Παράλληλος προγραμματισμός προγραμματιστική εργασία -1-

Σπυρίδων Θεοδωρόπουλος

Π2015035

1.0 Επεξήγηση εργασίας

Σκοπός της εργασίας ήταν η χρήση και η εκμάθηση των εντολών των εντολών ss2 με σκοπό την αύξηση της απόδοσης ενός απλού προγράμματος . Στην ουσία το πρόγραμμα τόσο στην απλή όσο και στην εξειδικευμένη μορφή πολλαπλασιάζει δυο δισδιάστατους πίνακες και στην συνέχεια αποθηκεύει τα αποτελέσματα σε έναν τρίτο.

Στο αρχείο matmul-normal.c βρίσκεται η απλή μορφή του κώδικα και στο αρχείο matmul-sse.c η μορφή με την χρήση των εντολών ss2 .

2.0 Περιγραφή κώδικα

Αρχικά για τις ανάγκες της εργασίας χρησιμοποιούμε μια συνάρτηση την get_walltime η οποία μας βοηθάει να υπολογίσουμε τον χρόνο εκτέλεσης των εντολών.

Όσο αναφορά και τους δύο κώδικες η ροή του προγράμματος είναι η εξής. Αρχικά γίνεται δήλωση και η δέσμευση των πινάκων καθώς και ο έλεγχος για αν έγινε σωστά η δέσμευση .Έπειτα αφού δηλώσουμε τις απαραίτητες μεταβλητές καλούμε την get_walltime έτσι ώστε να αρχίσει την χρονομέτρηση και στην συνέχεια εκτελούμε τους πολλαπλασιασμούς των πινάκων με την χρήση βρόγχων .Μετά αφού ξανακαλέσουμε την συνάρτηση get_walltime έτσι ώστε να αποθηκεύσουμε τον χρόνο εκτέλεσης των πολλαπλασιασμών , υπολογίζω την απόδοση που είχε (mflop = N*N*N/(τελικός_χρόνος – αρχικός_χρόνος)*1.000.000) Τέλος αποδεσμεύω την μνήμη .

Οι διάφορες στα δυο προγράμματα βρίσκονται κυρίως στο ότι στο δεύτερο πρόγραμμα γίνεται χρήση των εντολών ss2 αντί απλών εντολών. Άλλες διαφορές είναι ότι στο δεύτερο πρόγραμμα χρησιμοποιείτε και ένας επιπλέον πίνακας που μας βοηθάει να υπολογίσουμε τις τιμές του πολλαπλασιασμού , καθώς και δυο pointers (pc,phelp) για να αποφύγουμε σφάλματα προσπέλασης μνήμης .

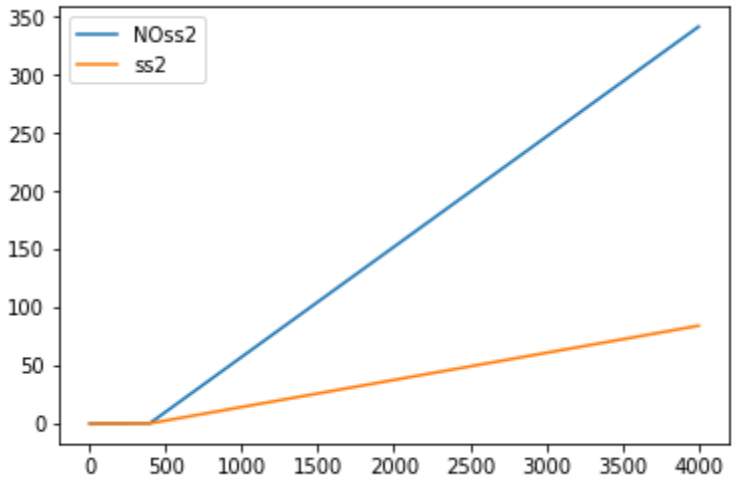
3.0 Αποτελεσματα :

	A	B	C	D	E
1	N	Time	Mflops	SS2 time	SS2 Mflops
2	4	1E-06	67.108864	0	inf
3	40	0.000362	176.834951	0.000104	615.677651
4	400	0.34524	185.378225	0.137685	464.828935
5	4000	341.647302	-1.142537	84.175384	-5.043154

Εικονα 1 -Αποτελέσματα

```
multispiros@multispiros-pc:~$ lscpu
Architecture:            x86_64
CPU op-mode(s):          32-bit, 64-bit
Byte Order:              Little Endian
CPU(s):                  1
On-line CPU(s) list:     0
Thread(s) per core:      1
Core(s) per socket:      1
Socket(s):               1
NUMA node(s):            1
Vendor ID:               GenuineIntel
CPU family:              6
Model:                   61
Model name:              Intel(R) Core(TM) i7-5500U CPU @ 2.40GHz
Stepping:                4
CPU MHz:                 2394.454
BogoMIPS:                4788.90
Hypervisor vendor:       KVM
Virtualization type:     full
L1d cache:               32K
L1i cache:               32K
L2 cache:                256K
L3 cache:                4096K
NUMA node0 CPU(s):       0
Flags:                   fpu vme de pse tsc msr pae mce cx8 apic sep mtrr
```

Εικόνα 2 - Αποτέλεσμα εντολής lscpu



Εικόνα 3 – Διάγραμμα με την χρήση matplotlib (χρόνος - N)

4.0 ΕΠΕΞΗΓΗΣΗ ΑΠΟΤΕΛΕΣΜΑΤΩΝ

Το πείραμα έγινε πάνω σε virtualbox και Ubuntu οπότε είναι λογικό αν κάποια από τα αποτελέσματα δεν ανταποκρίνονται τόσο πολύ στην πραγματικότητα .Αυτό που δεν μπορώ να εξηγήσω είναι τα Mflops για N = 4000 αλλά πιστεύω ότι μάλλον οφείλονται στο παραπάνω γεγονός .

Παρόλα αυτά βλέπουμε ότι οι εντολες ss2 κάνουν την δουλειά τους ειδικά όσο αναφορά το διάστημα τιμών 4000>N>400 . Αυτό οφείλεται στον τρόπο που το ss2 κάνει τους υπολογισμούς(4 πράξεις ανά επανάληψη) . Επίσης παρατηρούμε ότι όσο το N μεγαλώνει μεγαλώνει και ο χρόνος εκτέλεσης .