Ανάπτυξη Λογισμικού Για
Πληροφοριακά Συστήματα
Τρίτο Μέρος

Γεώργιος Χαλεπούδης - A.M:1115201200192

Στυλιανός Σύριγγας - A.M:1115201600164

Συνοπτική περιγραφή όλων των επιπέδων του project

Στο πρώτο μέρος της εργασίας υλοποιήθηκε ένα αλγόριθμος ζεύξης μεταξύ δύο πινάκων, ο οποίος διαφοροποιείται στο γεγονός ότι είναι σχεδιασμένος για περιορισμένο χώρο μνήμης και ότι είναι εύκολα παραλληλοποιήσιμος. Αυτός ο αλγόριθμος ονομάζεται SortMergeJoin και αποτελείται από δύο στάδια, το στάδιο της ταξινόμησης και της ζεύξης. Πιο συγκεκριμένα ο αλγόριθμος της ταξινόμησης χωρίζει τον πίνακα σε κουβάδες όπου τα στοιχεία του έχουν κοινό πρόθεμα. Τέλος, και ο αλγόριθμος ταξινόμησης και ο αλγόριθμος της τελικής ζεύξης μπορούν να παραλληλοποιηθούν για να αυξηθεί η συνολική απόδοση.

Στο δεύτερο μέρος της εργασίας υλοποιήθηκε ένα σύστημα το οποία με δεδομένα πινάκες και επερωτήσεις έχουν ως έξοδο το αποτέλεσμα των προηγουμένων. Για να επιτευχθεί αυτό σχεδιάστηκαν δομές οι οποίες είτε αποθήκευαν τα στοιχεία των πινάκων, είτε κρατούσαν τα αποτελέσματα των κατηγορημάτων. Επίσης μεταβλήθηκε ο αλγόριθμος του προηγούμενου επίπεδου έτσι ώστε να είναι πιο αποδοτικός για πολλές ζεύξεις.

Στο τρίτος μέρος της εργασίας υλοποιήθηκαν δύο τρόποι έτσι ώστε η συνολική απόδοση του προγράμματος δηλαδή η ταχύτητα και η κατανάλωση μνήμης να βελτιωθεί. Ο πρώτος τρόπος ήταν μέσω παραλληλίας. Πιο συγκεκριμένα επιτεύχθηκε παραλληλία στο επίπεδο των επερωτήσεων, στον αλγόριθμο της ταξινόμησης και στον αλγόριθμος της ζεύξης. Ο δεύτερος τρόπος ήταν μέσω της βελτιστοποίησης των επερωτήσεων δηλαδή η σειρά όπου θα εκτελεστούν τα κατηγορήματα της επερώτησης. Για την υλοποίηση αυτό χρησιμοποιήθηκαν διάφορα στατιστικά τα οποία βοήθησαν στην εκτίμηση της πληθικότητας και των κοστών των συνδυασμών εκτέλεσης.

Join jobs

Η υλοποίηση των join jobs δεν είναι ολοκληρωμένη καθώς όταν χρησιμοποιούνται τα medium dataset το πρόγραμμα κρασάρει. Για αυτό το λόγο χρησιμοποιείται το γραμμικό join, το οποίο είναι αποδοτικότερο δηλαδή πιο γρήγορο και καταναλώνει λιγότερη μνήμη.

Κάποιες Παρατηρήσεις

-Υπάρχει μικρή βελτίωση αν τμηματοποιηθούν οι κάδοι σε 16 bits αντί για 8.

-Πολλοί πίνακες δεν έχουν ομοιόμορφα κατανανεμημένες τιμές οπότε η παραλληλοποίηση των ταξινομήσεων δεν είναι πολύ αποδοτική.

-Αν ο αριθμός των threads είναι σχετικά χαμηλός, ανεξαρτήτως των πυρήνων, υπάρχει καθυστέρηση στην εκτέλεση του προγραμμάτος και υπάρχει και κίνδυνος να κλειδωθεί το πρόγραμμα. Αυτό συμβαίνει γιατί κάποιο thread μπορεί να περιμένει τα υπόλοιπα όπως τα sort,join, οπότε αν ο αριθμός των threads είναι μικρός μπορεί όλα να περιμένουν, παρ'όλο που οι επεξεργαστές δεν δουλεύουν.

Μετρήσεις

Οι μετρήσεις έχουν γίνει στα μηχανήματα της σχολής, τα οποία έχουν 4 πυρήνες και 16gb RAM.

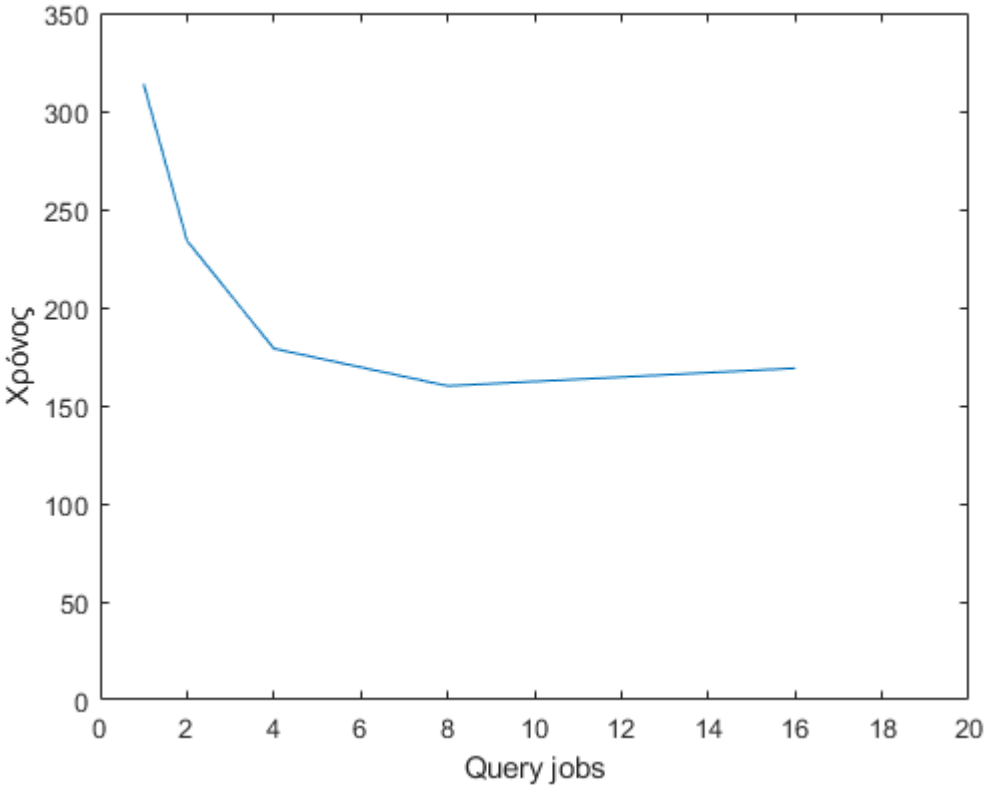
Query jobs:Ο αριθμός των query που εκτελούνται ανά batch(χωρίς αναγκαστικά να γίνει η εκτύπωση

Sort jobs:Ο αριθμός των jobs που χωρίζεται ένας πινάκας στη διάρκεια της ταξινόμησης.

Join jobs:Ο αριθμός των jobs που χωρίζεται ένας πινάκας στη διάρκεια της ζεύξης.

Πίνακας χρόνου και αριθμός query jobs με 1 sort job και 1 join jobs

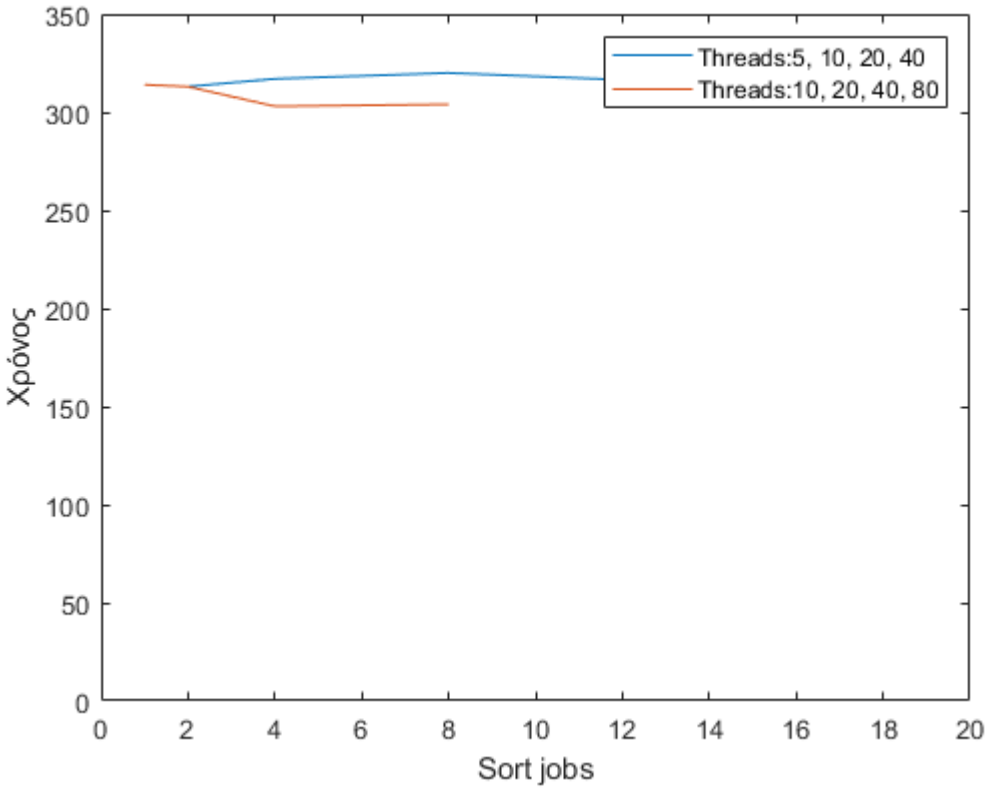
Threads	QueryJobs	Χρόνος(seconds)
4	1	314
8	2	234
16	4	179
32	8	160
64	16	169



Παρατηρείται ελάττωση του χρόνου καθώς αυξάνεται ο αριθμός των query jobs και των νημάτων. Αυτό είναι αναμενόμενο καθώς τα query jobs είναι ανεξάρτητα μεταξύ τους οπότε όσο αυξάνεται ο αριθμός τους τόσο ποιο γρήγορο γίνεται το πρόγραμμα, με όριο τους επεξεργαστές του μηχανήματος.

Πίνακας χρόνου και αριθμός sort jobs με 1 query job και 1 join jobs

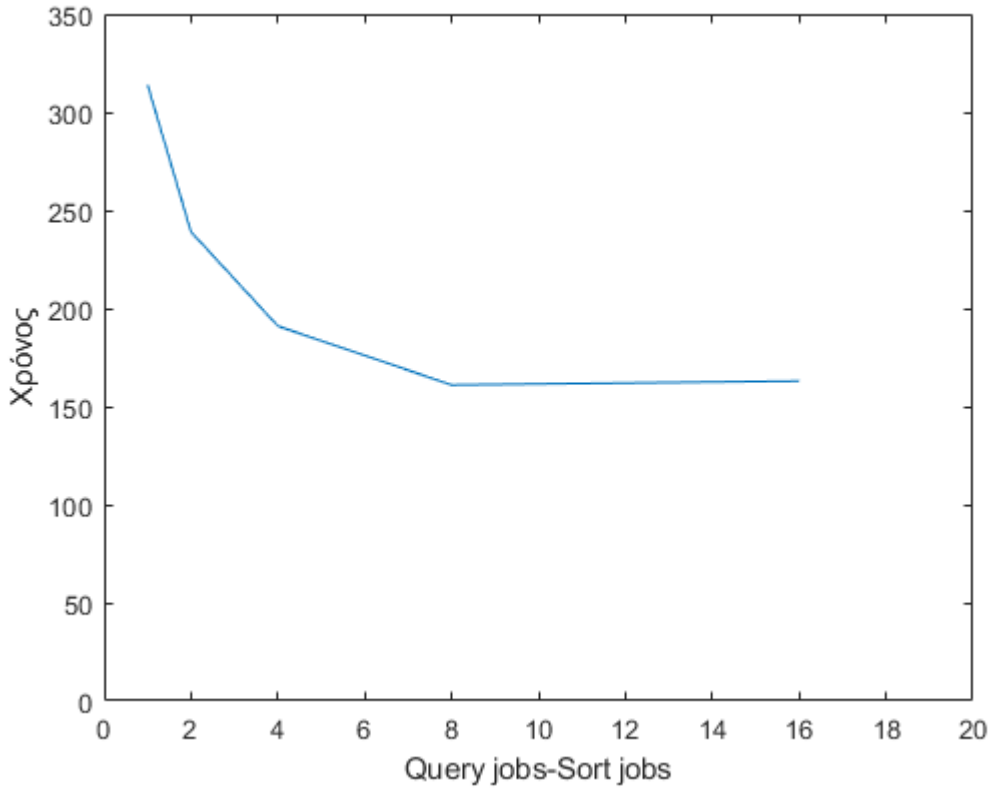
Threads	SortJobs	Χρόνος(seconds)
5	2	313
10	2	313
10	4	317
20	4	303
20	8	320
40	8	314
40	16	313
80	16	Killed



Φαίνεται ότι παρ'όλη την αύξηση του πλήθους των sort jobs η απόδοση του προγράμματος δεν αυξάνεται. Αυτό συμβαίνει γιατί η κατανομή των στοιχείων στους αντίστοιχους κουβάδες δεν είναι ομοιόμορφη, οπότε αφού κάθε sort job διαχειρίζεται έναν αριθμό από κουβάδες, πολλές φορές κα-ταλήγει μόνο ένα job να είναι παραγωγικό, αρά ο συνολικός χρόνος να είναι κοντινός στον γραμμικό.

Πίνακας χρόνου και αριθμός sort jobs και query jobs με 1 join jobs

Threads	QueryJobs-SortJobs	Χρόνος(seconds)
10	2-2	239
36	4-4	191
136	8-8	161
528	16-16	163



Περίπου τα ίδια αποτελέσμα με την περίπτωση με 1 sort job. Φαίνεται πάλι ότι η παραλληλοποίηση της ταξινόμησης δεν είναι αποδοτική.

Μνήμη και jobs

Ανεξαρτήτως από τον αριθμό των query jobs και sort jobs η μνήμη που διανέμει το πρόγραμμα είναι γύρω στα 37.690.557.000 bytes. Όμως με 2 join jobs ανά query η συνολική μνήμη φτάνει στα 285.855.502.717.

Δομές για βελτιστοποιήση χρόνου

Οι καλύτεροι παράμετροι για την εκτέλεση του προγράμματος φαίνεται να είναι τα 32 Threads, 8 Query jobs, 1 Sort job, 1 Join job. Όπως αναφέρθηκε και παραπάνω οι δοκιμές έγιναν σε μηχανές των τεσσάρων πυρήνων με 16gb RAM, οπότε αυτοί οι βέλτιστοι παράμετροι ενδέχεται να διαφοροποιούνται σε άλλους υπολογιστές.