

Παράλληλος Προγραμματισμός 2018

Προγραμματιστική Εργασία #1

Μέμου Παρασκευή

Π2013092

Για να εξασφαλιστεί η μη εξάλειψη των loops από το μεταγλωττιστή, χρησιμοποιήσαμε δύο βοηθητικούς πίνακες, rand1 και rand2, τους οποίους αρχικοποιήσαμε με τυχαίες τιμές. Χρησιμοποιήσαμε επίσης τη συνάρτηση srand με όρισμα την τρέχουσα χρονική στιγμή ως seed για να παίρνουμε σε κάθε εκτέλεση πραγματικά τυχαίους αριθμούς. Στη συνέχεια στο loop στον οποίο κάνουμε τις μετρήσεις πραγματοποιούμε μία τυχαία πράξη μεταξύ των δύο βοηθητικών πινάκων και αποθηκεύουμε το αποτέλεσμα στο table. Το πλήθος των γραμμών για το οποίο επιλέξαμε να τρέξουμε τα δοκιμαστικά προγράμματα ήταν: 100, 1.000, 10.000, 50.000, 100.000, ενώ το πλήθος των στηλών παρέμεινε σταθερό στις 100. Η τυχαία πράξη ήταν η ακόλουθη:

$$\text{table}[i,j] = \text{rand1}[i, j] + \text{rand1}[i, j] / \text{rand2}[i, j];$$

Τα i, j είναι ενδεικτικά, αλλά στον κώδικα ο δισδιάστατος πίνακας έχει μία διάσταση με το συνολικό πίνακα να καταλαμβάνει NROWSxNCOLS θέσεις μνήμης. Στο αρχείο matrix1.c η προσπέλαση γίνεται γραμμή προς γραμμή, ενώ στο αρχείο matrix2.c η προσπέλαση γίνεται στήλη προς στήλη. Ο τρόπος με τον οποίο επιτυγχάνεται η προσπέλαση σε κάθε περίπτωση είναι ο χειρισμός των indexes των πινάκων και παρουσιάζεται παρακάτω:

matrix1.c

```
for (i=0;i<NROWS;i++) {  
    for (j=0;j<NCOLS;j++){  
        table[i*NCOLS+j]=rand1[i*NCOLS+j]+rand2[i*NCOLS+j]/rand1[i*  
        NCOLS+j];  
    }  
}
```

matrix2.c

```
for (i=0;i<NROWS;i++) {  
    for (j=0;j<NCOLS;j++){  
        table[i+j*NCOLS]=rand1[i+j*NCOLS]+rand2[i+j*NCOLS]/rand1[i+  
        j*NCOLS];  
    }  
}
```

}

Για να μετρήσουμε την απόδοση για τις δύο περιπτώσεις υπολογίζουμε τα flops/sec (το πλήθος των υπολογισμών ανά δευτερόλεπτο), τα accesses/sec (πλήθος προσβάσεων στη μνήμη ανά δευτερόλεπτο) και το χρόνο που πέρασε για την εκτέλεση του loop. Το ts είναι η χρονική στιγμή ακριβώς πριν την αρχή του loop και te η χρονική στιγμή πριν το τέλος του. Το πλήθος των επαναλήψεων του loop είναι όσο το πλήθος των στοιχείων του πίνακα. Σε κάθε επανάληψη εκτελούνται 2 πράξεις (πρόσθεση και διαίρεση), ενώ έχουμε 4 προσπελάσεις της μνήμης (3 για την ανάκτηση των δεδομένων των 3 πινάκων και 1 για την αποθήκευση του αποτελέσματος). Έτσι προκύπτουν οι παρακάτω υπολογισμοί, ενώ πολλαπλασιάζουμε το χρονικό διάστημα με το 10^6 για να πάρουμε τα αποτελέσματα στην κλίμακα Mflops (και Maccesses).

```
mflops = (NROWS*NCOLS*2.0)/((te-ts)*1e6);
maccesses = (NROWS*NCOLS*4.0)/((te-ts)*1e6);
printf("Time elapsed: %f\n",te-ts);
printf("Mflops/sec: %f\n",mflops);
printf("Maccesses/sec: %f\n",maccesses);
```

Το πλήθος των γραμμών για το οποίο τρέξαμε τα προγράμματα ήταν: 100, 1.000, 10.000, 50.000, 100.000. Τα αποτελέσματα φαίνονται στους παρακάτω πίνακες:

matrix1.c

Γραμμές	Χρόνος (sec)	Mflops/sec	Maccesses/sec
100	0	inf	inf
1000	0.001002	199.586200	399.172401
10000	0.009507	210.367339	420.734678
50000	0.030024	333.066307	666.132613
100000	0.050024	399.807832	799.615663

matrix2.c

Γραμμές	Χρόνος (sec)	Mflops/sec	Maccesses/sec
100	0	inf	inf
1000	0.003004	66.576254	133.152508
10000	0.020025	99.875082	199.750164
50000	0.150069	66.636015	133.272030
100000	0.330167	60.575439	121.150879

Ο υπολογιστής στον οποίο τρέξανε τα προγράμματα διαθέτει τον επεξεργαστή [i7-4510U](#), ο οποίος έχει τις ακόλουθες μνήμες cache.

Level 1 Cache	128 KB
Level 2 Cache	512 KB
Level 3 Cache	4 MB

Πηγή: <https://www.notebookcheck.net/Intel-Core-i7-4510U-Notebook-Processor.115083.0.html>

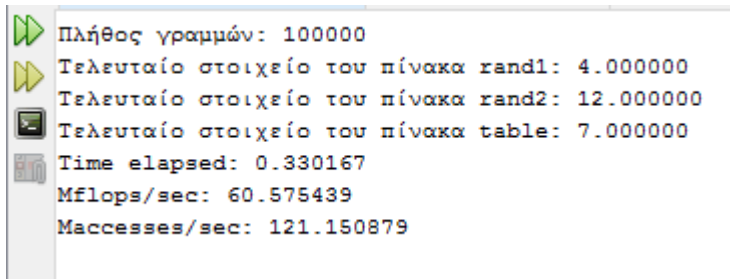
Το μέγεθος του κάθε πίνακα του προγράμματός μας σε κάθε περίπτωση προκύπτει από τον υπολογισμό $NROWS * NCOLS * \text{sizeof}(\text{double})$:

Γραμμές	Μέγεθος ενός πίνακα (MB)	Μέγεθος τριών πινάκων (MB)
100	0.08	0.24
1000	0.800	2.4
10000	8	24
50000	40	120
100000	80	240

Όσο αυξάνεται το πλήθος των γραμμών, τόσο αυξάνεται και ο χρόνος εκτέλεσης, αφού αυξάνεται το πλήθος των δεδομένων για επεξεργασία. Στον υπολογιστή μας στην περίπτωση $NROWS=100$, ο υπολογισμός γίνεται σε πολύ μικρό χρόνο, που δεν είναι ικανός να καταγραφεί. Στο *matrix1*, όσο αυξάνεται το πλήθος των γραμμών, αυξάνονται τα Mflops/sec και τα Maccesses/sec. Στο *matrix2*, τα Mflops/sec και τα Maccesses/sec αυξάνονται μέχρι την τιμή 10.000, αλλά στη συνέχεια παρουσιάζουν φθίνουσα πορεία.

Όταν το πρόγραμμα που εκτελείται ζητά κάτι από τη μνήμη και δεν το βρίσκει στη cache, τότε εκτός των δεδομένων που ζήτησε μεταφέρονται και ομάδες δεδομένων από κοντινές θέσεις μνήμης που κατά κανόνα είναι πολύ πιθανό να ζητηθούν στη συνέχεια του προγράμματος. Στο *matrix1* η προσπέλαση γίνεται γραμμή-προς-γραμμή που σημαίνει ότι το πρόγραμμα ζητά δεδομένα από συνεχόμενες θέσεις μνήμης, τα οποία μεταφέρονται κατά ομάδες στη μνήμη cache του μηχανήματος. Αντίθετα, στο *matrix2*, που η προσπέλαση γίνεται στήλη-προς-στήλη, απαιτείται προσπέλαση θέσεων μνήμης που απέχουν μεταξύ τους κατά $NCOLS=100$ θέσεις, οπότε υπάρχουν περισσότερες αστοχίες στη μνήμη cache. Αυτός είναι ο λόγος που ο χρόνος είναι μεγαλύτερος στο *matrix2* από ότι στο *matrix1* σε όλες τις περιπτώσεις, ενώ όταν το μέγεθος του κάθε πίνακα φτάνει τα 40MB υπάρχουν πλέον πολλές αστοχίες λόγω της απόστασης που έχουν τα δεδομένα μεταξύ τους στην περίπτωση *matrix2*. Αυτό εξηγεί την πτώση των Mflops/sec και Maccesses/sec στις δύο τελευταίες μετρήσεις στο *matrix2*. Αντίθετα, στο *matrix1* οι δείκτες αυτοί συνεχίζουν να αυξάνονται καθώς το γεγονός ότι το πρόγραμμα ζητά δεδομένα από συνεχείς θέσεις μνήμης βοηθά στο να γίνονται περισσότεροι υπολογισμοί ανά δευτερόλεπτο, καθώς τα περισσότερα δεδομένα τα βρίσκει στη μνήμη cache.

Τα δοκιμαστικά προγράμματα αναπτύχθηκαν στο Netbeans 8.1 σε περιβάλλον Windows 10. Το πλήθος των γραμμών ορίστηκε μέσω ορίσματος στο μεταγλωττιστή. Στην παρακάτω εικόνα φαίνεται ενδεικτικά η έξοδος του προγράμματος στην περίπτωση matrix2 για 100.000 γραμμές.



```
Πλήθος γραμμών: 100000
Τελευταίο στοιχείο του πίνακα rand1: 4.000000
Τελευταίο στοιχείο του πίνακα rand2: 12.000000
Τελευταίο στοιχείο του πίνακα table: 7.000000
Time elapsed: 0.330167
Mflops/sec: 60.575439
Maccesses/sec: 121.150879
```