Stephanie Galvan

Parallel Computing

MW 4:30PM – 5:50PM

Lab 1: Serial Matrix Multiplication

Description:

This first lab consisted in the creation of a serial matrix program that will later be compared to the results of the same problem but with a parallel approach. In order to simplify programming, all matrices are assumed to be square and some utility functions were provided to help in the finalization of the program.

Assumptions and Constraints:

- Only two matrices will be multiplied at a time.
- No APIs allowed.
- All matrices being multiplied will be squares.
- A matrix will be composed of one single value.
- The given python can be used to finish the assignment.
- A small section of the matrix must be printed to verify answers.

Approach:

First, since the matrixes are expected to be square and composed of one single value each; a function called *enhanced_multiply_matrix* was created in which only the first row of matrix A and the first column of matrix B were used for finding the value of the resultant matrix, and then a utility function was used to generate the resultant matrix. However, this approach was quite fast (around 0.2 seconds); thus, a second approach was needed.

The final approach for the program consisted of going through each row and column in order to properly multiply matrix A and matrix B. This was accomplished with the use of three nested loops. This approach took around 4 to 6 minutes to run given the default values. Although the timing seems excessive, this is necessary in order to compare these results with parallel programming which is predicted to be faster than this approach.

Instructions and Conclusion:

To use the default values, the python file can be run normally through the terminal or an IDE. However, the default values are 1024 for size and 1 for value, and thus the program will take between 4 to 6 minutes to run. This slow approach was chosen in order to generate longer time periods which would allow a better distinction between serial programs and parallel programming. An example of running the file with default values is shown below:

```
"C:\Users\f\Documents\CS\Parallel Computing\SerialMatrixMultiply\venv\Scripts\python.exe"
1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1


1025 1025 1025 1025 1025 1025 1025 1025 1025
1025 1025 1025 1025 1025 1025 1025 1025 1025
1025 1025 1025 1025 1025 1025 1025 1025 1025
1025 1025 1025 1025 1025 1025 1025 1025 1025
1025 1025 1025 1025 1025 1025 1025 1025 1025
1025 1025 1025 1025 1025 1025 1025 1025 1025
1025 1025 1025 1025 1025 1025 1025 1025 1025
1025 1025 1025 1025 1025 1025 1025 1025 1025
1025 1025 1025 1025 1025 1025 1025 1025 1025
Total time:
307.2985532283783 seconds
```

Moreover, for customized values, the program can be run by defining the values on the terminal as below. For this example, the size of the matrices was selected as 500 and the value was 2. As shown on the terminal, the program took around 36 seconds to run with those values.

```
C:\Users\f\Documents\CS\Parallel Computing\SerialMatrixMultiply\SerialMatrixMultiply>MultiplyMatrix.py -s 500 -v 2
2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2

2001 2001 2001 2001 2001 2001 2001 2001 2001
2001 2001 2001 2001 2001 2001 2001 2001 2001
2001 2001 2001 2001 2001 2001 2001 2001 2001
2001 2001 2001 2001 2001 2001 2001 2001 2001
2001 2001 2001 2001 2001 2001 2001 2001 2001
2001 2001 2001 2001 2001 2001 2001 2001 2001
2001 2001 2001 2001 2001 2001 2001 2001 2001
2001 2001 2001 2001 2001 2001 2001 2001 2001
2001 2001 2001 2001 2001 2001 2001 2001 2001
Total time:
36.16491508483887 seconds
```