ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΑΝΑΠΤΥΞΗ ΛΟΓΙΣΜΙΚΟΥ ΓΙΑ ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ 2015-2016

Βαλελής Χρήστος 1115201100139
Κάπαλος Νικόλαος 1115201100103
Κιοσές Χαράλαμπος 1115201100059

Γενικά Σχόλια – Παρατηρήσεις

- Η εργασία υλοποιήθηκε σύμφωνα με τις προδιαγραφές των εκφωνήσεων Part1, Part2 (χωρίς το bonus της επικάλυψης) και Part3 (με υλοποίηση job scheduler).
- Η ανάγκη για λίστες πολλών διαφορετικών δεδομένων μας οδήγησε στην δημιουργία Template λίστας.
- Αντίστοιχα δημιουργήσαμε Template ευρετήριο λόγω της διαφορετικότητας των δεδομένων.
- Εκτός από τις βασικές συναρτήσεις που ζητήθηκαν, δημιουργήσαμε και άλλες οι οποίες λειτουργούν βοηθητικά επιτελώντας πιο εξειδικευμένες λειτουργίες για επίτευξη μεγαλύτερης ταχύτητας στην εκτέλεση (π.χ. getJournalJR2, getJournalJR3).
- Το ευρετήριο υλοποιήθηκε μέσω της κλάσης Hashtable, η οποία αποτελείται από ένα πίνακα της κλάσης Bucket. Κάθε Bucket περιέχει πίνακα από διαφορετικά κλειδιά και έναν πίνακα της κλάσης Container. Κάθε στοιχείο του πίνακα των κλειδιών αντιστοιχεί 1-1 με ένα στοιχείο του πίνακα της κλάσης Container. Κάθε Container περιέχει έναν επεκτατό πίνακα από στοιχεία με το ίδιο κλειδί, υλοποίηση που επιτρέπει μεγαλύτερη ταχύτητα σε όλες τις λειτουργίες του ευρετηρίου (Split,Double,Merge,Halve).
- Για επίτευξη μεγαλύτερης ταχύτητας, υλοποιήθηκε το ευρετήριο με τέτοιο τρόπο ώστε να υποστηρίζεται η ύπαρξη ενός ή περισσότερων διαφορετικών κλειδιών (άσχετα με το πλήθος των εγγραφών για κάθε κλειδί) στο ίδιο Bucket, ανάλογα με τις εκάστοτε ανάγκες.
- Για επίτευξη μεγαλύτερης ταχύτητας σε ότι αφορά την σειριακή αντιγραφή δεδομένων όπου χρειάστηκε χρησιμοποιήθηκε η συνάρτηση memcpy.
- Παρατηρήθηκε ότι οι μεγαλύτερες καθυστερήσεις στις εκτελέσεις όλων των σεναρίων εμφανίζονται κατά την σειριακή εισαγωγή των δεδομένων από τη δομή Journal σε λίστα όπως ζητήθηκε.
- Για αναζήτηση σε ταξινομημένους πίνακες χρησιμοποιήθηκε binary search όπου ήταν δυνατό.
- Σε δοκιμές που κάναμε σε μηχανήματα σχετικά χαμηλών προδιαγραφών παρατηρήθηκε ότι η χρήση νημάτων δεν είχε κάποια διαφορά στον χρόνο εκτέλεσης για τα αρχεία basic.bin και small.bin , ενώ στο μηχάνημα υψηλότερων προδιαγραφών (όπου εν τέλει δοκιμάστηκαν τα σενάρια) παρατηρήθηκε βελτίωση.

Πειραματικές Δοκιμές

- Δημιουργήσαμε επτά σενάρια λειτουργίας τα οποία εκτελούνται με ευκολία για κάθε αρχείο που μας έχει δοθεί, μέσω του Makefile που παρέχεται. Τα σενάρια αυτά αποτελούνται από στοιχεία όλων των επιπέδων (Parts) και διαρθρώνονται ως εξής :

    1. Χρήση του ευρετηρίου του πρώτου επιπέδου (getHashJR) για εύρεση των υποψήφιων εγγραφών και υπολογισμός.
    2. Χρήση του ευρετηρίου του δεύτερου επιπέδου (getTransactionIndexRecord) για εύρεση των υποψήφιων εγγραφών και υπολογισμός.
    3. Χρήση του ευρετηρίου του δεύτερου επιπέδου (getTransactionIndexRecord) για εύρεση των υποψήφιων εγγραφών και υπολογισμός μέσω της δομής ValidationIndex.
    4. Χρήση του ευρετηρίου του πρώτου επιπέδου (getHashJR) σε συνδυασμό με το ευρετήριο του δεύτερου επιπέδου (getTransactionIndexRecord) για εύρεση των υποψήφιων εγγραφών και υπολογισμός.
    5. Το σενάριο 1 με χρήση νημάτων κατά τον υπολογισμό.
    6. Το σενάριο 2 με χρήση νημάτων κατά τον υπολογισμό.
    7. Το σενάριο 4 με χρήση νημάτων κατά τον υπολογισμό.

- Ακολουθούν ενδεικτικές εκτελέσεις του κάθε σεναρίου για κάθε διαφορετικό αρχείο εισόδου που μας έχει δοθεί. Σημειώνεται ότι σε κανένα από αυτά δεν παρατηρήθηκε λάθος κατά τον υπολογισμό τους. Επίσης μετά από εκτέλεση Valgrind παρατηρήθηκε ότι δεν μένουν δεσμευμένα bytes στην μνήμη.
- Δεν έχουμε ενδεικτικές εκτελέσεις για basic.bin καθώς οι διαφορές κυμαίνονται σε microseconds

Οδηγίες Εκτέλεσης

Για ευκολία στην εκτέλεση χρησιμοποιούμε το Makefile ως εξής:
make <scenario>run<fileId>

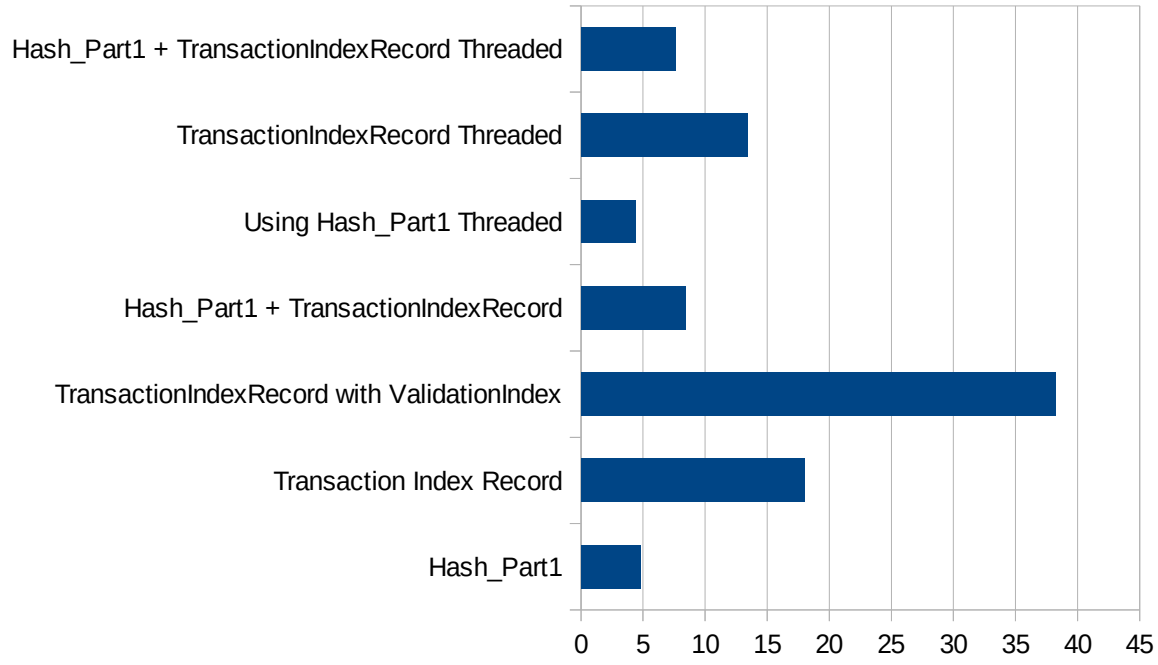
όπου scenario από 1 μέχρι 7 και όπου fileId, 1 για basic.bin, 2 για small.bin, 3 για medium.bin

Για παράδειγμα για να εκτελέσουμε το σενάριο 5 για το αρχείο small.bin γράφουμε:
make 5run2

Ενδεικτικές Εκτελέσεις

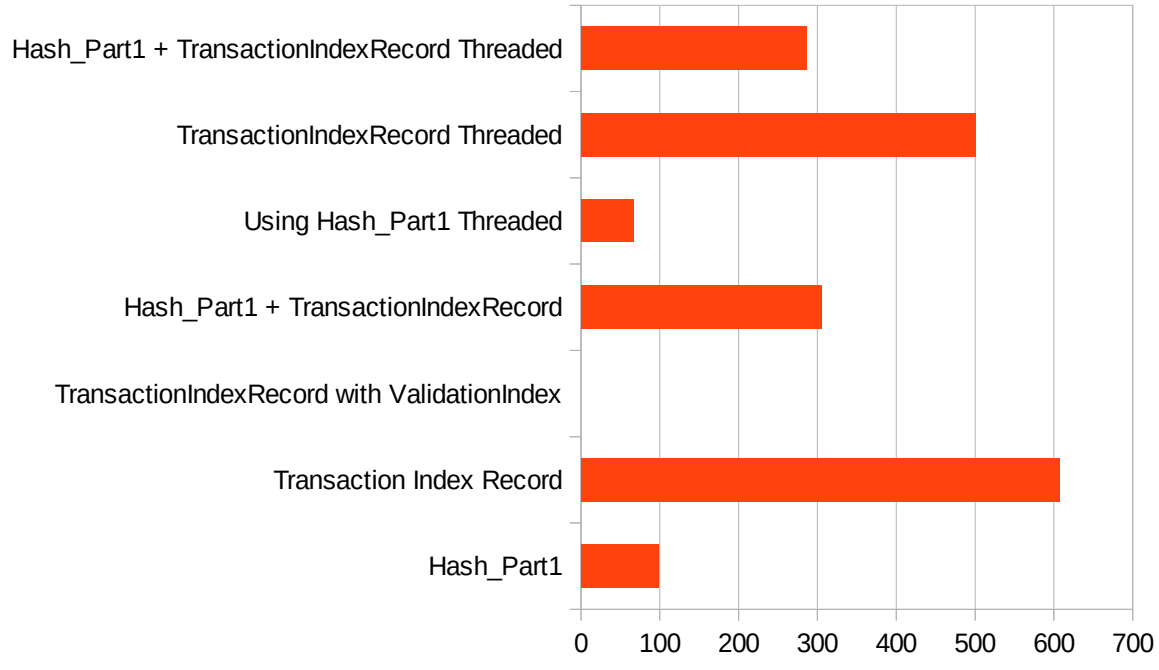
Small.bin

Σενάριο	Time complexity(secs)	Space complexity(bytes)
Hash_Part1	4.79579	26,054,410 allocs, 26,054,410 frees, 713,161,452 bytes
Transaction Index Record	18.0016	212,422,744 allocs, 212,422,744 frees, 3,864,484,559 bytes
TransactionIndexRecord with ValidationIndex	38.1994	-
Hash_Part1 + TransactionIndexRecord	8.39731	-
Using Hash_Part1 Threaded	4.38997	26,292,906 allocs, 26,292,906 frees, 710,231,508 bytes
TransactionIndexRecord Threaded	13.4132	230,711,372 allocs, 230,711,372 frees, 3,980,038,084 bytes
Hash_Part1 + TransactionIndexRecord Threaded	7.61441	81,572,671 allocs, 81,572,671 frees, 1,594,707,748 bytes



Medium.bin

Σενάριο	Time complexity(secs)	Space complexity(bytes)
Hash_Part1	98.5515	-
Transaction Index Record	607.718	-
TransactionIndexRecord with ValidationIndex	-	-
Hash_Part1 + TransactionIndexRecord	305.438	923,896,453 allocs, 923,896,453 frees, 16,464,364,616 bytes allocated
Using Hash_Part1 Threaded	66.46	-
TransactionIndexRecord Threaded	500.66	-
Hash_Part1 + TransactionIndexRecord Threaded	286.66	-



*Σημειώνεται ότι όπου υπάρχουν παύλες στους παραπάνω πίνακες είναι διότι το Valgrind αργούσε υπερβολικά.

Οι παραπάνω εκτελέσεις έτρεξαν σε μηχάνημα με τα ακόλουθα χαρακτηριστικά:

Linux mint 17.2 64bit
Πυρήνας Linux 3.16.0-38-generic x86_64
MATE 1.10.2
Intel® Core™ i7-2600K CPU @ 3.40GHz × 4 (8 threads)
8 GB Ram