**Project 1 Report – Fully Sequential vs Distributed Calculation of Finite Sum of Numbers**

**Course**: 103A-CSCSN-MSA-EPNM

**Period**: Summer Semester 2022/2023

**Professor**:

**Student**: Bruno Axel Kamere

**E-Mail**: bruno\_axel.kamere.stud@pw.edu.pl

**Date of Submission**: 27/03/2023

Table of Contents

[Question 3](#_Toc130856174)

[Answer 4](#_Toc130856175)

[1. Fully Sequential Calculation 4](#_Toc130856176)

[2. Distributed Calculation 5](#_Toc130856177)

[3. Fully Sequential vs Distributed Calculations 8](#_Toc130856178)

Question

1. Write a MATLAB m-file, which will calculate values of the following finite sum of numbers

in two versions:

a) fully sequential,

b) distributed (using the parfor loop).

2. Run both programs for different values of n, e.g., n = 1000000, n = 1000000000 and n = 10000000000.

3. Compare results of calculations and execution times for all versions.

4. Write the report on your work and results.

5. Save the report and your MATLAB files on server Studia, in module Reports, in report under the name

**Project 1**

Answer

The equation below was written in MATLAB both in a sequential form using for-loop and distributed form using the parfor loop.

1. Fully Sequential Calculation

The MATLAB code for the fully sequential calculation is:



The above code is split into 3 different for-loops that calculate the sum of numbers for different values of n; n = 1000000, n = 1000000000 and n = 10000000000.

In addition to calculating the sum of numbers, the execution time for each for-loop is also being calculated. Below are the results of the calculations.

Graphical user interface, table

Description automatically generated

As can be seen from the above results, the for-loop execution time (elapsed\_time\_n1[2,3] in seconds) increases as the value of n increases. The disadvantage of this type of calculation is that for resource hungry calculations, the execution time is longer since the calculations are sequential and not parallelized in any form whatsoever. This doesn’t take good use of a computer’s cores.

1. Distributed Calculation

In this case, the same scenario was implemented but in distributed manner using parfor loop. Below is the MATLAB code used:



In this case, the calculation is performed using the parfor loop for each value of n.

MATLAB issues and coordinates workers to perform the calculation in the loop body in parallel on the workers in a parallel pool. This allows the calculations to be performed much faster than using the sequential for-loop, since the calculation tasks are distributed on each of the provisioned workers.

The loop was executed two times. As will be observed, the execution time for n1 is significantly larger; and this is because on the first execution of the parfor loop, MATLAB spends some time initializing the workers in a parallel pool. The execution time significantly lowers though for each following parallel calculation.

Below are the results for the first calculations.

Graphical user interface

Description automatically generated with medium confidence

As can be seen from the above parallel loop calculations results; the elapsed\_time\_n1 takes approx. 93 seconds to execute, far longer than the following calculations where n is even larger. And this is due to what’s explained in the previous paragraph that MATLAB spends time initially initializing the parallel pool of workers.

Results for the second execution of the calculation code:

Graphical user interface, application

Description automatically generated with medium confidence

In the second execution of the parallel calculations, it can be observed that the execution time is significantly lower compared to the initial execution of the parallel calculations.

1. Fully Sequential vs Distributed Calculations

**Results comparison**:

|  |  |
| --- | --- |
|  |  |
| Fully Sequential Calculation Results (for-loop) | Distributed Calculations Results (parfor-loop) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Value of n** | **Execution Time – Fully Sequential** | **Execution Time – Distributed** | **Performance improvement** |
| 1000000 | 0.0111 seconds | 0.1276 seconds | 1049.5% slower, when using parfor. |
| 1000000000 | 2.1387 seconds | 1.5137 seconds | 40.8% faster, when using parfor |
| 10000000000 | 23.2253 seconds | 14.1662 seconds | 39% faster, when using parfor. |

**Takes**:

The above comparison is quite interesting.

1. One can see that for the lowest value of n, the calculation takes significantly longer time to complete when calculated parallelly than sequentially. This might be because the calculation is not resource hungry that when calculated parallelly that by trying to perform parallel calculation, MATLAB introduces an overhead in dividing the work to various workers, which then results in the time increase compared to a simple for-loop. Note that this is only observed for a lower value of n.
2. For the rest of the values of n, performing the calculations parallelly is ~40% faster than performing the calculations sequentially. And this is where the advantage of parallel computing is observed.