

1η Εργαστηριακή Ασκήσεων
Σχεδιασμός Ενσωματωμένων Συστημάτων

Λεωνίδας Αβδελάς | AM: 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 πρόγραμμά μας και τρέχοντας το 10 φορές, έχουμε τα παρακάτω αποτελέσματα:

	Average	Maximum	Minimum
Time	11796.1us	15874us	11335us

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 λωοπς 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 αποτελέσματά μας για αυτό το βήμα είναι:

	Average	Maximum	Minimum
Time	10975us	15245us	10475us

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

4.

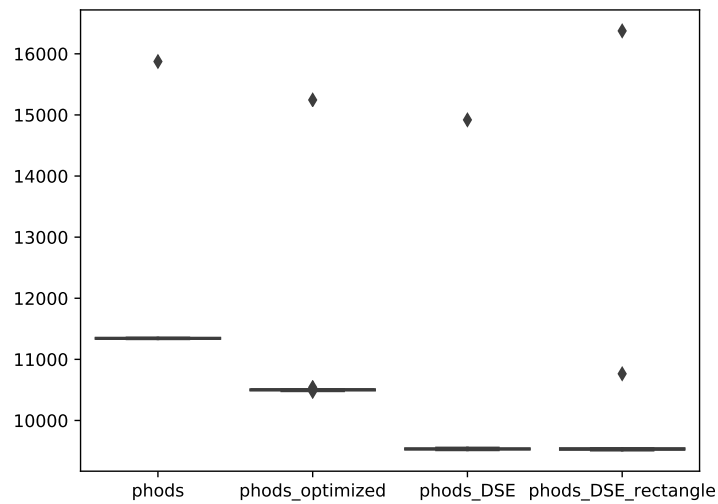
Χρησιμοποιώντας DSE, για το μέγεθος του B για τιμές που είναι κοινοί διαιρέτες του M και του N , βλέπουμε ότι το βέλτιστο μέσο χρόνο τον έχουμε για $B = 16$, με μέση τιμή 9532.4 .

Δεν υπάρχει κάποια ακριβής συσχέτιση μεταξύ των δεδομένων του ερωτήματος 1 και των αποτελεσμάτων εδώ. Εφόσον 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 ζαση 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.

Ζητούμενο 2: Αυτοματοποιημένη βελτίωση κώδικα

1.

Για το μη μετασχηματισμένο πρόγραμμα, έχουμε τα παρακάτω αποτελέσματα:

	Average	Maximum	Minimum
Time	742166.6us	747550us	733782us

2.

Για το **exhaustive search**, το Unrolling Factor ήταν 3. Για το **Randomsearch**, το Unrolling Factor ήταν 4. Για το **Simplex**, το Unrolling Factor ήταν 4.

Παρατηρώντας τις εξόδους των τριών σεαρχης μπορούμε εύκολα να δούμε ότι η διαφορά στο αποτέλεσμα οφείλεται στους παρακάτω λόγους:

- Στο Random Search, δεν έγινε έλεγχος της τιμής 3 για το UF αλλά έγινε στην τιμή 4. Πιστεύουμε ότι το αποτέλεσμα αυτό είναι ικανοποιητικό για την μείωση των παραμέτρων από 33 σε 10.
- Στο Simplex, το πρόβλημα ήταν ότι δεν βρέθηκε ο μέσος όρος του χρόνου τρεξίματος για τα πιθανά αποτελέσματα, καθώς αυτό θα καθυστέρουσε πολύ την ευριστική (τουλάχιστον αυτό είναι το συμπέρασμα μας από την έλεγχο των logs). Για την μια φορά τρεξίματος των παραμέτρων, η τιμή για UF 4 ήταν καλύτερη από 3. Ξανά θεωρούμε πως η μείωση των παραμέτρων άξιζει την μικρή διαφορά σε χρόνο εκτέλεσης.

3.

	Average	Maximum	Minimum
Non-Optimized Time	742166.6us	747550us	733782us
Exhaustive Optimized Time	643717.50us	649229us	640655us
Rand & Simplex Optimized Time	668284.70us	673756us	662471us

Τα αποτελέσματα είναι όπως περιμέναμε, αισθητά βελτιωμένα χάρη στο loop unrolling, και η διαφορά μεταξύ των UF 3 και 4 αρκετά μικρή.