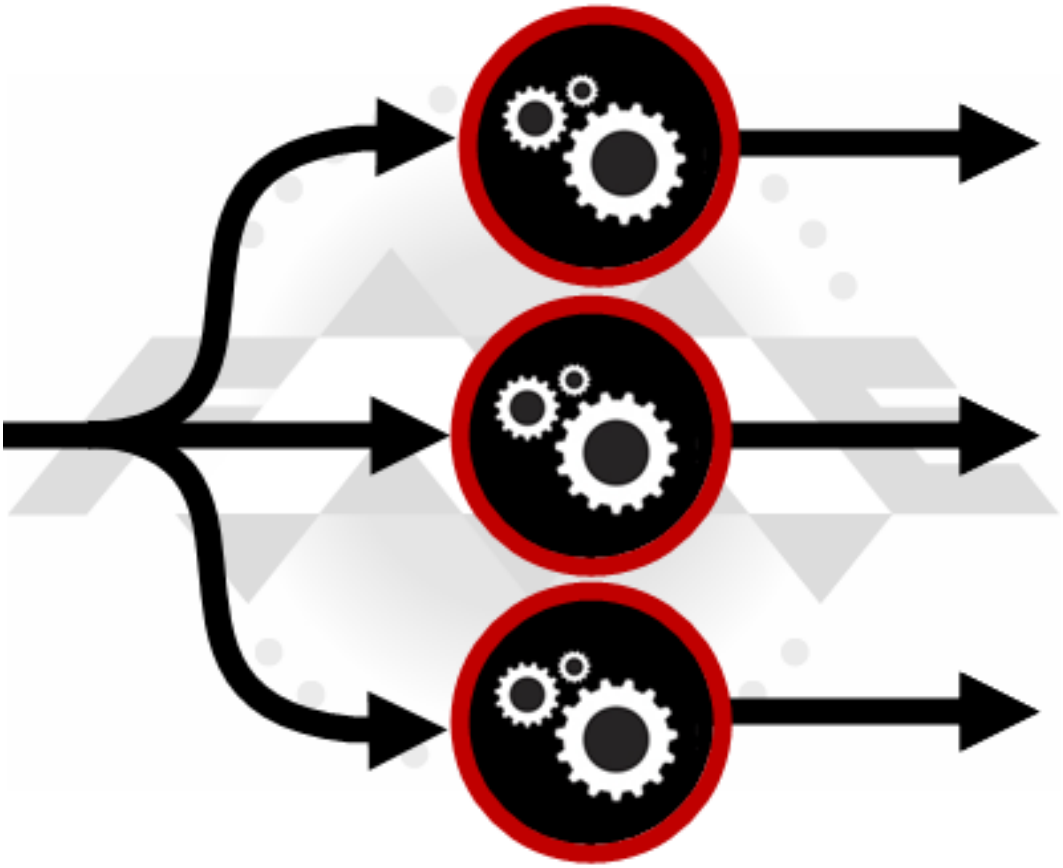

Εργασία 4 2021-2022

“A comparative study of multi-threading systems: pthreads, openCilk and multiCilk”

| Παράλληλα & Διανεμημένα Συστήματα |

Κωνσταντινίδης Κωνσταντίνος
AEM:9162, email: konkonstantinidis@ece.auth.gr
Θεσσαλονίκη, 25/12/2021

Παρουσίαση της μελέτης

Η παρούσα εργασία αποτελεί μια μελέτη των τριών συστημάτων παραλληλισμού pthreads, openCilk, multiCilk.

Για να γίνει αυτό, επιλέχθηκαν δύο προβλήματα επάνω στα οποία έχει υλοποιηθεί αρχικά η σειριακή λύση και έπειτα παραλληλοποιήθηκε κάνοντας χρήση καθενός εκ των τριών προαναφερθέντων συστημάτων.

Σε κάθε περίπτωση, ελέγχεται η εγκυρότητα της εξόδου των προγραμμάτων και μετράται ο χρόνος εκτέλεσης τους.

Τα προβλήματα

Για τους σκοπούς αυτής της μελέτης επιλέχθηκαν τα εξής δύο προβλήματα.

Πρώτον, έχουμε:

- Υπολογισμός του ολοκληρώματος μιας συνάρτησης f(R) → R μέσω αριθμητικής ολοκλήρωσης με βάση τον σύνθετο κανόνα τραπεζίου.

  Ειδικότερα, ολοκληρώνεται ενδεικτικά η $f(x) = x^3$ στο διάστημα [0,50000] με 1.000.000 διαμερίσεις του διαστήματος. Με την επιλογή αυτού του προβλήματος προβλέπεται να αναδειχθεί αφενός η χρονική ανωτερότητα των παράλληλων υπολογιστικών μεθόδων, καθώς και μία πρωταρχική ταξινόμηση μεταξύ τους όσον αφορά την αποδοτικότητα τους. Ωστόσο, εκτιμάται ότι η απλότητα του προβλήματος είναι τέτοια που δεν θα επιτρέψει να φανούν σημαντικές διαφορές μεταξύ των τριών συστημάτων παραλληλισμού, καθώς ο κύριος διαφοροποιητικός παράγοντας (load balancing) δεν παρουσιάζει κάποια εις βάθος πολυπλοκότητα, είναι δηλαδή εύκολο να φροντίσουμε να έχουν όλα τα νήματα ίσο φόρτο εργασίας και στην περίπτωση των pthreads (όπου αυτό πρέπει να γίνει από τον προγραμματιστή). Επομένως, δεν δίνεται η ευκαιρία στο run-time work-stealing της openCilk να αναδειχθεί.

Για αυτόν τον λόγο, εισάγεται το δεύτερο πρόβλημα:

- Quicksort

Ειδικότερα, ταξινομείται ένα διάνυσμα 10.000.000 ακέραιων αριθμών. Όντως μία από τις βασικότερες προγραμματιστικές εφαρμογές, η quicksort επιλέχθηκε για να αναδείξει τη σημαντικότητα του load balancing όταν έχουμε να κάνουμε με παράλληλες υλοποιήσεις. Είναι σημαντικά πιο πολύπλοκο το πρόβλημα ανάθεσης ίσου φόρτου εργασίας σε όλα τα νήματα πλέον, και άρα όταν αυτό αφήνεται στον προγραμματιστή και όχι σε κάποιο έξυπνα εξειδικευμένο σύστημα, συνήθως υπάρχουν μικρότερα speed-up.

Παρουσίαση των αποτελεσμάτων

Ο αριθμός των νημάτων για τις pthreads υλοποιήσεις κάθε φορά διερευνάται και θέτεται ίσο με τον βέλτιστο (κοντά στον πραγματικό αριθμό των επεξεργαστών του υπολογιστικού συστήματος).

Εκτελώντας πολλαπλές εκτελέσεις για κάθε πρόγραμμα και παίρνοντας τον μέσο όρο των χρόνων εκτέλεσης (είτε προγραμματιστικά είτε χειροκίνητα), έχουμε τον εξής πίνακα και για τα δύο προβλήματα, όπου αναγράφονται ο μέσος χρόνος εκτέλεσης και το αντίστοιχο speed-up της εκάστοτε υλοποίησης έναντι της σειριακής:

Υλοποίηση	Μέσος Χρόνος Εκτέλεσης για το **πρόβλημα 1** (ms) [speedup]	Μέσος Χρόνος Εκτέλεσης για το **πρόβλημα 2** (ms) [speedup]
Serial	2.711501 [1]	758.900391 [1]
Pthreads	0.841321 [3.222]	613.162903 [1.237]
OpenCilk	0.844926 [3.209]	203.498245 [3.729]
MultiCilk	0.760514 [3.565]	168.407394 [4.506]

Πίνακας 1, Χρόνοι Εκτέλεσης και speed-up

Συμπεράσματα

Από τα πειραματικά δεδομένα του πίνακα 1, αρχικά εύκολα αντιλαμβανόμαστε την αυτονόητη χρησιμότητα των παράλληλων υλοποιήσεων, όταν μπορούμε να εξασφαλίσουμε speed-up έως και 4.5 έναντι της σειριακής.

Έπειτα, κυρίως μέσα από το πρόβλημα 2, φαίνεται η ανωτερότητα των openCilk και multiCilk υλοποιήσεων έναντι των αντίστοιχων pthreads υλοποιήσεων. Αυτό συμβαίνει γιατί σε πιο σύνθετα προβλήματα, όπου το load balancing είναι δυσκολότερο να καθοριστεί a priori, η work stealing δυνατότητα της Cilk αποδεικνύεται πολύ αποδοτική (και χωρίς καμία σχεδόν προστιθέμενη δυσκολία ή διαμεσολάβηση του προγραμματιστή).

Επιπλέον, όσον αφορά την σύγκριση μεταξύ της openCilk και της multiCilk, ακόμα και στην εφαρμογή της quicksort φαίνεται σε κάποιο βαθμό η ανωτερότητα του δεύτερου συστήματος να μπορεί να εκκινήσει πολλαπλά runtimes (αυξάνοντας έτσι το βάθος του παραλληλισμού κατά ένα επίπεδο, πρακτικά δημιουργώντας νήματα μέσα από νήματα) και τα χρονικά οφέλη αυτού. Ενδεχομένως σε προβλήματα ειδικότερης φύσης να είναι ακόμη πιο έντονη η διαφορά μεταξύ των δύο αυτών συστημάτων.

Τέλος, δεν αποκλείεται ωστόσο το ενδεχόμενο η χρήση της multiCilk να αποδειχθεί πιο δαπανηρή χρονικά για κάποια προβλήματα έναντι της OpenCilk, λόγω του επιπλέον overhead που υπάρχει. Αυτό υπάρχει και γενικά εξάλλου στον χώρο των παράλληλων υλοποιήσεων, με ένα σχετικό παράδειγμα να είναι το αξιοσημείωτο overhead εκκίνησης ενός pthread και η σύγκριση του με το όφελος που θα προκύψει από το pthread αυτό.

Παραδοτέο κώδικα

Ολόκληρος ο κώδικας, μαζί με το Makefile και όποια script χρησιμοποιήθηκαν για την παραγωγή βοηθητικών αρχείων, μπορεί να βρεθεί στο παρακάτω σύνδεσμο:

https://github.com/kon-konstantinidis/Parallel-And-Distributed-Systems/tree/main/exe4