Ανάπτυξη Λογισμικού για Πληροφοριακά Συστήματα

Τελική Εργασία

Χωριανοπούλου Αικατερίνη A.M. 1115201100084
Βασιλακόπουλος Χαράλαμπος A.M. 1115201100011

Για το compilation περιλαμβάνεται makefile. Η εντολή εκτέλεσης είναι ως εξής:

./acid -l[1,2,3] -th[1-8] -r[>0] -t

Το όρισμα **-l** ορίζει το επίπεδο στο οποίο θα τρέξει το πρόγραμμα.
Το όρισμα **-th** ορίζει τον αριθμό των threads που θα χρησιμοποιηθούν.
Το όρισμα **-r** ορίζει τον αριθμό rounds μέχρι να υπολογιστούν validations.
Το όρισμα **-t** ορίζει το αν θα χρησιμοποιηθεί το transactionHash.

Οι δομές που υλοποιήθηκαν για το σκοπό της άσκησης είναι οι εξής:

- **Hash<typename T>:** Μία templated δομή hash που χρησιμοποιείται με όρισμα RangeArray, uint32_t και ValidationEntry για τις διάφορες χρήσεις που απαιτούνται. Έχει μεταβλητό αριθμό επιτρεπόμενων εγγραφών ανά κάδο κατά την αρχικοποίηση της.
- **Journal:** Μία δομή που περιέχει ένα πίνακα από structs που μέσα τους περιέχουν ένα uint32_t(transactionId) και έναν πίνακα από τιμές που είναι τα columns της εκάστοτε εγγραφής. Το transactionId μπήκε ξεχωριστά για να διευκολύνει τη δυαδική αναζήτηση στον πίνακα.
- **RangeArray:** Η δομή που κρατάει για κάθε primary key(column0) τα transactionId's που το “αγγίζουν”. Περιέχει ένα πίνακα από structs που κρατάνε transactionId και τα offsets στο journal που πειράχτηκε αυτό το primary key. Υπάρχει ένα για κάθε διαφορετικό primary key.
- **List<typename T>:** Μια απλή templated single linked λίστα που χρησιμοποιείται για να κρατάμε τα validations μέχρι να εξεταστούν καθώς και pointers προς αυτά μέσα στο ValidationEntry.
- **ValidationEntry:** Περιέχει ένα predicate καθώς και ένα bitset που κρατάει τα αποτελέσματα αυτού του prediacate πάνω στο range του. Σε περίπτωση που του ζητηθεί range που δεν έχει ελεγχθεί ακόμα αναλαμβάνει να γεμίσει τα κενά. Υπάρχει ένα για κάθε διαφορετικό predicate που έχει συναντηθεί σε validation.
- **InfoArray:** Κρατάει τα transactionId's σε ένα πίνακα για γρήγορη δυαδική αναζήτηση. Χρησιμοποιείται στο επίπεδο 2 για να αποφευχθεί η δυαδική αναζήτηση στο journal που είναι ποιο αργή καθώς έχουμε μεγάλο αριθμό εγγραφών ανά transactionId. Υπάρχει ένα για κάθε διαφορετικό relation που έχουμε και έχει δυναμικό μέγεθος που είτε μεγαλώνει αν δε χωράει άλλες εγγραφές είτε μικραίνει ως αποτέλεσμα forget.
- **JobScheduler:** Δομή που χρησιμοποιείται μόνο στο επίπεδο 3 και σκοπό έχει να οργανώσει μέσω pthread_mutex και pthread_cond τα threads ώστε να παίρνουν validations από μια εσωτερική του λίστα και να τα υπολογίζουν παράλληλα.

Αρχικά, είχαμε υλοποιήσει το ValidationEntry έτσι ώστε να κρατάει εσωτερικά ένα πίνακα με offsets στο bitset ώστε να ξέρουμε που να ξεκινήσουμε να διαβάζουμε όταν μας δοθεί ένα επιθυμητό range, αλλά από δοκιμές που κάναμε συνειδητοποιήσαμε ότι όχι μόνο είναι περιττό αλλά πιάνει και πάρα πολύ χώρο στη μνήμη. Οπότε και μετατρέψαμε αυτή τη δομή στο InfoArray που περιγράφεται παραπάνω ως συμβιβασμό μεταξύ ταχύτητας εκτέλεσης και απαιτούμενης μνήμης.

Όπως θα δούμε και πιο κάτω στα διαγράμματα για χρήση μνήμης το μεγαλύτερο ποσοστό μνήμης το καταλαμβάνει το Hash<RangeArray> σε όποιο επίπεδο και να τρέξουμε το πρόγραμμα. Καθώς θα είναι ασύμφορο να υλοποιηθεί η deleteTransaction με κάποιο διαφορετικό τρόπο και επίσης προκαλεί μεγάλη επιτάχυνση και στα τρία επίπεδα όταν χρησιμοποιείται για τον υπολογισμό των validations, επιλέξαμε να τρέχει και στα τρία επίπεδα by default.

Η συνάρτηση που προσθέτει αποτελέσματα στο bitset του ValidationEntry είναι η addrange. Εντοπίσαμε ότι αυτή η συνάρτηση είναι το bottleneck του δευτέρου επιπέδου καθώς για κάθε predicate που εξετάζει (και δεν έχει ήδη αυτούσιο το επιθυμητό range) το validation στο οποίο βρισκόμαστε κάνει αναζήτηση και προσπέλαση στο journal. Μια σκέψη που είχαμε για τη βελτίωση του χρόνου εκτέλεσης της ήταν να κάνει προσπέλαση το journal μόνο μία φορά και να συμπληρώνει παράλληλα όλα τα αναγκαία bitsets αλλά λόγω της πολυπλοκότητας αυτής της λύσης και του χρόνου που είχαμε δεν προλάβαμε να το ελέγξουμε αρκετά ώστε να διορθώσουμε τα bugs και να τη περάσουμε απο δοκιμές.

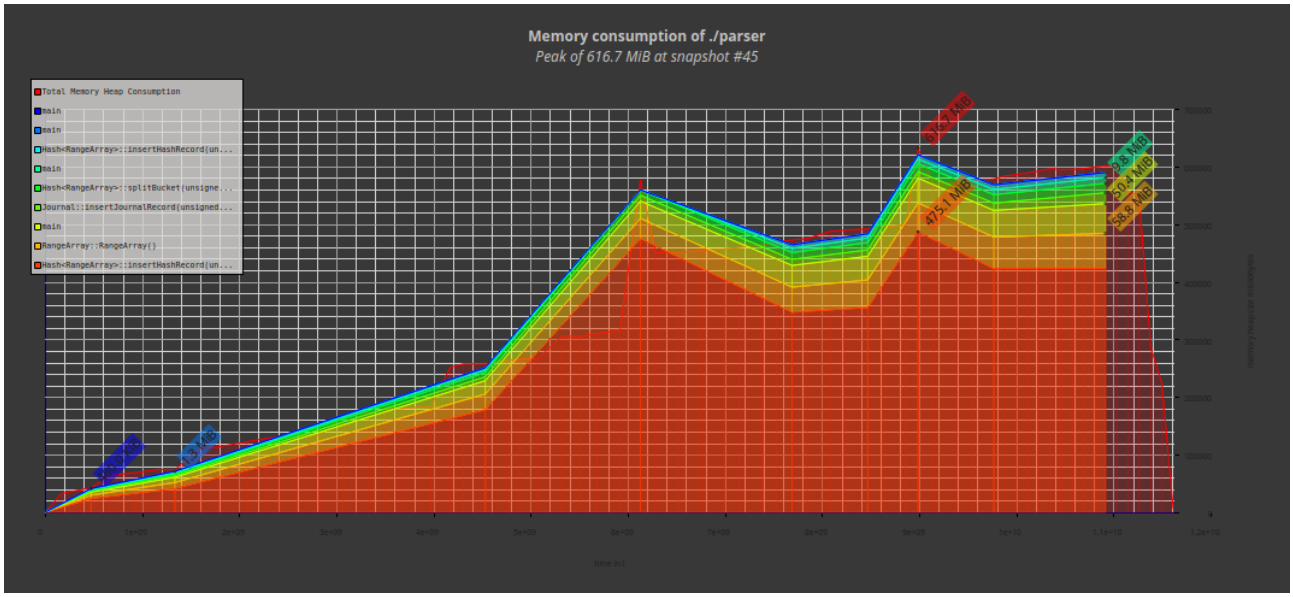
Το πρόγραμμα δοκιμάστηκε σε υπολογιστή i7 930 2.8Ghz με 8gb DDR3 RAM αλλά μέσα από virtualbox οπότε τα αποτελέσματα μπορεί να μην είναι αντιπροσωπευτικά του τρεξίματος χωρίς virtual περιβάλλον. Οι χρόνοι είναι μέσος όρος 10 εκτελέσεων. Η χρήση μνήμης βγήκε με τη χρήση του tool massif του valgrind.

Execution Times	**./acid -l1**	**./acid -l2 -t**	**./acid -l3 -t -th4**
Small dataset	4,9622667 sec	7,44428093 sec	4,3922172 sec
Medium dataset	37,642041 sec	113,080061 sec	22,5677646 sec

./acid -l1 (small dataset)

./acid -l1 (medium dataset)

./acid -l2 -t (small dataset)



./acid -l2 -t (medium dataset)

./acid -l3 -t -th4 (small dataset)

./acid -l3 -t -th4 (medium dataset)