Buta Denisa-Elena

332CA

README

Solutia generala pe care am ales-o se poate rezuma prin paralelizarea mai multor operatii, care au dus la un timp de executie mai scurt si o optimizare a programului. Pentru a efectua acest lucru ne-am folosit de instrumente din biblioteca pthread.h astfel:

In fiserul tema1.c:

In prumul rand, dupa ce am adaugat biblioteca necesara, am creat o variabila de tip cores care lua valoarea celui de al treilea parametru introdus in linia de comanda si care reprezenta numarul de thread-uri pe care faceam paralelizarea, dupa care, ne-am creat o variabila de tip pointer var(structura declarata in genetic_algorithm.h care are rolul de a contine toti parametri pe care ii vom folosi mai departe astfel incat sa nu aveam varibile globale).Ne-am creat de asemenea si o bariera pe care am initializat-o si foiosit mai departe. In continuare intr-un for am dat valori acestei variabile(cu cores numar de elemente) si am creat thread-urile folosindu-ne de functia pthread_create, in parametri aparand functia paralelizata de noi, run_genetic_algorithm, aceasta la randul ei fara vreun parametru. Am folosit functia pthread_join pentru a uni rezultatele thread-urilor. In cele din urma am distrus bariera si am eliberat spatiul de memorie.

In fisierul genetic_algorithm.c:

-in functia compute_fitness_function:

Am paralelizat for-ul existent adaugand la parametri un start si un end pe care l-am calculat corespunzator in cele ce urmeaza

-in functia cmpfunc:

Problema la paralelizare era ca, de fiecare data cand apelam qsort, pentru comparare se apela cmpfunc unde era calculat suma pentru fiecare individ in parte si acest lucru ridica de cateva ori complexitatea qsort-ului, crescand de asemenea si timpul de executie. Pentru rezolvarea acesteia am precalculat sumele pentru fiecare individ in parte in functia run_genetic_algorithm.

-in functia run_genetic_algorithm:

In primul rand am convertit parametrul acesteia din void* in var* pentru a ne putea folosi de parametrii alesi de noi. Apoi, dupa initializarea fiecarui element din structura am trecut la paralelizarea propriu-zis. Am calculat cate un start si un end pentru fiecare for pe care il foloseam(in afara de acela unde se facea iteratia pentru fiecare generatie deoarece fiecare iteratie depindea de cea precedenta si nu se putea rula decat secvential aceasta parte) astfel ca acestea sa fie executate in paralel de thread-urile create in main si am folosit bariere oriunde am crezut ca este necesar astfel ca thread-urile sa aiba o executie corecta. Acolo unde nu am paralelizat (in qsort spre exemplu) ne-am asigurat ca numai unul dintre thread-uri executa operatia respectiva pentru a nu duce la probleme neprevazute cum ar fi suprascriere sortare gresita etc.