Γλώσσες Προγραμματισμού II

Άσκηση 6:
Παραλληλισμός και Ταυτοχρονισμός στη Haskell

Δανάη Ευσταθίου, 10ο εξάμηνο

ΑΜ : 03115122

Περιγραφή

Για την άσκηση αυτή ζητείται η ανάπτυξη κώδικα σε Haskell για παραλληλοποίηση ή ταυτοχρονισμό του υπολογισμού του $\binom{n}{k} \mod p$, κάνοντας ελεύθερη χρήση των δοσμένων μεθόδων. Στη συγκεκριμένη περίπτωση αναπτύχθηκαν τρία προγράμματα:

1. Το par_combs, το οποίο χρησιμοποιεί το Strategy parList για τη δημιουργία sparks για κάθε κλήση της γενικής συνάρτησης υπολογισμού του $\binom{n}{k} \mod p$.
2. Το more_par_combs, το οποίο χρησιμοποιεί το Strategy parList για τη δημιουργία sparks για κάθε κλήση της γενικής συνάρτησης υπολογισμού του $\binom{n}{k} \mod p$, καθώς και για κάθε κλήση της συνάρτησης υπολογισμού του παραγοντικού.
3. Το conc_combs, το οποίο χρησιμοποιεί την forkIO για τη δημιουργία ξεχωριστού thread για καθένα υπολογισμό του $\binom{n}{k} \mod p$.

Στα διαγράμματα που ακολουθούν φαίνεται το speedup που επιτυγχάνει καθένα από αυτά τα προγράμματα με τη χρήση 1, 2 ή 4 πυρήνων, σε καθένα από 3 tests εισόδου που δημιουργήθηκαν για αυτόν τον σκοπό.
Στο πρώτο test εισόδου (test) δίνεται προς υπολογισμό 10 φορές το ίδιο $\binom{n}{k} \mod p$, δύσκολο ως προς τον υπολογισμό, έτσι ώστε στην εκτέλεση του par_combs να είναι ξεκάθαρος ο τρόπος ανάθεσης των sparks στους πυρήνες.
Στο δεύτερο test εισόδου (test2) δίνονται 36 συνδυασμοί με κλιμακούμενη δυσκολία και 8 επίπεδα δυσκολίας, με το πρώτο επίπεδο να αποτελείται απο 8 συνδυασμούς και τα υψηλότερα από 4 το καθένα.
Στο τρίτο test εισόδου (test3) δίνονται 500 τυχαίοι συνδυασμοί προς υπολογισμό.
Για την δημιουργία των διαγραμμάτων αναπτύχθηκε σειριακή έκδοση του προγράμματος υπολογισμού των συνδυασμών (combs), οι χρόνοι εκτέλεσης του οποίου δίνονται για κάθε test εισόδου στον παρακάτω πίνακα:

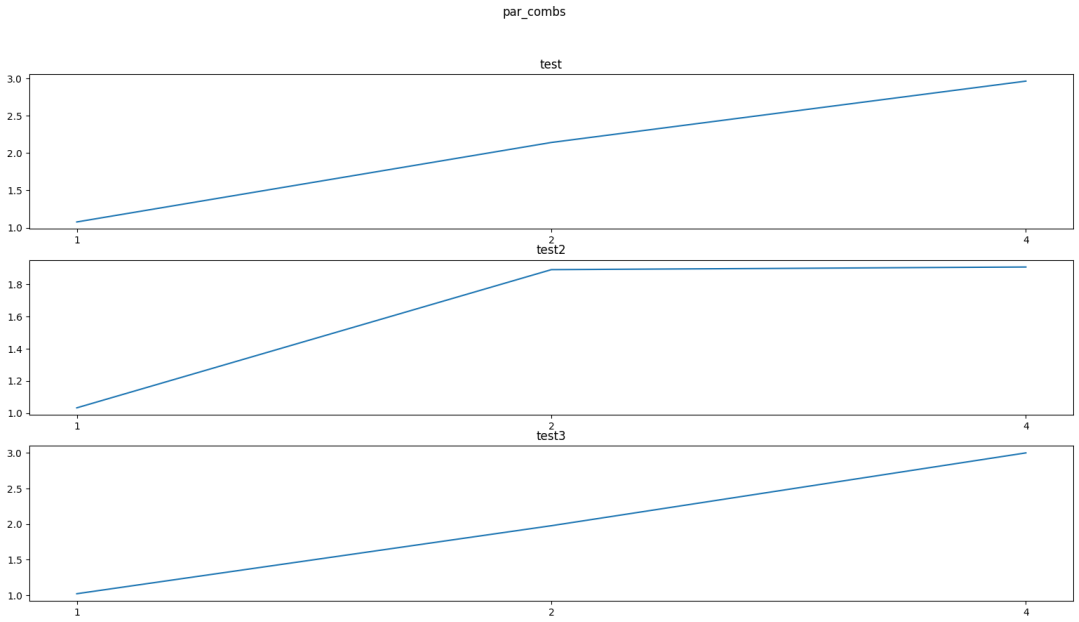
test	test2	test3
8.498	2.192	37.686

Πρόγραμμα par_combs

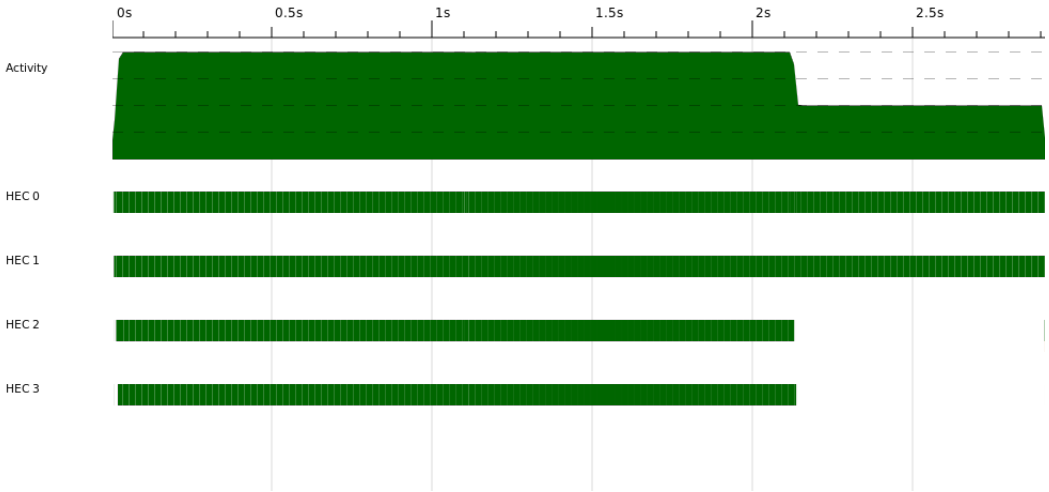
Χρόνοι εκτέλεσης του προγράμματος par_combs σε sec

είσοδος	1 πυρήνας	2 πυρήνες	4 πυρήνες
test	7.896	3.968	2.867
test2	2.122	1.159	1.149
test3	36.891	19.058	12.558

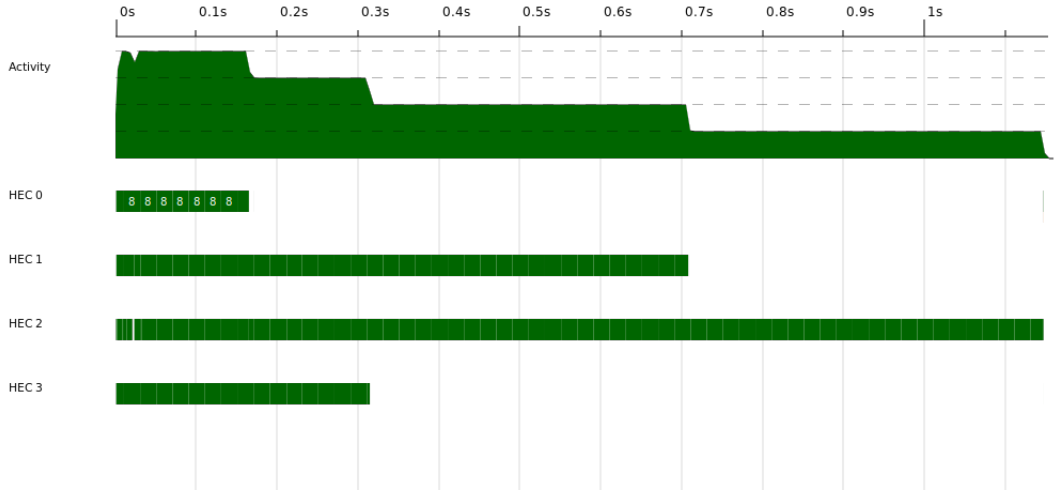
Διάγραμμα πυρήνων-speedup του προγράμματος par_combs



Από τον 1 στους 2 πυρήνες βλέπουμε να γίνεται σε όλα τα tests διπλασιασμός του speedup, πράγμα αναμενόμενο, αφού καθένα από τα 3 tests είναι σχεδιασμένο με τέτοιο τρόπο, ώστε να μπορούν τα επιμέρους testcases να κατανέμονται ομοιόμορφα στους 2 πυρήνες. Ανίθετα, από τον 1 στους 4 πυρήνες βλέπουμε τριπλασιασμό του speedup στο πρώτο και στο τρίτο test εισόδου και όχι τετραπλασιασμό αυτού. Αυτό μπορεί να οφείλεται στο ότι τα testcases δεν κατανέμονται ομοιόμορφα στους 4 πυρήνες, όπως για παράδειγμα στο test, στο οποίο όλα τα testcases είναι 10 στο πλήθος και ίδια, επομένως στους 4 πυρήνες θα υπολογιστούν με τη σειρά δύο τετράδες testcases και τέλος η υπολειπόμενη δυάδα, πετυχαίνοντας έτσι τον τριπλασιασμό του speedup. Ο ισχυρισμός αυτός μπορεί να φανεί και με την χρήση του εργαλείου threadscope:



Στο δεύτερο test εισόδου βλέπουμε από την άλλη σχεδόν σταθερό speedup, το οποίο λογικά οφείλεται στη μεγαλύτερη δυσκολία υπολογισμού ενός testcase έναντι των άλλων. Με τη χρήση του threadscope η θεωρία αυτή επιβεβαιώνεται:

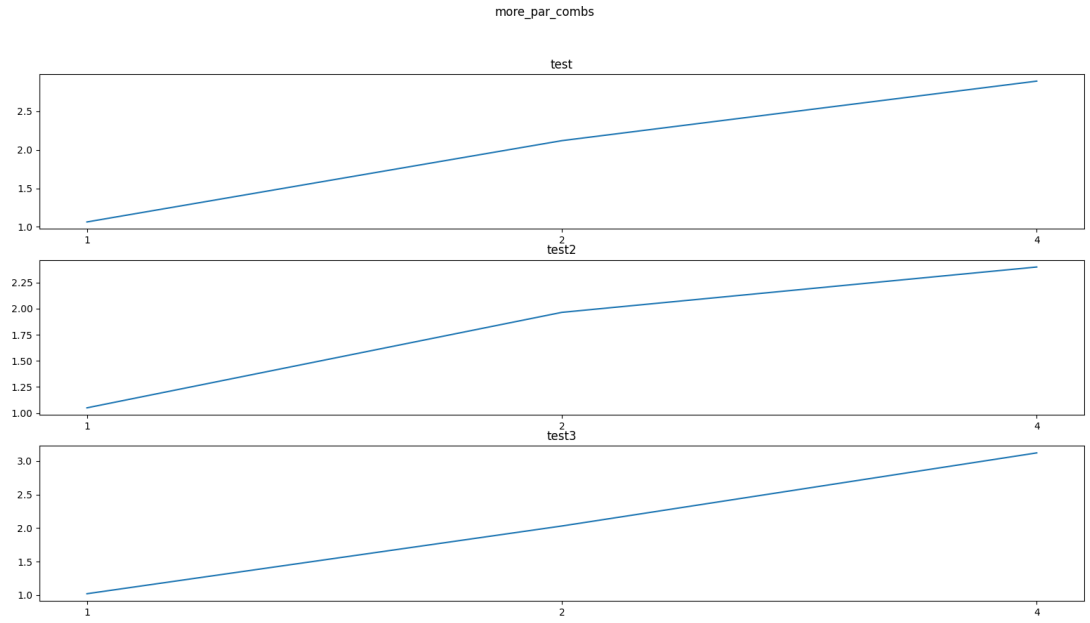


Πρόγραμμα more_par_combs

Χρόνοι εκτέλεσης του προγράμματος more_par_combs σε sec

είσοδος	1 πυρήνας	2 πυρήνες	4 πυρήνες
test	7.989	4.012	2.940
test2	2.087	1.116	0.914
test3	36.902	18.553	12.073

Διάγραμμα πυρήνων-speedup του προγράμματος more_par_combs



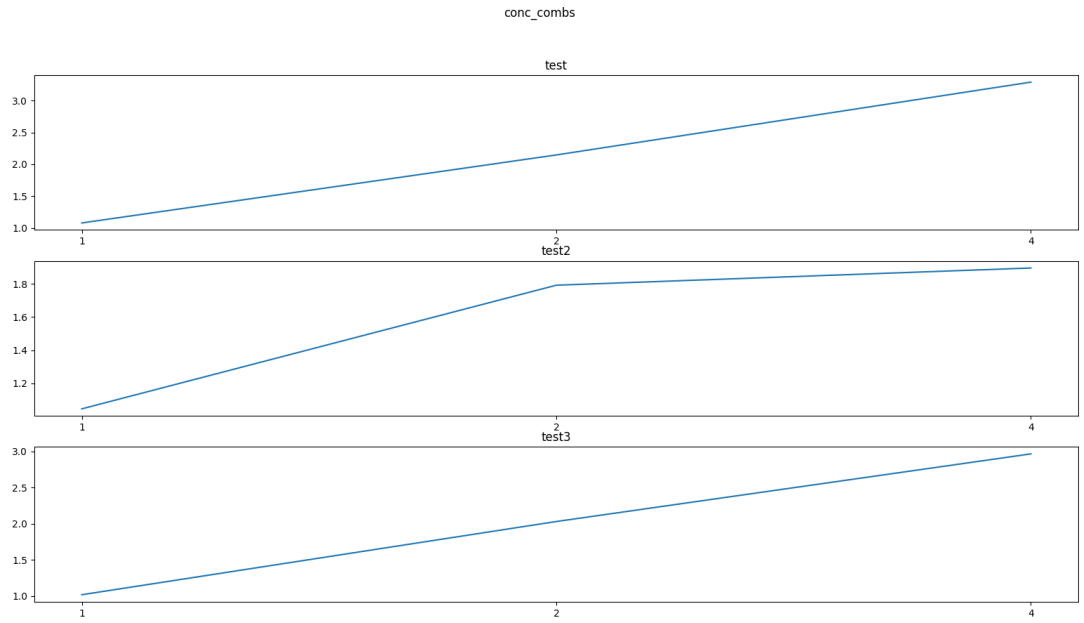
Εδώ, έγινε παραλληλισμός και σε καθέναν υπολογισμό του παραγοντικού, προκειμένου να βελτιωθεί η συμπεριφορά για το test2, πράγμα το οποίο φαίνεται ότι κατορθώθηκε, χωρίς όμως και πάλι να επιτευχθεί ο τετραπλασιασμός του speedup στους 4 πυρήνες, πιθανώς για τους ίδιους λόγους με παραπάνω. Η συμπεριφορά στις υπόλοιπες περιπτώσεις είναι παρόμοια με αυτή του προγράμματος par_combs.

Πρόγραμμα conc_combs

Χρόνοι εκτέλεσης του προγράμματος conc_combs σε sec

είσοδος	1 πυρήνας	2 πυρήνες	4 πυρήνες
test	7.989	4.012	2.940
test2	2.087	1.116	0.914
test3	36.902	18.553	12.073

Διάγραμμα πυρήνων-speedup του προγράμματος conc_combs



Εδώ χρησιμοποιήθηκαν threads για τον υπολογισμό καθενός συνδυασμού, έναντι της παραλληλοποίησης του par_combs. Όπως είναι αναμενόμενο, η συμπεριφορά του conc_combs είναι όμοια με εκείνη του par_combs, με τη μόνη διαφορά ένα πολύ μικρό προστιθέμενο overhead στην περίπτωση των threads, λόγω του τρόπου δημιουργίας και δρομολόγησής τους.