Ressource

TP 1: Parallel programming with Open MP and Pthreads

ParPro

HoussamEddine Zahaf
{prénom.nom}@univ-nantes.fr

Objectives: The goal of this practical lecture is to get used to shared-memory parallel programming through OpenMP and Pthreads.

The C-code shared with you represents a moving object inside a 2D area. It includes the different tools to control the moving object (`play_utils.c` and `play_utils.h`), to build the graphical interface (`sdl_utils.c` and `sdl_utils.h`) and finally the functional part that invokes the different functionalities (`threads.c` and `threads.h`). The main file spawns the main threads that handles the back-end and the front-end to simulate the moving object.

The goal of your work is to detect as fast as possible the position of the object and to be able to predict its next position, using pthreads, open mp parallel region and parallel loops.

The baseline version of the code includes two threads : `void *GUI(void *args)` and `void *generate_data(void *args)`. The first draws the object and will further draw a blue circle around the object when the latter is detected (Lines 61-64). The location of the moving object must be defined therefore through variables `is` and `js` for respectively the x-axis and y-axis coordinates. The second thread generates boolean data matrix (`data`). The matrix represents a binarized version of the surface area to monitor.

Finally you will will write several versions of `locate_object` a thread/functionality that will locate the thread.

1. Write a sequential version of the function `locate_object`, and spawn it as a *thread*.
2. parallelize this function using pthreads, OpenMP parallel region and Open MP parallel for.
3. Compute the speed up factor of every implementation
4. Improve the parallel version of the second question, by adding communications between the different threads.
5. Compute the speed up factor of every implementation
6. Improve the parallel version by predicting the next area where the object will be found and process a subpart of the area rather than the full area. Compute the speed up factor for your implementation.