HY586 - Κατανεμημένος Υπολογισμός
Διδάσκουσα: Παναγιώτα Φατούρου

3ο Σετ Ασκήσεων

Προθεσμία Παράδοσης: Δευτέρα 7/12/09, στο μάθημα

Άσκηση 1

1. Εξηγήστε που θα υπήρχε πρόβλημα στην απόδειξη ορθότητας του αλγορίθμου UnboundedSnapshot, αν αφαιρούσαμε το πεδίο tag από κάθε καταχωρητή R_i, καθώς και τον κώδικα που αφορά το πεδίο αυτό από τον αλγόριθμο.
2. Consider a modification of the UnboundedSnapshot algorithm in which process p_i increments R_i.tag when it performs a scan operation, as well as when it performs an update operation. (The val and view fields of register R_i are not changed, and the embedded scan operation is not modified in any way.)

   Is the modified algorithm still correct? Either prove that it is or give a counterexample execution. Assume that the increase of R_i.tag by p_i's SCAN is performed just before the SCAN returns (i.e., after the computation of the vector that the SCAN returns).
3. i) Θεωρήστε την εξής παραλλαγή του αλγορίθμου UnboundedSnapshot:

UPDATE στο τμήμα A_i με τιμή v:
increment p_i's tag;
write(R_i, <v, tag, r_i.view>);
view := SCAN;
write(R_i,<v, tag,view>);

SCAN από τη διεργασία p:
repeatedly execute set of reads, in each of which all registersR_1, …, R_n are read until see:

1. either the same values in R_1, …, R_n in two consecutive sets of reads (then, return the vector of the val fields of R_1, .., R_n read during the second set of reads), or
2. three different values in some register R_i (return the view field of R_i read the third time);

Είναι ο παραπάνω αλγόριθμος σωστή υλοποίηση ενός snapshot object; Αν ναι, παρουσιάστε απόδειξη, αν όχι παρουσιάστε αντιπαράδειγμα στο οποίο να καταστρατηγείται η συνέπεια του αλγορίθμου.

ii) Θεωρήστε την εξής παραλλαγή του αλγορίθμου UnboundedSnapshot. Στην έκδοση αυτή, κάθε καταχωρητής περιέχει εκτός από το tag field και ένα sq field (sequence number) που συνοδεύει το view field και εκφράζει το sequence number του διανύσματος view που είναι αποθηκευμένο στον καταχωρητή R_i.

Είναι ο παραπάνω αλγόριθμος σωστή υλοποίηση ενός snapshot object; Αν ναι, παρουσιάστε απόδειξη, αν όχι παρουσιάστε αντιπαράδειγμα στο οποίο να καταστρατηγείται η συνέπεια του αλγορίθμου.

UPDATE στο τμήμα A_i με τιμή v:
increment p_i's tag;
write(R_i, <v, tag, r_i.view, R_i.sq>);
view := SCAN;
write(R_i,<v, tag,view, tag>);

SCAN από τη διεργασία p:
repeatedly execute set of reads, in each of which all registers R_1, …, R_n are read until you see:

1. either the same values in R_1, …, R_n in two consecutive sets of reads (then, return the vector of the val fields of R_1, .., R_n read during the second set of reads), or
2. three different values in some register R_i (let sq be the value of R_i.sq read the 3rd time, and let tag be the value of Ri.tag read the 3rd time)

If (sq >= tag) then
return the view field of R_i read the third time;
else continue; // with the repeated sets of reads

iii) Θα ήταν ο αλγόριθμος του ερωτήματος ii) σωστός αν η εντολή "If (sq >= tag) then" στον κώδικα της SCAN αντικατασταθεί από την "If (sq == tag) then";

Άσκηση 2

Θεωρήστε ένα ατομικό στιγμιότυπο μνήμης το οποίο επιτρέπει την εκτέλεση μιας μόνο λειτουργίας SCAN κάθε χρονική στιγμή (δηλαδή δεν επιτρέπεται να είναι περισσότερες από μια SCAN ενεργές ταυτόχρονα). Σας δίνεται η παρακάτω υλοποίηση ενός τέτοιου ατομικού στιγμιοτύπου μνήμης (atomic snapshot).

shared registers seq, V[m][∝]; // *initially, seq = 0, V[1][j] = ⊥ και V[i][j] = null, 1≤j≤m, ∀i>1*

```
void UPDATE(int j , value v) {
// UPDATE της συνιστώσας j με την τιμή v
    int lseq;

    lseq = seq;
    V[lseq][j] = v;
}
```

```
value *SCAN(void) {
    value a[m];
    int lseq, l;

    lseq = seq + 1;
    seq = lseq;
    for j = 1 to m do
        k = lseq;
        repeat {
            k = k – 1;
            a[j] = V[k][j];
        } until (a[j] ≠ NULL);
    return a;
}
```

Η υλοποίηση χρησιμοποιεί έναν δισδιάστατο πίνακα με μη-πεπερασμένο πλήθος γραμμών και m στήλες (μία για κάθε συνιστώσα).

Στην υλοποίηση αυτή, κάθε φορά που εκτελείται μια SCAN αυξάνεται ένας μετρητής (sequence number), ο οποίος αποθηκεύεται στον καταχωρητή seq. Παρατηρήστε ότι ο seq εγγράφεται μόνο από SCANS και διαβάζεται μόνο από UPDATES. Αφού μία μόνο SCAN

είναι ενεργή κάθε χρονική στιγμή, η ακολουθία τιμών που αποθηκεύονται στο seq είναι αύξουσα και κάθε SCAN χαρακτηρίζεται από το sequence number που εγγράφει στο seq.

Μια UPDATE U στη συνιστώσα A_j με τιμή v εγγράφει τη v σε κάποιον από τους καταχωρητές της στήλης j του πίνακα V. Η γραμμή του V στην οποία πρέπει να γράψει η U καθορίζεται από την τιμή που η U διάβασε στον seq (δηλαδή οι SCANS καθορίζουν που θα γράψουν οι UPDATES).

Μια SCAN S, η οποία γράφει s στο seq, διαβάζει, για κάθε j, τους καταχωρητές της στήλης j ξεκινώντας από αυτόν της γραμμής s-1 προς χαμηλότερες γραμμές. Η S επιστρέφει για τη συνιστώσα j την πρώτη τιμή ≠ NULL που βρίσκει σε αυτούς τους καταχωρητές.

Αποδείξτε ότι η παραπάνω υλοποίηση είναι μια σωστή υλοποίηση ατομικών στιγμιοτύπων. Πιο συγκεκριμένα, τοποθετήστε σημεία σειριοποίησης στις SCAN και στις UPDATE και αποδείξτε ότι (1) για κάθε λειτουργία το σημείο σειριοποίησης της είναι μέσα στο διάστημα εκτέλεσής της, καθώς και (2) πως οι SCAN επιστρέφουν συνεπή διανύσματα τιμών.

Παρατήρηση: Είναι σημαντικό στην απόδειξη της ορθότητας του αλγορίθμου πως μόνο μια SCAN μπορεί να είναι ενεργή κάθε χρονική στιγμή.