Faculty of Electronics and Information Technology

WARSAW UNIVERSITY OF TECHNOLOGY

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

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

$$\sum_{i=1}^{n} \frac{1}{i(i+1)(i+2)} = \frac{1}{1\,.\,2\,.\,3} + \frac{1}{2\,.\,3\,.\,4} + \cdots + \frac{1}{n(n+1)(n+2)}$$

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.

$$\sum_{i=1}^{n} \frac{1}{i(i+1)(i+2)} = \frac{1}{1\,.\,2\,.\,3} + \frac{1}{2\,.\,3\,.\,4} + \cdots + \frac{1}{n(n+1)(n+2)}$$

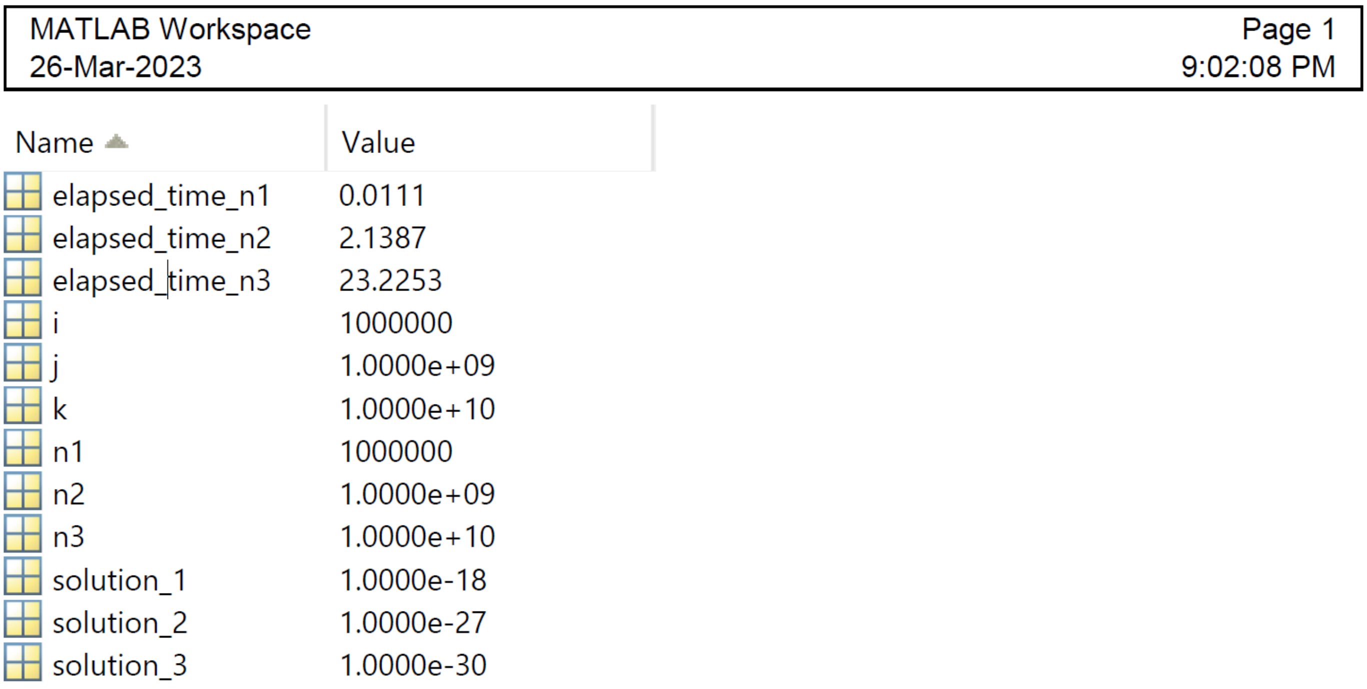
1. Fully Sequential Calculation

The MATLAB code for the fully sequential calculation is:

```
format long e

%%% Sequential Calculation n = 1000000 %%%
n1 = 1000000;
tic;
for i = 1:n1
    solution_1 = 1./(i.*(i+1).*(i+2));
end
elapsed_time_n1 = toc;
%%% END %%%

%%% Sequential Calculation n = 1000000000 %%%
n2 = 1000000000;
tic;
for j = 1:n2
    solution_2 = 1./(j.*(j+1).*(j+2));
end
elapsed_time_n2 = toc;
%%% END %%%

%%% Sequential Calculation n = 10000000000 %%%
n3 = 10000000000;
tic;
for k = 1:n3
    solution_3 = 1./(k.*(k+1).*(k+2));
end
elapsed_time_n3 = toc;
%%% END %%%

x = [n1, n2, n3];
y = [elapsed_time_n1, elapsed_time_n2, elapsed_time_n3];
plot(x,y);
```

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.



MATLAB Workspace
26-Mar-2023

Page 1
9:02:08 PM

Name	Value
elapsed_time_n1	0.0111
elapsed_time_n2	2.1387
elapsed_time_n3	23.2253
i	1000000
j	1.0000e+09
k	1.0000e+10
n1	1000000
n2	1.0000e+09
n3	1.0000e+10
solution_1	1.0000e-18
solution_2	1.0000e-27
solution_3	1.0000e-30

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.

2. Distributed Calculation

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

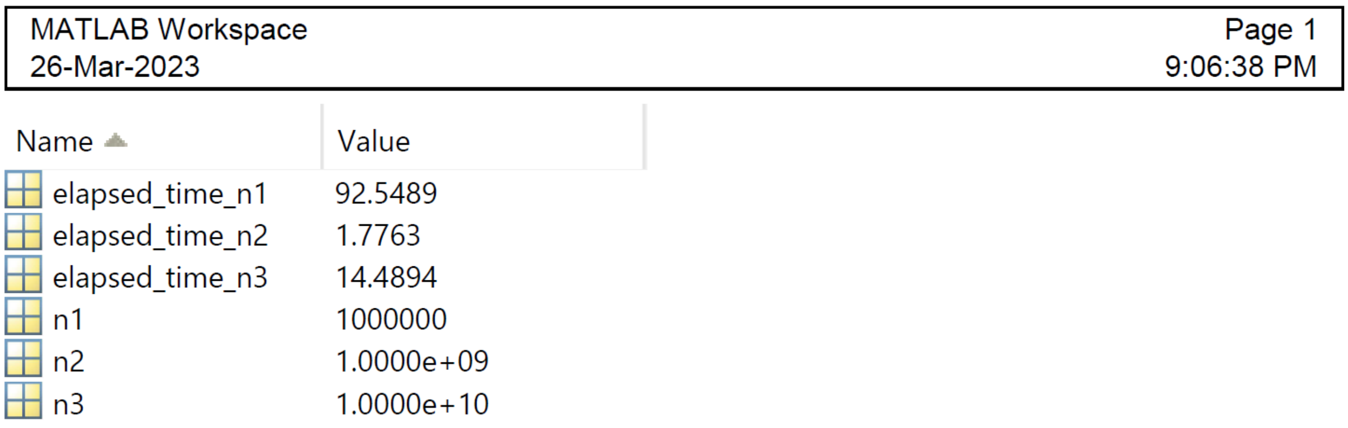
```
format long e

%%% Sequential Calculation n = 1000000 %%%
n1 = 1000000;
tic;
parfor i = 1:n1
    solution_1 = 1./(i.*(i+1).*(i+2));
end
elapsed_time_n1 = toc;
%%% END %%%

%%% Sequential Calculation n = 1000000000 %%%
n2 = 1000000000;
tic;
parfor j = 1:n2
    solution_2 = 1./(j.*(j+1).*(j+2));
end
elapsed_time_n2 = toc;
%%% END %%%

%%% Sequential Calculation n = 10000000000 %%%
n3 = 10000000000;
tic;
parfor k = 1:n3
    solution_3 = 1./(k.*(k+1).*(k+2));
end
elapsed_time_n3 = toc;
%%% END %%%

x = [n1, n2, n3];
y = [elapsed_time_n1, elapsed_time_n2, elapsed_time_n3];
plot(x,y);
```

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.

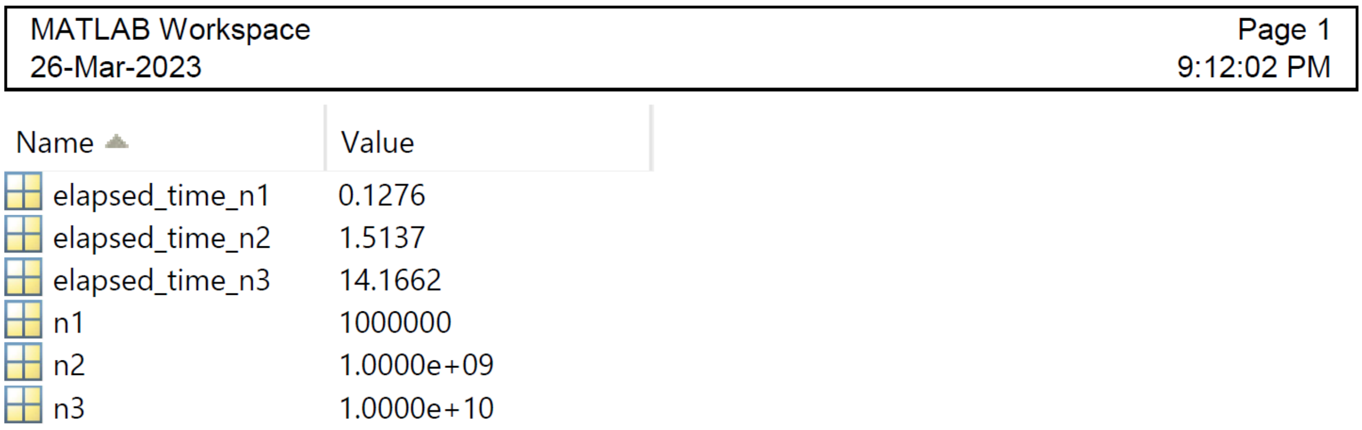


MATLAB Workspace — Page 1
26-Mar-2023 — 9:06:38 PM

Name	Value
elapsed_time_n1	92.5489
elapsed_time_n2	1.7763
elapsed_time_n3	14.4894
n1	1000000
n2	1.0000e+09
n3	1.0000e+10

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:



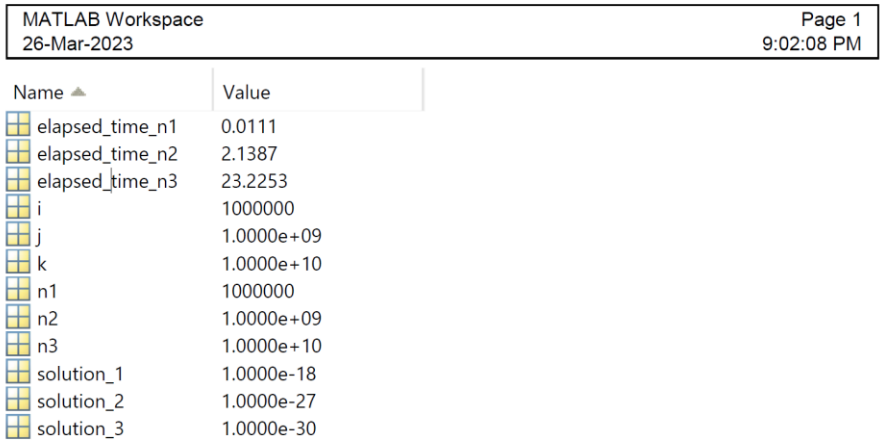
MATLAB Workspace — Page 1
26-Mar-2023 — 9:12:02 PM

Name	Value
elapsed_time_n1	0.1276
elapsed_time_n2	1.5137
elapsed_time_n3	14.1662
n1	1000000
n2	1.0000e+09
n3	1.0000e+10

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.

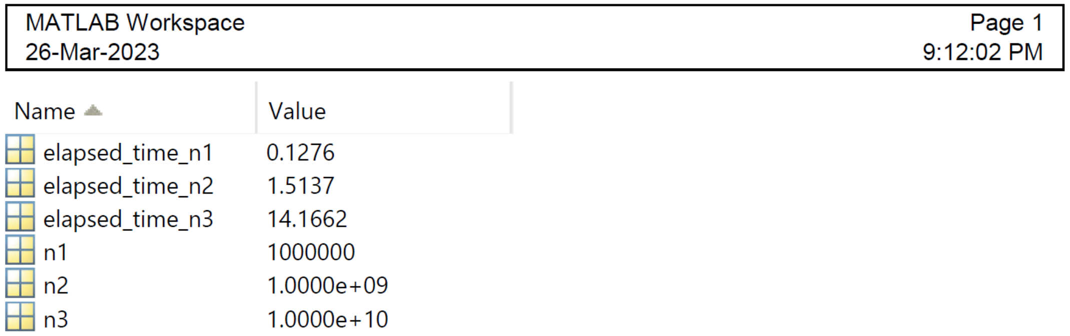
3. Fully Sequential vs Distributed Calculations

Results comparison:



```
MATLAB Workspace                                   Page 1
26-Mar-2023                                    9:02:08 PM

Name            Value
elapsed_time_n1 0.0111
elapsed_time_n2 2.1387
elapsed_time_n3 23.2253
i               1000000
j               1.0000e+09
k               1.0000e+10
n1              1000000
n2              1.0000e+09
n3              1.0000e+10
solution_1      1.0000e-18
solution_2      1.0000e-27
solution_3      1.0000e-30
```

Fully Sequential Calculation Results (for-loop)



```
MATLAB Workspace                                   Page 1
26-Mar-2023                                    9:12:02 PM

Name            Value
elapsed_time_n1 0.1276
elapsed_time_n2 1.5137
elapsed_time_n3 14.1662
n1              1000000
n2              1.0000e+09
n3              1.0000e+10
```

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.