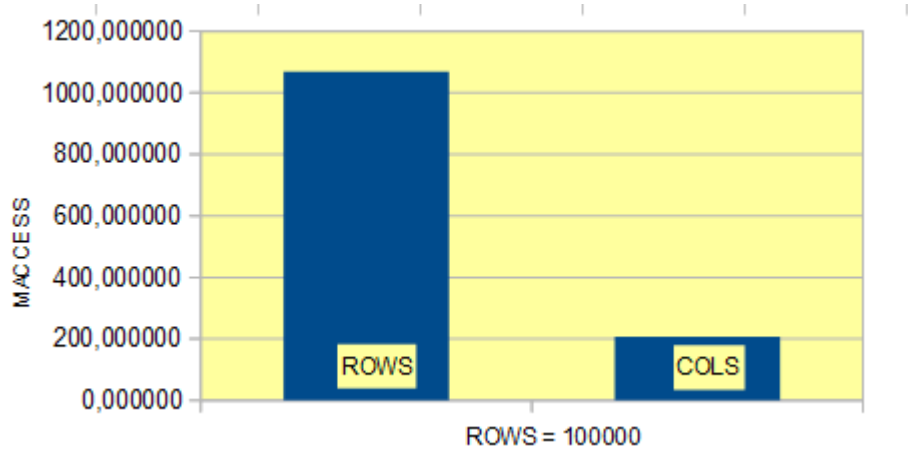
Διάγραμμα μέσης τιμής Maccess μεταξύ προσπέλασης στήλης και γραμμής για ROWS = 100



Διάγραμμα μέσης τιμής Maccess μεταξύ προσπέλασης στήλης και γραμμής για ROWS = 1.000



Διάγραμμα μέσης τιμής Maccess μεταξύ προσπέλασης στήλης και γραμμής για ROWS = 10.000



Διάγραμμα μέσης τιμής Maccess μεταξύ προσπέλασης στήλης και γραμμής για ROWS = 100.000

Συμπέρασμα

Άπο τα παραπάνω διαγράμματα αλλά και τα αποτελέσματα των πινάκων βγαίνουν κάποια συμπεράσματα. Αρχικά όταν μιλάμε για προσπέλαση ενός πίνακα που έχει λίγα στοιχεία η διαφορά στην απόδοση του επεξεργαστή είναι αρκετά μικρή οπότε πολλές φορές μπορεί να είναι και αμελητέα. Όσο αυξάνονται οι γραμμές φαίνεται μια αρκετά μεγάλη διαφορά τόσο στο χρόνο που χρειάζεται ο επεξεργαστής για να προσπελάσει τους 2 πίνακες αλλά και στα Maccess . Αυτό συμβαίνει καθώς η γλώσσα C είναι μια γλώσσα Row – major order . Αυτό σημαίνει πως στην ουσία το πρόγραμμα αποθηκεύει τα στοιχεία του πίνακα σε συνεχόμενες θέσεις μνήμης. Επιπρόσθετα όταν το πρόγραμμα ζητάει μια συγκεκριμένη θέση μνήμης ο επεξεργαστής την φέρνει στην cache αλλά φέρνει και τις κοντινές θέσεις μνήμεις μαζί καθώς “θεωρεί” πως θα τις χριαστεί το πρόγραμμα σύντομα. Όταν τώρα το πρόγραμμα ζητάει της τιμές από τις επόμενες γραμμές ζητάει το αντίθετο από ότι στην ουσία κάνει η C (αποθηκεύει τον πίνακα γραμμή-γραμμή) οπότε όταν μιλάμε για πολλά στοιχεία ο επεξεργαστής αλλάζει τα στοιχεία που έχει φέρει στην Cache από την κύρια μνήμη . Αυτή η διαδικασία απαιτεί αρκετό χρόνο για αυτό και υπάρχει τόση απόκληση μεταξύ των δύο περιπτώσεων. Βέβαια όπως φάνηκε και στα παραδείγματα αυτό συμβαίνει όταν μιλάμε για “μεγάλο” αριθμό στοιχείων (παραπάνω από 100x100).

Πηγές

1. https://en.wikipedia.org/wiki/Row-_and_column-major_order
2. https://stackoverflow.com