Assignment #2: Coin Counter

Programação Paralela e Concorrente (2023/2024)

Author: Daniel Lopes – Fc56357

1. What was the parallelization strategy chosen?

The chosen parallelization strategy was Fork-Join with Granularity Control. To manage granularity, three types of conditions were implemented:

- MaxLevelCondition: This condition checks if the current level (or index) has reached or exceeded a specified limit. This condition is suitable for controlling the depth of recursion, in order to avoid the creation of too many tasks.
- QueuedTaskCondition: This condition controls the execution of tasks in parallel based on the number of tasks in the queue (RecursiveTask.getQueuedTaskCount()) of the Java concurrency framework, typically managed by the ForkJoinPool.
- SurplusCondition: This condition manages task execution in parallel based on the number of tasks waiting in the surplus queue (RecursiveTask.getSurplusQueuedTaskCount()) of the Java concurrency framework, also typically managed by the ForkJoinPool.

2. Exactly why did you choose that parallelization strategy?

The strategy was chosen due to the nature of the problem, which involves irregular and recursive calls. Utilizing these conditions ensures that all available threads are actively performing tasks, taking full advantage of parallel processing.

3. What results did you achieve? (Note: Presentation and Benchmarking will be evaluated)

CPU used: AMD Ryzen 7 3700X 8-Core Processor,16 Logic Processors, 3.6 GHz of Frequency.

Both the MaxLevelCondition and SurplusCondition resulted in substantial performance enhancements, achieving approximately 94.90% and 93.04% improvement, respectively. These improvements translated to significant speedups of 19.62 and 14.37, while the occupancy levels reached 1.226 and 0.898, respectively. This improvement is visualized in the PerformanceImprovement.webp graph available in the data folder. In the sequential approach, the average time taken was approximately 3.7 seconds. With MaxLevelCondition, the time was reduced to approximately 0.19 seconds, and with SurplusCondition, it was reduced to approximately 0.26 seconds. However, despite experimenting with various values, the QueuedTaskCondition did not yield any performance improvements. A boxplot was also created, comparing the best values achieved with MaxLevel and SurplusCondition against the sequential values, which can be found in the data folder (FinalBoxPlot.webp).

4. What fine-tuning did you do to improve the performance?

Fine-tuning to enhance performance involved extensive testing of all Granularity conditions with various values until the optimal configuration was identified. For SurplusCondition, the best value was determined to be 2, among those tested from 1 to 5. In the case of MaxLevelCondition, values ranging from 3 up to the size of the coins array were examined, and the best value found was 15. Even after running the parallel version 40 times, values such as 13, 15, and 17 occasionally stood out as the best-performing options. To identify the best values for both conditions, execution times were saved in CSV files, and the column with the lowest average value represented the optimal setting for each condition. For both MaxLevelCondition and SurplusCondition, box plots were created and can be found in the data directory to visualize how the code behaves with different condition limits.