Γιάννης Τζώρτζης
ΑΜ:2008030061

Αρχιτεκτονική παράλληλων και κατανεμημένων υπολογιστών

Project μαθήματος

Σκοπός:

Σκοπός της άσκησης αυτής είναι να υλοποιήσουμε έναν παράλληλο task server, ο οποίος θα δέχεται tasks και θα τα αναθέτει για εκτέλεση σε ένα σύνολο από workers. Ο task server θα είναι «ελαστικός», δηλαδή ο αριθμός των workers που θα χρησιμοποιεί θα εξαρτάται δυναμικά (στον χρόνο) από το συνολικό φόρτο εργασίας που υπάρχει.

Υλοποίηση:

Για καλύτερη οργάνωση και ευκολότερη κατανόηση του κώδικα, δημιουργήθηκαν τρία αρχεία πηγαίου κώδικα, τα οποία είναι τα *task_server.c*, *worker.c* και *queue.c*. Το πρώτο αρχείο περιέχει όλες εκείνες τις συναρτήσεις που έχουν να κάνουν με την δημιουργία των tasks, τη δημιουργία, το συγχρονισμό και κλείσιμο των threads κτλ. Το δεύτερο αρχεί περιλαμβάνει τις συναρτήσεις που υλοποιούν τα τρία είδη των tasks που περιγράφονται και στην εκφώνηση του project. Στο αρχείο queue.c υπάρχουν οι συναρτήσεις που σχετίζονται με τις λειτουργίες της ουράς, εισαγωγή, διαγραφή, εκτύπωση, αρχικοποίηση.

task_server.c:

- **void *worker_basic_function(void *t):** αυτή είναι η βασική συνάρτηση κάθε worker του προγράμματος και ουσιαστικά εκτελεί τα tasks.
- **void *aux_function():** η συνάρτηση αυτή είναι η βασική συνάρτηση του νήματος aux_thread, το οποίο ξυπνάει ανά τακτά διαστήματα Tp και ανάλογα με υπολογισμούς, δημιουργεί ή τερματίζει νήματα.

- **void *task_generator_handler():** η συνάρτηση αυτή αναλαμβάνει να δημιουργήσει νέα tasks και να τα αναθέσει σε ενεργά ή νέα threads.
- **void create_new_thread():** η συνάρτηση αυτή δημιουργεί ένα νέο thread και μια νέα ουρά για το thread αυτό.
- **void quit_signal_handler(int signum):** αυτή η συνάρτηση καλείται όταν ο χρήστης του προγράμματος επιθυμεί τον τερματισμό του και πατήσει Control-c.

worker.c:

- **void prime_numbers(int n):** υπολογίζει το πλήθος των πρώτων αριθμόν στο διάστημα 1..n, καθώς επίσης και το άθροισμα τους.
- **void memory_copy(int A, int B, int N):** αντιγράφει N στοιχεία από τη διεύθυνση A στην διεύθυνση B.

queue.c:

- **void print_queue(queue q):** εκτυπώνει τα στοιχεία της ουράς q
- **queue init_queue(queue q, long id):** αρχικοποιεί την ουρά q.
- **queue enqueue(queue q, int task):** εισάγει νέο task στην ουρά q
- queue dequeue(queue q): εξάγει ένα task από την ουρά q

Εκτέλεση:

- make
- ./task_server

Λεπτομέρειες:

Στο αρχείο task_server.h υπάρχουν οι μεταβλητές NUM_TASKS(αριθμός tasks για εκτέλεση), Tp(χρονικό διάστημα κατά το οποίο το aux_thread θα "κοιμάται"), Tc(χρονικό διάστημα κατά το οποίο το task_gen θα "κοιμάται"), MEAN_TAIL_THRESHOLD(όριο που ελέγχει το aux_thread, αν ξεπεραστεί τότε δημιουργείται ένα νέο thread), IDLE_TIME_THRESHOLD(όριο που αν ξεπεραστεί, τότε αφαιρείται ένα thread).

Αποτελέσματα:

NUM_TASKS = 100, Tp = 2000, Tc = 10, MEAN_TAIL_THRESHOLD = 4, IDLE_TIME_THRESHOLD = 0.0003

- Program excecution time: 0.120000
- Program excecution time: 0.180000
- Program excecution time: 0.130000

NUM_TASKS = 100, Tp = 2000, Tc = 100, MEAN_TAIL_THRESHOLD = 4, IDLE_TIME_THRESHOLD = 0.0003

- Program excecution time: 0.180000
- Program excecution time: 0.190000
- Program excecution time: 0.290000

NUM_TASKS = 100, Tp = 2000, Tc = 500, MEAN_TAIL_THRESHOLD = 4, IDLE_TIME_THRESHOLD = 0.0003

- Program excecution time: 0.170000
- Program excecution time: 0.160000
- Program excecution time: 0.150000

NUM_TASKS = 100, Tp = 5000, Tc = 500, MEAN_TAIL_THRESHOLD = 4, IDLE_TIME_THRESHOLD = 0.0003

- Program excecution time: 0.190000
- Program excecution time: 0.190000
- Program excecution time: 0.190000

NUM_TASKS = 100, Tp = 5000, Tc = 500, MEAN_TAIL_THRESHOLD = 2, IDLE_TIME_THRESHOLD = 0.0003

- Program excecution time: 0.320000
- Program excecution time: 0.370000
- Program excecution time: 0.320000

NUM_TASKS = 400, Tp = 2000, Tc = 500, MEAN_TAIL_THRESHOLD = 2, IDLE_TIME_THRESHOLD = 0.0003

- Program excecution time: 3.080000
- Program excecution time: 4.030000
- Program excecution time: 8.550000

NUM_TASKS = 400, Tp = 5000, Tc = 50, MEAN_TAIL_THRESHOLD = 2, IDLE_TIME_THRESHOLD = 0.0003

- Program excecution time: 17.610001

NUM_TASKS = 400, Tp = 2000, Tc = 10, MEAN_TAIL_THRESHOLD = 2, IDLE_TIME_THRESHOLD = 0.0003

- Program excecution time: 17.610001
- Program excecution time: 19.160000
- Program excecution time: 19.400000

Γενικά μπορούμε να παρατηρήσουμε ότι οι χρόνοι Tp και Tc δεν πρέπει να έχουν τεράστια διαφορά αλλά θα πρέπει να είναι συγρίσιμοι. Λογικό βέβαια αφού αν tasks παράγονται πολύ γρήγορα αλλά δημιουργούνται πολύ αργά νέα threads τότε το πρόγραμμα θα καθυστερεί. Ανάποδα αν tasks σημιουργόυνται πολύ αργά αλλά νέα threads πολύ γρήγορα τότε θα έχουμε μεγάλη αναμονή στα threads και άσκοπη δημιουργία. Επίσης καλό είναι όσο πιο μεγάλος είναι ο αριθμός των tasks τόσο πιο μικρό πρέπει να είναι το MEAN_TAIL_THRESHOLD.

Τέλος Αναφοράς