1η Εργαστηριακή Ασκήσεων
Σχεδιασμός Ενσωματωμένων Συστημάτων

Λεωνίδας Αβδελάς | ΑΜ: 03113182

Ζητούμενο 1ο - Loop Optimizations & Design Space Exploration

1.

- Για την έκδοση λειτουργικού και την έκδοση πυρήνα Linux, θα χρησιμοποιήσουμε τις εντολές `cat /etc/os-release` και `uname -r`, αντίστοιχα. Τα αποτελέσματα είναι PRETTY_NAME=Debian GNU/Linux 10 (buster) και 4.19.0-6-amd64.
- Για την ιεραρχία μνήμης, χρησιμοποιήσαμε την εντολή `sudo lshw -C memory`. Τα αποτελέσματα φαίνονται στον παρακάτω πίνακα:

	L1 cache	L2 cache	L3 cache	RAM
Size	32KiB	256KiB	3MiB	8GiB

- Τις πληροφορίες για τους πυρήνες θα τις βρούμε στο αρχείο /proc/cpuinfo. Έτσι, για τον αριθμό των πυρήνων, τρέχουμε:
  `cat /proc/cpuinfo | grep processor | wc -l`
  από όπου παίρνουμε την απάντηση 4 και για την ταχύτητα τους, τρέχουμε :
  `cat /proc/cpuinfo | grep 'cpu MHz'`
  από όπου παίρνουμε την απάντηση 800Mhz.

2.

Προσθέτοντας τον υπολογισμό χρόνου στο πρόγραμμα μας και τρέχοντας το 10 φορές, έχουμε τα παρακάτω αποτελέσματα:

	Average	Maximum	Minimum
Time	11796.1us	15874us	11335ms

Ένα πράγμα που παρατηρούμε ότι η διαδοχικές εκτελέσεις του προγράμματος επωφελούνται από το caching των δεδομένων. Έτσι, η πρώτη φορά που τρέχει το πρόγραμμα απαιτεί τον μέγιστο χρόνο, και οι επόμενες απαιτούν αισθητά λιγότερο χρόνο.

3.

Εξατάζουμε τον κώδικα, η πρώτη αλλαγή που βλέπουμε ότι μπορεί να γίνει είναι ένα loop merging των λοοπς για τον άξονα ξ. Έτσι γλυτώνουμε B επαναλήψεις για κάθε pixel.

Μετά από μελέτη των μετασχηματισμών, δεν μπορέσαμε να βρούμε κάποιον άλλο που θα μπορούσε να μειώσει την ταχύτητα του προγράμματος και να μπορεί να γίνει χειροκίνητα. Τα αποτελέσματα μας για αυτό το βήμα είναι:

	Average	Maximum	Minimum
Time	10975.us	15245us	10475ms

Όπως είναι εύκολο να δούμε, υπάρχει μια μεγάλη μείωση στον χρόνο εκτέλεσης του προγράμματος μας, αφού η ταχύτητα μειώθηκε κατα σχεδόνο $1000us$, δηλαδή $1ms$.

4.

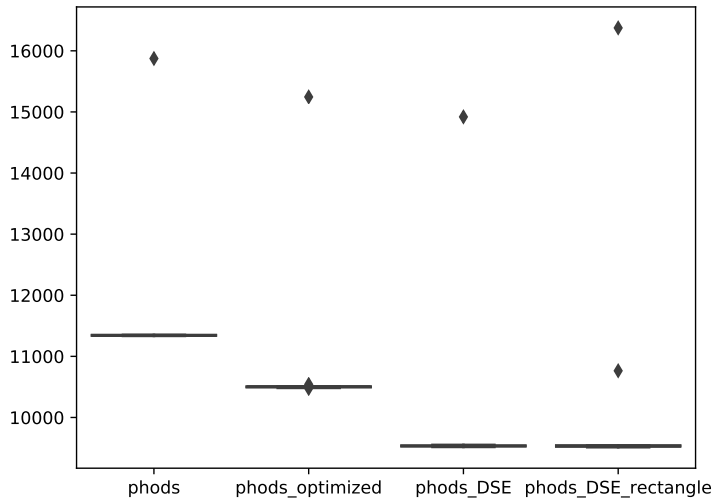
Χρησιμοποιώντας DSE, για το μέγεθος του B για τιμές που είναι κοινοί διαιρέτες του M και του N, βλέπουμε ότι το βέλτιστο μέσο χρόνο τον έχουμε για $B = 16$, με μέση τιμή 9532.4.

Δεν υπάρχει κάποια ακριβής συσχέτιση μεταξύ των δεδομένων του ερωτήματος 1 και των αποτελεσμάτων εδώ. Εφ'όσον το B είναι δύναμη του 2 και οι caches του υπολογιστή επίσης είναι δύναμη του 2 και πολλαπλάσιο του 16, το μεγαλύτερο μέγεθος B εξασφαλίζει μικρότερο αριθμό επαναλήψεων στ εξωτερικό loop.

5.

Το ζεύγος B_x και B_y για το οποίο πετυχαίνουμε την καλύτερη απόδοση είναι το $(144, 176)$ με μέσο χρόνο $9858.22us$. Για να περιορίσουμε την αναζήτηση σε μικρότερο αριθμό παραμέτρων, θα μπορούσαμε πρώτα από όλα να δοκιμάσουμε για τιμές μεγαλύτερες του 16 τόσο για το B_x όσο και για το B_y, αφού στο προηγούμενο ερώτημα ήδη έχουμε βρει ότι η καλύτερη απόδοση για τετράγωνο B είναι το 16. Έτσι θα αποφεύγαμε αρκετές επαναλήψεις στις χαμηλότερες τιμές.

6.



Παρατηρούμε ότι όλα τα προγράμματα έχουν ένα τρέξιμο με πολύ μεγάλο χρόνο

σχετικά με τον μέσο όρο. Αυτό συμβαίνει γιατί στα διαδοχικά τρεξίματα έχουμε επαναχρησιμοποίηση των δεδομένων στην ςαχε και έτσι μειωμένους χρόνος. Ακόμα, η βελτίωση μεταξύ του βήματος 4 και βήματος 5 είναι πολύ μικρή, και αρα η μεγαλύτερη αναζήτηση δεν είναι απαραίτητα χρήσιμη. Αυτό συμβαίνει γιατί μετά από κάποιο σημείο (και επειδή ο υπολογιστής που χρησιμοποιήσαμε είναι αρκετά σύγχρονος και γενικού σκοπού), οι καθυστερήσεις οφείλονται σε άλλους παράγοντες και όχι στο ίδιο το πρόγραμμα.