PROGRAMAÇÃO PARALELA E CONCORRENTE

Actor Model

REPORT

Mestrado em Engenharia Informática
2023

Ana Teixeira - fc56336

Description of the overall architecture

This project utilizes the Actor Model to parallelize the Knapsack problem. The architecture contains four actors, each responsible for a specific task: measuring fitness, calculating the best individual, performing crossover and managing mutation. Additionally, there is an actor managing the start and end of each generation and another actor to initialize randomly the individuals in the population. The communication between actors occurs through messages, forming a sequential chain. Each actor sends a message to the actor responsible for the next step and when the final step (mutation) is completed a message is sent back to the first step now with a new population (because some steps require the entire population). Messages may send the current individual modified or the entire population modified. Diagram 1 visually represents the overall architecture of this program and the relation between actors and messages.

Rationale for choosing that particular architecture

The architecture was designed to parallelize steps without relying on shared memory. By sending messages containing a single individual, steps can proceed independently once completed for each individual, minimizing the wait time for the entire population to finish each step and improving the parallelization. This way, due to each actor being assigned to a specific task, the execution of multiple tasks simultaneously is allowed. Furthermore, this architecture allows scalability, as a new actor can be introduced without causing problems in the structure. Additionally, the use of messages to pass the information ensures a modular system, where each actor can grow independently without affecting the others.

Explanation of how the necessary synchrotion as implemented

Using the actor model, synchronization mechanisms were unnecessary due to having no shared memory. In the case of the actors that received a message for each individual, the concurrent linked queue mailbox ensures that messages are processed in the order they are received, so the first message received is the first message taken by the actor. This eliminates race conditions and avoids the need for synchronization.

Brief benchmark of this version against the sequential and the first assignment, and its explanation

For benchmarking, the actor model version, the sequential version and the Phaser using 4 threads version - best parallelization strategy discovered in assignment 1 - were executed 30 times. The results of each method's times are represented in the boxplot in figure 1. By the analysis of the boxplot, it is notable to see that the actor model version has the worst efficiency compared to the others: it has a median runtime value around 329 seconds, while the sequential median value is around 170 seconds and the phaser is around 119 seconds. This was expected because of the overhead generated by the excessive exchange of messages between actors. In several steps of the actor model, a new message for each individual modified in that step is sent, causing an increased overhead. Other messages contain the entire population resulting in an overhead on the mailbox and causing a reduction of speed.

Experiment setup

The CPU model of the machine used to execute the program is 1,6 GHz dual-core Intel Core i5 and has 2 cores and 4 threads.