2019-20

README

PART 3

ΗΛΙΑΣ ΜΗΝΔΡΙΝΟΣ – ΔΗΜΗΤΡΗΣ ΧΑΣΚΟΠΟΥΛΟΣ

Λειτουργικότητα Part 3

Το πρόγραμμα έχει δοκιμαστεί και υλοποιηθεί σε Virtual Machine Ubuntu με μέγιστη RAM 8GB και 4 cores. Υπάρχουν μερικά defines στο Radix.h που αλλάζουν την συμπεριφορά του προγράμματος και είναι χρήσιμα για την διεξαγωγή διαφόρων test καθώς και για την συνολική λειτουργικότητα του τελευταίου Part της εργασίας. Αρχικά το print_in_line χρησιμοποιείται για να τυπώνονται τα αποτελέσματα με την σειρά που μπήκαν τα queries (μόνο ουσιαστικά όταν έχουμε big_threads). Το do_big_thread είναι σχεδιασμένο για να κάνει ή όχι threading σε επίπεδο query. Τα υπόλοιπα big_threads, join_threads και sort_threads αφορούν το πλήθος των threads που θα δημιουργηθούν κατά την εκτέλεση, αν είναι 0 ή μικρότερο θα εκτελεστεί ο κώδικας χωρίς threads στο αντίστοιχο κομμάτι.

Σχεδιαστικές Επιλογές

Κατά την ανάπτυξη και υλοποίηση της εργασίας υπήρχαν αρκετές αλλαγές και σχεδιαστικές επιλογές όπου επηρέασαν την αποδοτικότητα και την λειτουργικότητα του συνολικού προγράμματος. Παρακάτω παρατίθενται μερικές αν όχι όλες από αυτές.

- Αρχικά είχε υλοποιηθεί ο αλγόριθμος sorting Radix με βάση μιας δικιάς μας υλοποίησης που βασιζόταν στα παραδείγματα της εκφώνησης του 1ου μέρους. Δημιουργούσε ένα πίνακα sum_list που έδειχνε την αρχή κάθε bucket που ήταν να γίνει sort. Αυτό όμως άλλαξε και έγινε αναδρομικά με DFS. Κάθε αναδρομική κλήση του εαυτού της έκανε sort το εκάστοτε bucket για τα πρώτα μέχρι και τα τελευταία 8 bits, εφόσον ξεπερνιόταν το όριο για quicksort. Σε διαφορετική περίπτωση γινόταν quicksort και τα μετέφερε από τον πίνακα A στον πίνακα B.

- Μια άλλη σχεδιαστική επιλογή ήταν η λίστα με τα αποτελέσματα από το join να αλλάξει σε πίνακα για το 2ο μέρος της εργασίας καθώς δημιουργούσε τα ενδιάμεσα αποτελέσματα μέσα στο ίδιο query γιατί είναι πιο άμεση και λιγότερο χρονοβόρα η πράξη sort.

- Σχεδιαστική επιλογή αποτέλεσε η λειτουργία ή όχι των threads όπως περιγράφεται και στην ενότητα **Λειτουργικότητα Part 3**, καθώς η συμπεριφορά των thread δεν ήταν η αναμενόμενη.

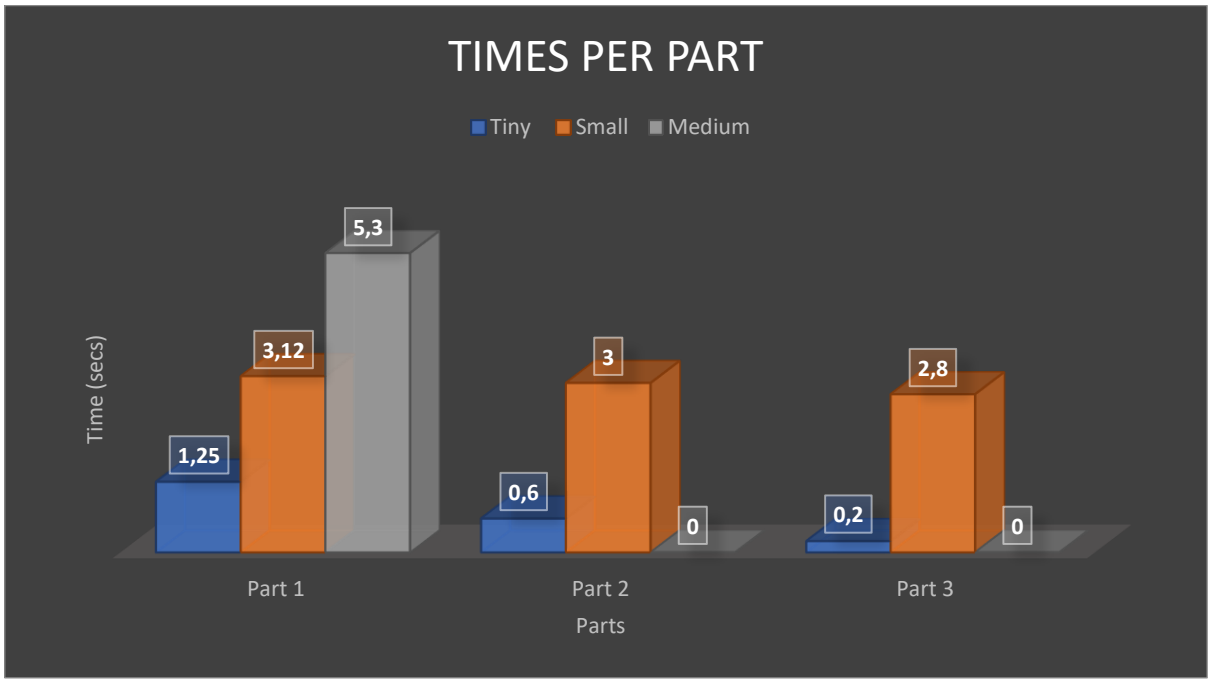
- Στο 2ο μέρος γινόταν αναζήτηση του μικρότερου πίνακα μέσα στο query και δημιουργούσε έναν δέντρο εκτέλεσης πράξεων στο οποίο δεν θα υπήρχε καρτεσιανό γινόμενο. Αυτό άλλαξε στο 3ο μέρος και πλέον εφαρμόζονται τα φίλτρα τα οποία υποδεικνύονται στην εκφώνηση για την καλύτερη εκτέλεση των πράξεων. Επίσης ψάχνει να βρει αν κάποια κολώνα του ίδιου πίνακα ξαναχρησιμοποιείται και τις συνδυάζει για μην ξανακάνει sort.

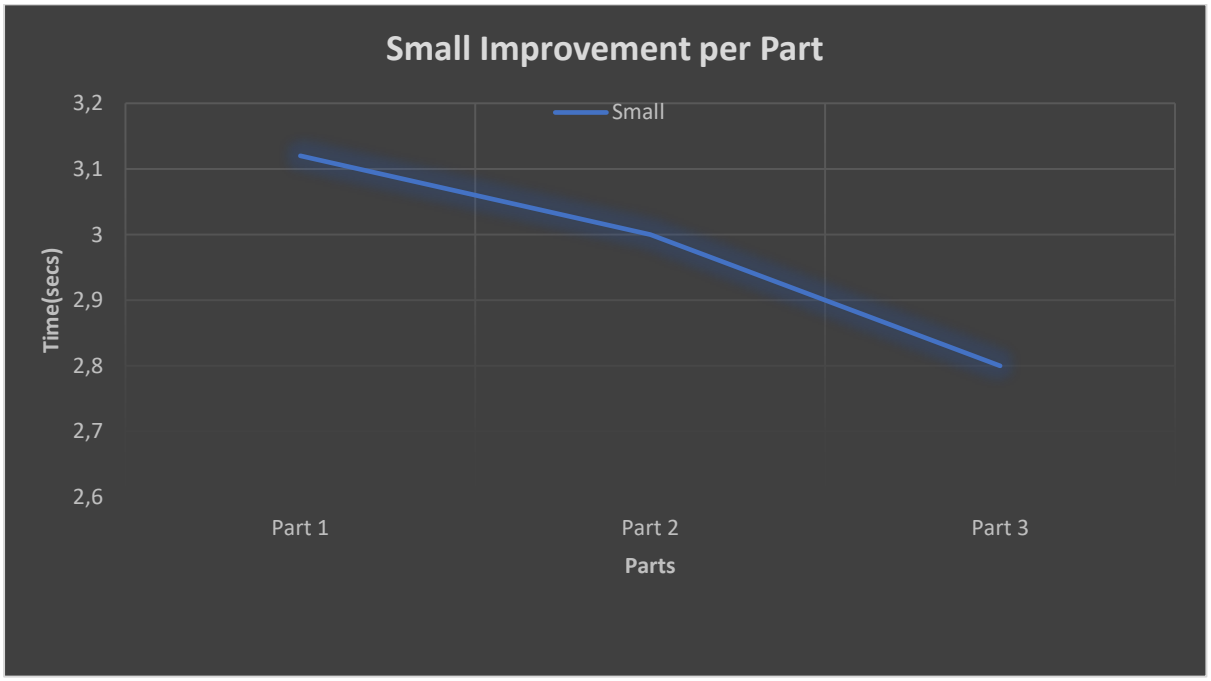
- Το join αρχικά δεν διέτρεχε για κάθε στοιχείο του πίνακα A τον πίνακα B, ωστόσο αυτό άλλαξε και έγινε σωστά στο 2ο μέρος όπου πλέον ξεκινάει από το τελευταίο στοιχείο του B που ταίριαζε με το προηγούμενο στοιχείο A και σταματάει μέχρι να το προσπεράσει.

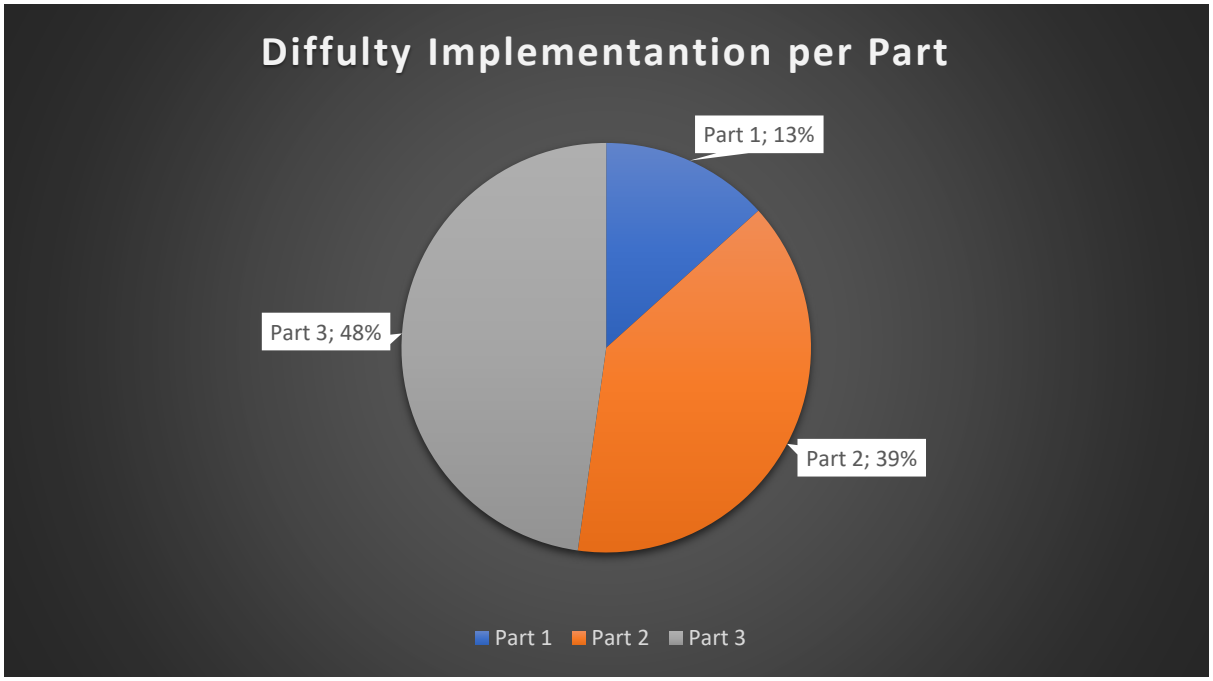
- Στο 3ο μέρος της εργασίας έγινε αποτυχημένη απόπειρα για Job Scheduler ωστόσο λόγο τεχνικών προβλημάτων δεν μπόρεσε να εφαρμοστεί. Αντί αυτού έγιναν 3 ξεχωριστά Thread Pools με τα 2 από αυτά να συγχρονίζονται και το ένα όχι, ενώ θα μπορούσαν να γίνουν εμφωλευμένα.

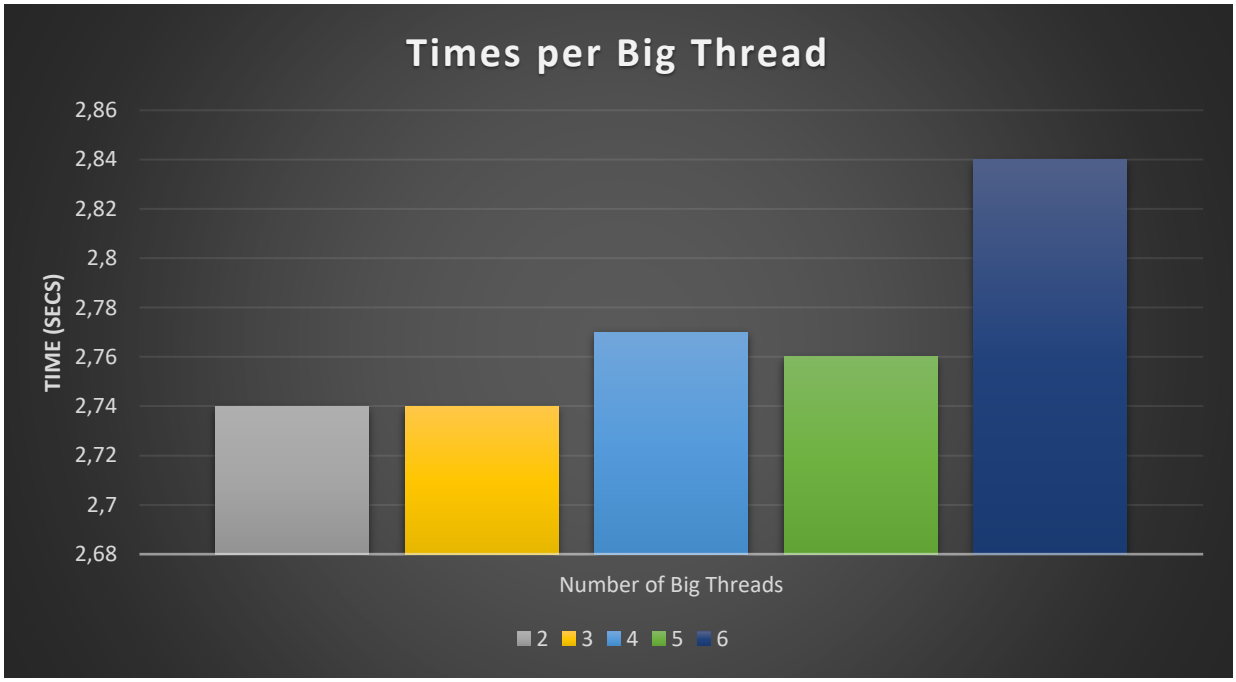
Διαγράμματα και Πίνακες

Παρακάτω παρατίθενται ενδεικτικά διαγράμματα και πίνακες από τις διάφορες φάσεις του προγράμματος.









Σημαντικές Σημειώσεις

Έχοντας υλοποιήσει το 2ο μέρος φτιάξαμε και τα memory leaks τα οποία είχε και πλέον είναι στα 0!!! Ωστόσο δεν μπορέσαμε να τρέξουμε το Medium file στα δικά μας μηχανήματα και αυτό εξολοκλήρου λόγω έλλειψης μνήμης RAM 8GB, πιστεύουμε ότι στα μηχανήματα της σχολής που έχουν 16GB θα τρέχει κανονικά όπως είχε ειπωθεί και μέσα στην τάξη. Για αυτόν ακριβώς τον λόγο μας ήταν αδύνατο να τρέξουμε το Medium file με Threads γιατί όπως πάλι ειπώθηκε 2 Threads μαζί μπορεί να τερμάτιζαν την μνήμη ακόμη και σε εργασίες που είχαν τρέξει το Medium στο 2ο μέρος. Ελπίζουμε μέχρι την ημέρα εξέτασης να το έχουμε τρέξει στα μηχανήματα της σχολής ή και να το τρέξουμε εκεί μαζί σας την μέρα της εξέτασης γιατί σας χρωστάγαμε κάποιες διορθώσεις που κάναμε στα Part 1 & 2 καθώς προχωράγαμε στο επόμενο.

Εντολές Εκτέλεσης Προγράμματος

make all
./myexe small.init small.work
make clean