Parallel Haskell

Ιάσονας Νικολάου
ΑΜ: 03116129

Ακ. Έτος 2020-2021 - Ροή Λ

Εισαγωγή

Στην παρούσα εργασία θα πειραματιστούμε με την parallel Haskell. Αρχικά, θα υλοποιήσουμε ένα σειριακό πρόγραμμα το οποίο υπολογίζει διωνυμικούς συντελεστές $C(n,k) = \frac{n!}{k!(n-k)!}$. Στην συνέχεια, θα λύσουμε το ίδιο πρόβλημα υλοποιώντας δύο παράλληλα προγράμματα με την χρήση τού Par monad και θα συγκρίνουμε τις επιδόσεις τους.

Σειριακό Πρόγραμμα

Επειδή ο αριθμός C(n, k) μπορεί να είναι πολύ μεγάλος, υπολογίζουμε το: C(n, k) mod P, όπου P ένας μεγάλος πρώτος αριθμός. Πιο συγκεκριμένα έχουμε:

$$C(n,k) = \frac{n!}{k!(n-k)!} = \frac{n*(n-1)*...*(n-k+1)}{k!} \equiv n*(n-1)*...*(n-k+1)*(k!)^{P-2} mod P$$

Στην τελευταία ισότητα χρησιμοποιήθηκε το μικρό θεώρημα του Fermat για να υπολογίσουμε το $\frac{1}{k!} mod P$. Στο σειριακό πρόγραμμα (binom.hs) χρησιμοποιήσαμε την ιδιότητα:

$$(a*b) \mod P = (a \mod p)*(b \mod p) \mod p$$

καθώς και επαναλαμβανόμενο τετραγωνισμό για να υπολογίσουμε το $(k!)^{P-2}$. Για κάθε query C(n, k) καλούμε την συνάρτηση binom, η οποία υπολογίζει το ζητούμενο αποτέλεσμα.

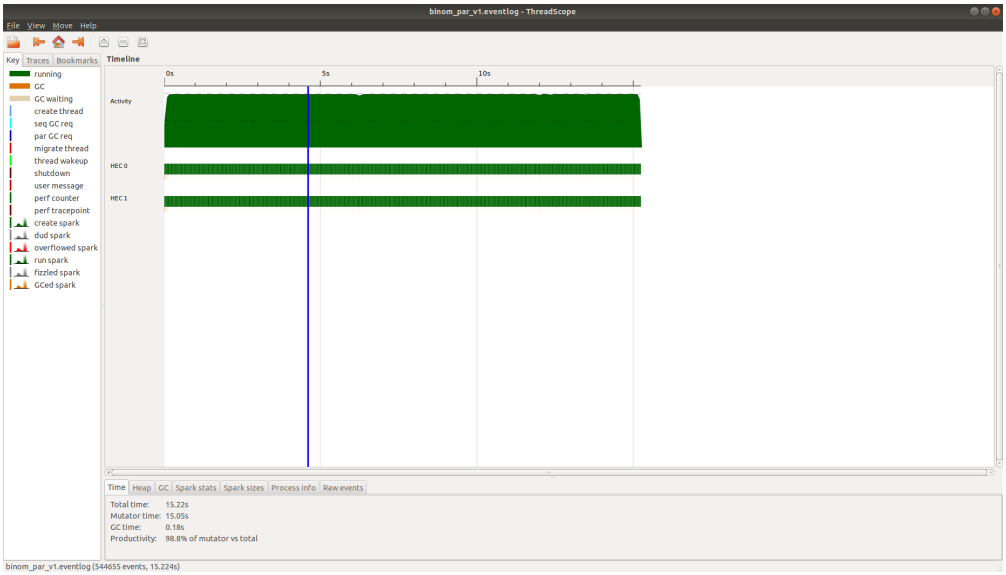
Παράλληλα Προγράμματα

Έχουμε υλοποιήσει τα ακόλουθα παράλληλα προγράμματα:

- binom_par_v1: Αυτό το πρόγραμμα είναι πανομοιότυπο με το σειριακό, αλλά η συνάρτηση binom, η οποία υπολογίζει τον ζητούμενο διωνυμικό συντελεστή, καλείται παράλληλα. Έτσι, μπορούμε να υπολογίζουμε πολλά query C(n, k) παράλληλα.
- binom_par_v2: Σε αυτό το πρόγραμμα έχουμε παραλληλισμό όχι μόνο στην κλήση τής συνάρτησης binom αλλά και εντός τής συνάρτησης. Πιο συγκεκριμένα, υπολογίζουμε παράλληλα τον αριθμητή $n*(n-1)*...*(n-k+1) \mod P$ και τον παρανομαστή $(k!)^{P-2} \mod P$.

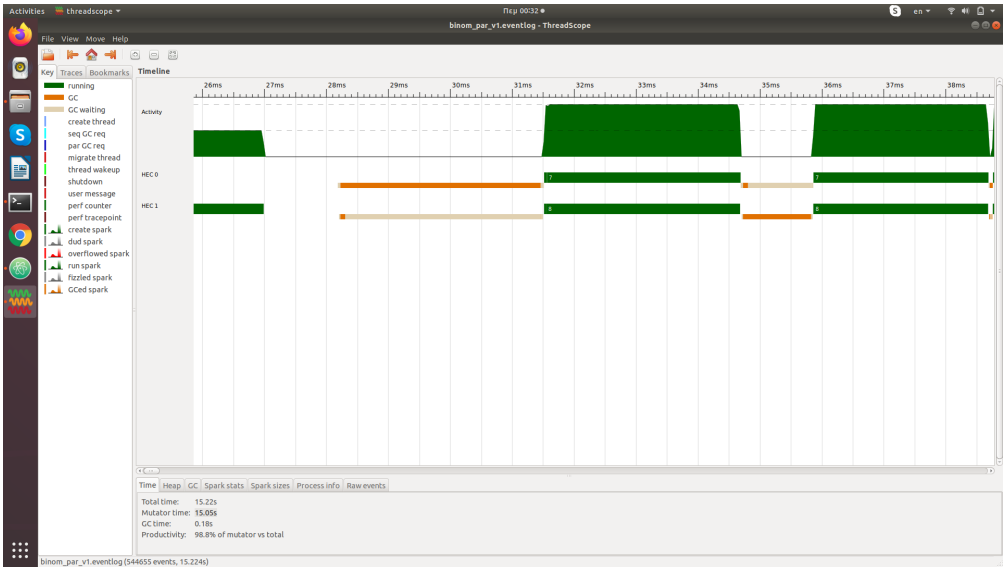
Επιδόσεις

Για την μελέτη και την σύγκριση των επιδόσεων των προγραμμάτων θα χρησιμοποιήσουμε το εργαλείο threadscope, καθώς και testcases διαφόρων μεγεθών. Ο υπολογιστής μου διαθέτει Inter Core i7-6500U CPU, ο οποίος είναι διπύρηνος.
Αρχικά, θα υπολογίσουμε το θεωρητικό μέγιστο speedup που μπορούμε να πετύχουμε με 2 πυρήνες. Για τον σκοπό αυτό θα χρησιμοποιήσουμε ένα testcase με 1000 queries. Τρέχοντας το σειριακό πρόγραμμα binom.hs μερικές φορές υπολογίζουμε ότι ολοκληρώνεται σε μέσο χρόνο 30.382 s. Τρέχοντας το παράλληλο πρόγραμμα binom_par_v2 και χρησιμοποιώντας το threadscope μπορούμε να υπολογίσουμε το μέρος τού προγράμματος που παραλλαληλοποιείται. Γνωρίζουμε, για παράδειγμα ότι η ανάγνωση τού input γίνεται σειριακά. Σημειώνουμε ότι η ανάγνωση τού input γίνεται ρητά στην αρχή τού προγράμματος μέσω τής εντολής: evaluate (length input)



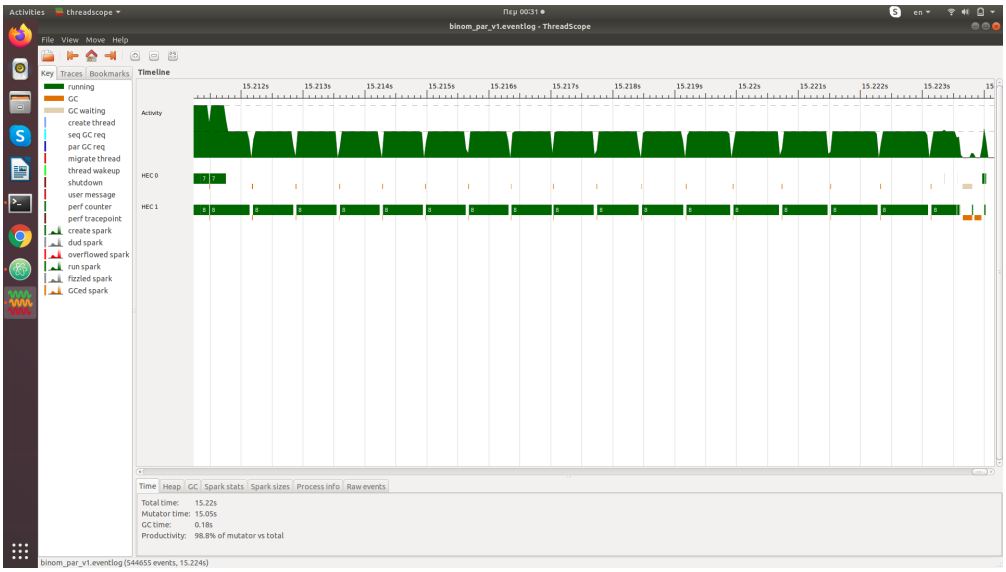
Σχήμα 1: threadscope

Στην αρχή και στο τέλος τής εκτέλεσης παρατηρούμε ότι δεν χρησιμοποιούνται και οι 2 επεξεργαστές. Πράγματι, μεγεθύνοντας τις 2 περιοχές μπορούμε να το επιβεβαιώσουμε. Στα πρώτα 35 ms παρατηρούμε ότι διαβάζεται το input.



Σχήμα 2: threadscope: start of the program

Επίσης, στα τελευταία 15ms τού προγράμματος χρησιμοποιείται και πάλι μόνο ο ένας επεξεργαστής.



Σχήμα 3: threadscope end of the program

Χρησιμοποιώντας τον νόμο τού Amdahl υπολογίζουμε το μέγιστο speedup:

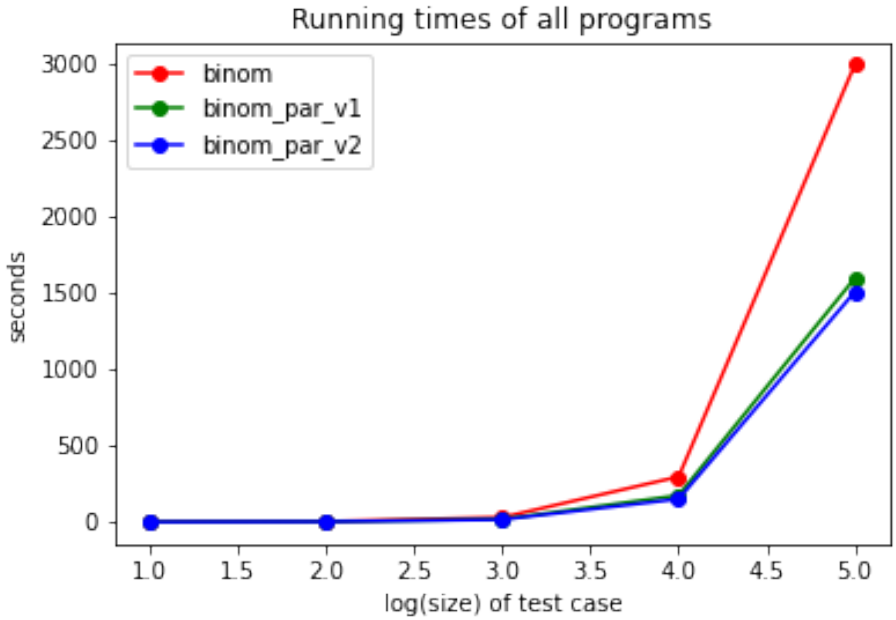
$$speedup = \frac{1}{(1-P) + \frac{P}{N}} = 1.996 \approx 2$$

, όπου $P = \frac{30.382 - 0.05}{30.082} = 0.998$ είναι το ποσοστό τού προγράμματος που παραλληλοποιείται και $N = 2$ ο αριθμός των διαθέσιμων επεξεργαστών.
Τρέξαμε και τα 3 προγράμματα (binom, binom_par_1, binom_par_2) για testcases με 10, 100, 1000, 10000 και 100000 queries. Στα testcase περιορίσαμε τα queries στο εύρος $0 \leq K \leq N \leq 10^6$, καθώς για μεγαλύτερες τιμές τα προγράμματα καθυστερούσαν πολύ. Παρ'όλα αυτά, χρησιμοποιήσαμε και ένα testcase (input10_big.txt) με 10 queries και μεγαλύτερους αριθμούς στο εύρος $0 \leq K \leq N \leq 10^9$, το οποίο παρουσιάζεται στο τέλος.

log(size)	binom	binom_par_v1	binom_par_v2
1	0.456	0.450	0.351
2	3.856	2.275	1.884
3	30.382	17.586	15.842
4	298.042	174.270	150.601
5	∼3000	1599.310	1509.719

Πίνακας 1: completion times

Τα αποτελέσματα συνοψίζονται στον ακόλουθο πίνακα.
Παρατηρούμε ότι το πρόγραμμα binom_par_1, αν και βελτιώνει σημαντικά την επίδοση τού απλού σειριακού προγράμματος, δεν φτάνει το θεωρητικό speedup = 2. Αντίθετα, πολύ κοντά στο θεωρητικό speedup βρίσκεται το πρόγραμμα binom_par_2.
Στα μικρά testcases μέσω τού παραλληλισμού γλιτώνουμε λίγα δευτερόλεπτα. Ωστόσο, στα μεγαλύτερα testcases εξοικονομούμε σημαντικά περισσότερο χρόνο. Αυτό φαίνεται και στην ακόλουθη γραφική παράσταση.



Τέλος, τρέχοντας τα 3 προγράμματα με το testcase input10_big.txt, το οποίο περιέχει queries με μεγαλύτερους αριθμούς παρατηρούμε παρόμοια συμπεριφορά. Πιο συγκεκριμένα, οι χρόνοι ολοκλήρωσης κάθε προγράμματος είναι οι εξής:

log(size)	binom	binom_par_v1	binom_par_v2
10	263.133	154.048	133.573

Πίνακας 2: input10_big.txt completion times

Παρατηρούμε ότι τα παράλληλα προγράμματα και κυρίως το binom_par_v2 σχεδόν υποδιπλασιάζει τον απαιτούμενο χρόνο.